Real-time Scheduling

Introduction to Real-Time

A Robotic Design Example (Revisited)

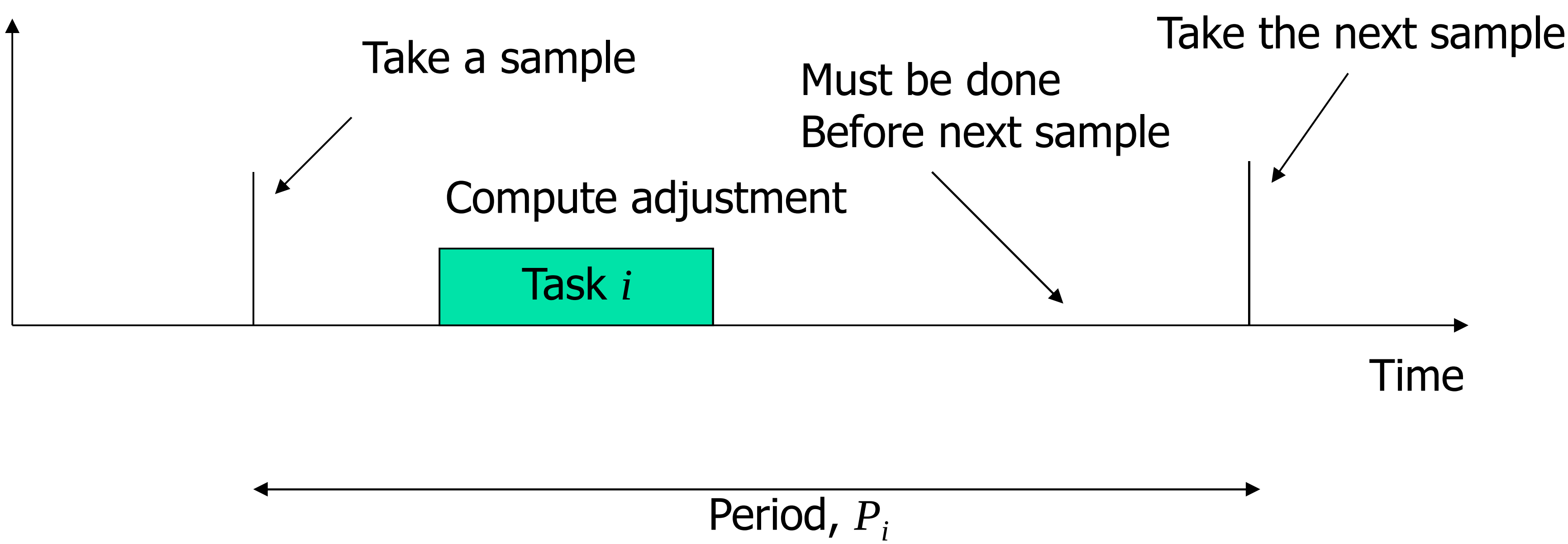
- A robot has a camera that detects obstacles with probability 70%, a bump sensor that detects imminent collisions with a probability of 99.9% (when an obstacle is 1 inch away), and a cliff sensor that detects imminent falls off a cliff with a probability of 99.9% (when the cliff is 1 inch away). The robot has breaks that can stop it within 0.1 second. The mission is to deliver supplies from point A to point B, safely.
  - What are safety-critical requirements?
  - What are mission-critical (i.e., performance) requirements?
  - What is a safe state?
  - How to ensure well-formed dependencies?
  - What is a safe speed for the robot?
  - Is the algorithm that computes speed based on preferred arrival time and route safety-critical or mission-critical?

The Schedulability Question: Drive-by-Wire Example

- Consider a control system in an autonomous robot
  - Navigation guidance is computed every 10 ms – wheel positions adjusted accordingly (computing the adjustment takes 4.5 ms of CPU time)
  - Threats and obstacles are reassessed every 4 ms – breaks adjusted accordingly (computing the adjustment takes 2ms of CPU time)
  - Optimal speed is computed every 15 ms – robot speed is adjusted accordingly (computing the adjustment takes 0.45 ms)
  - For safe operation, adjustments must always be computed before the next sample is taken
- Is it possible to always compute all adjustments in time?

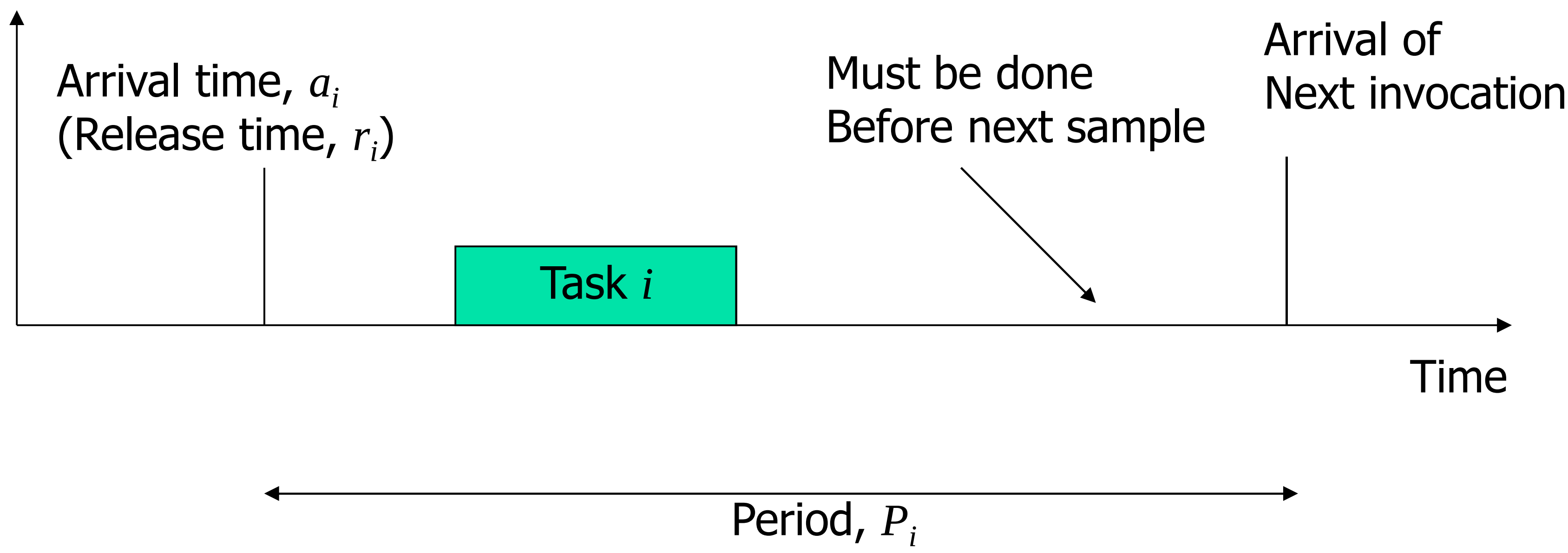
Some Terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



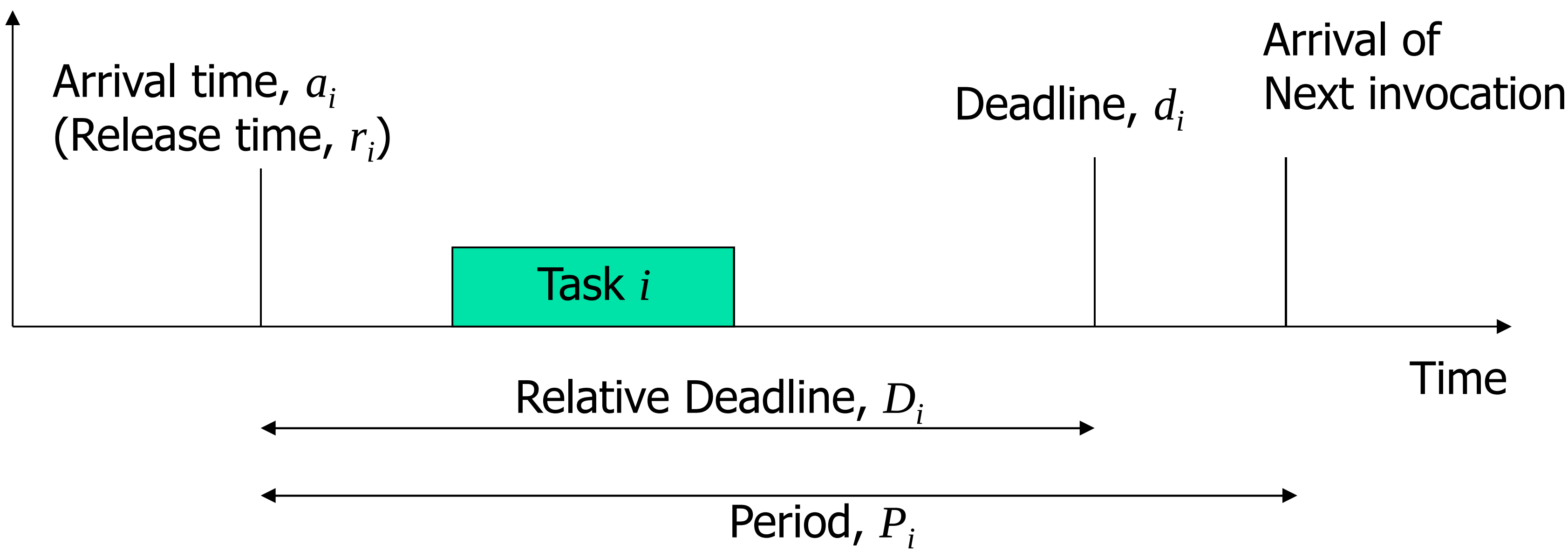
Some Terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



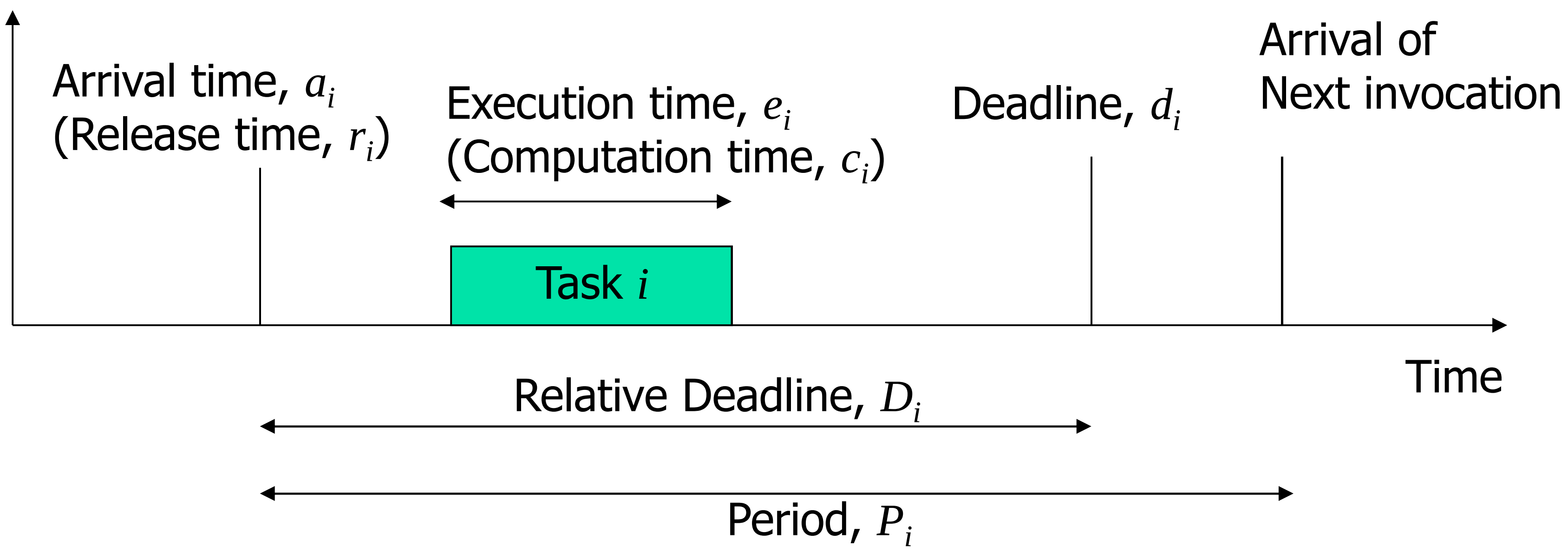
Some Terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



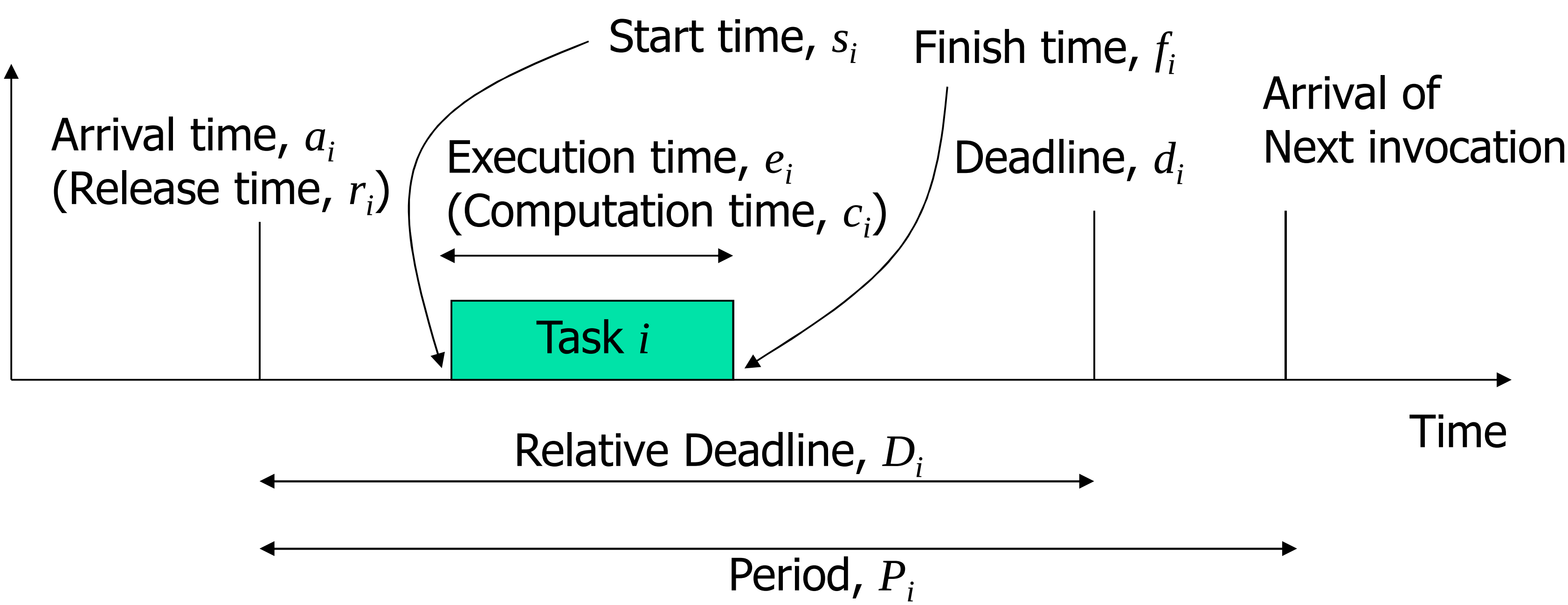
Some Terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



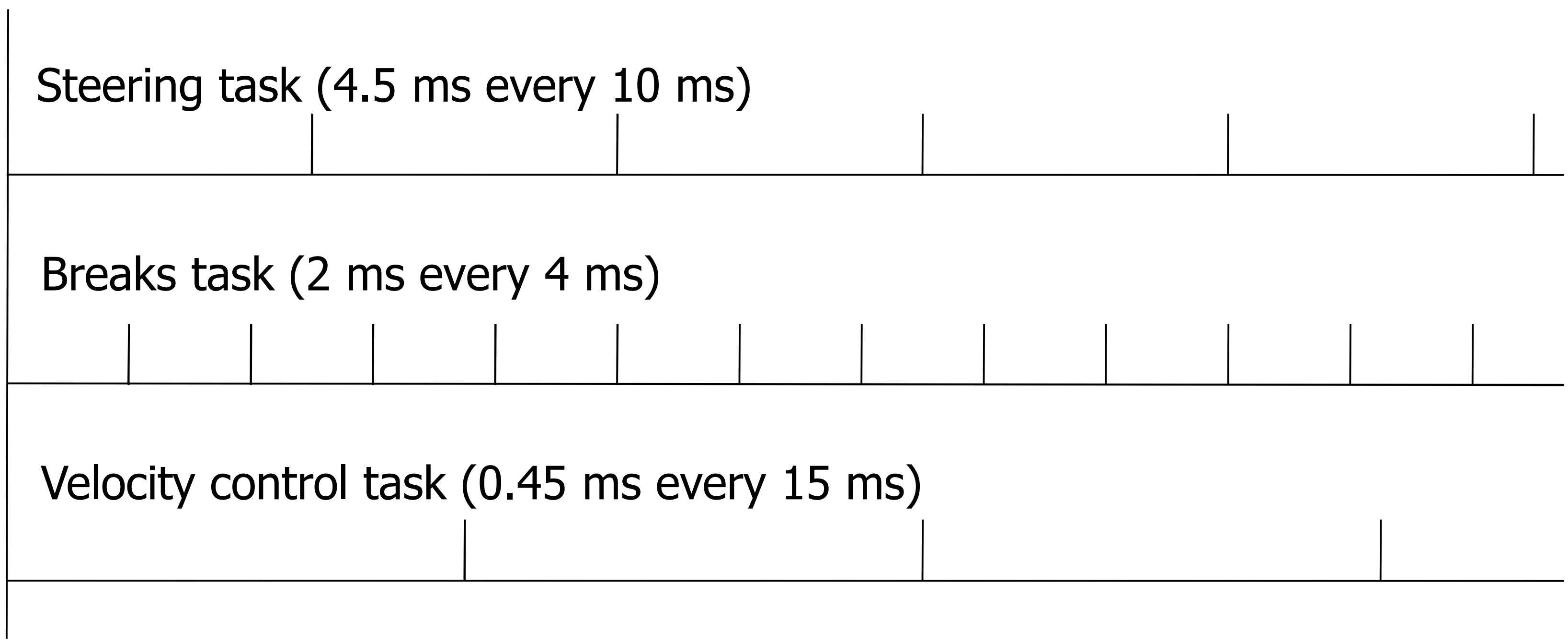
Some Terminology

- Tasks, periods, arrival-time, deadline, execution time, etc.



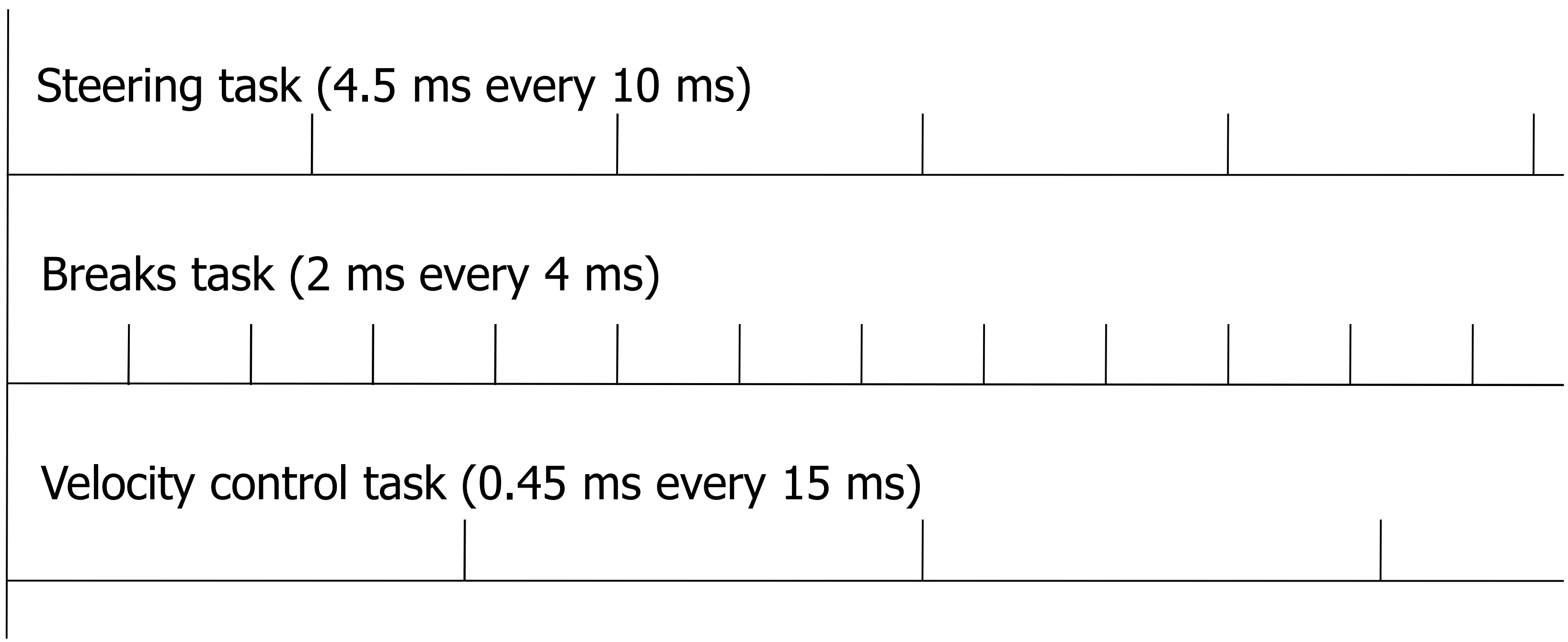
Back to Drive-by-Wire Example

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



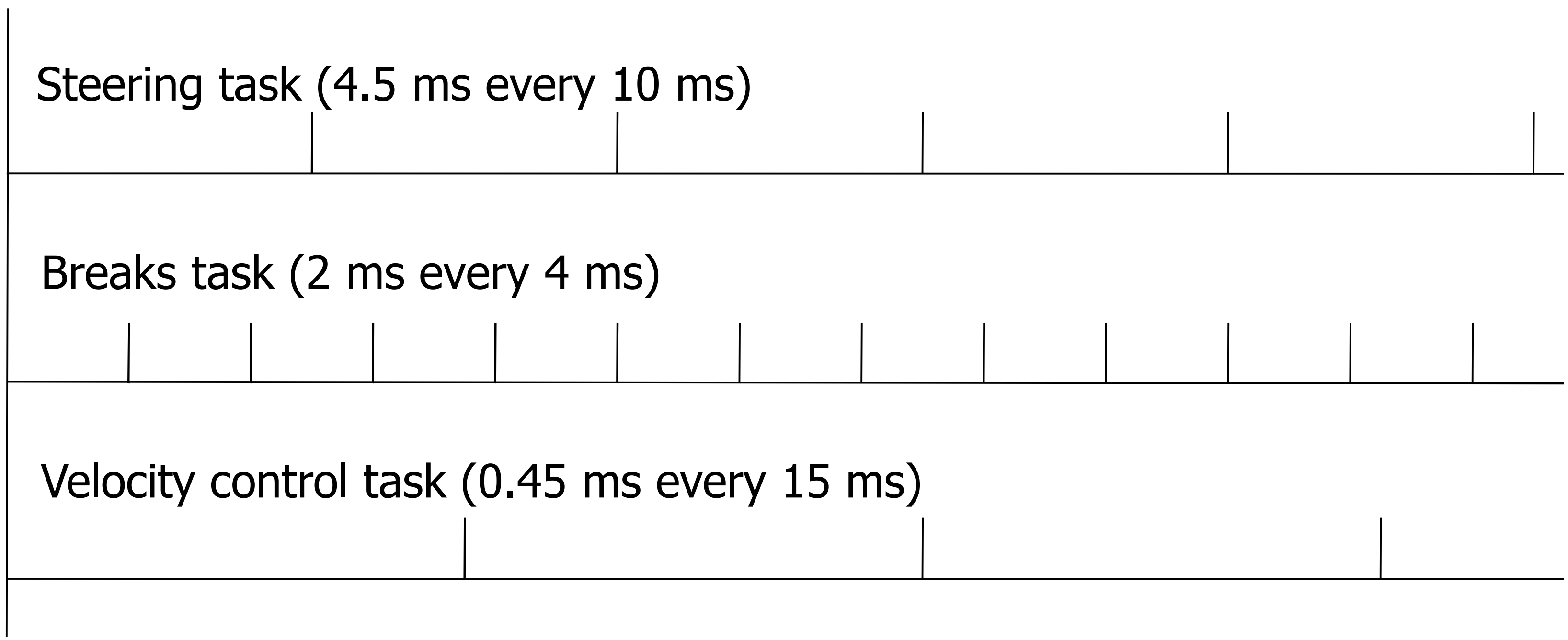
Back to
Drive-by-Wire Example

- Sanity check #1: Is the processor over-utilized? (e.g., if you have 5 homeworks due this time tomorrow, each takes 6 hours, then 5x6 = 30 > 24 → you are overutilized)



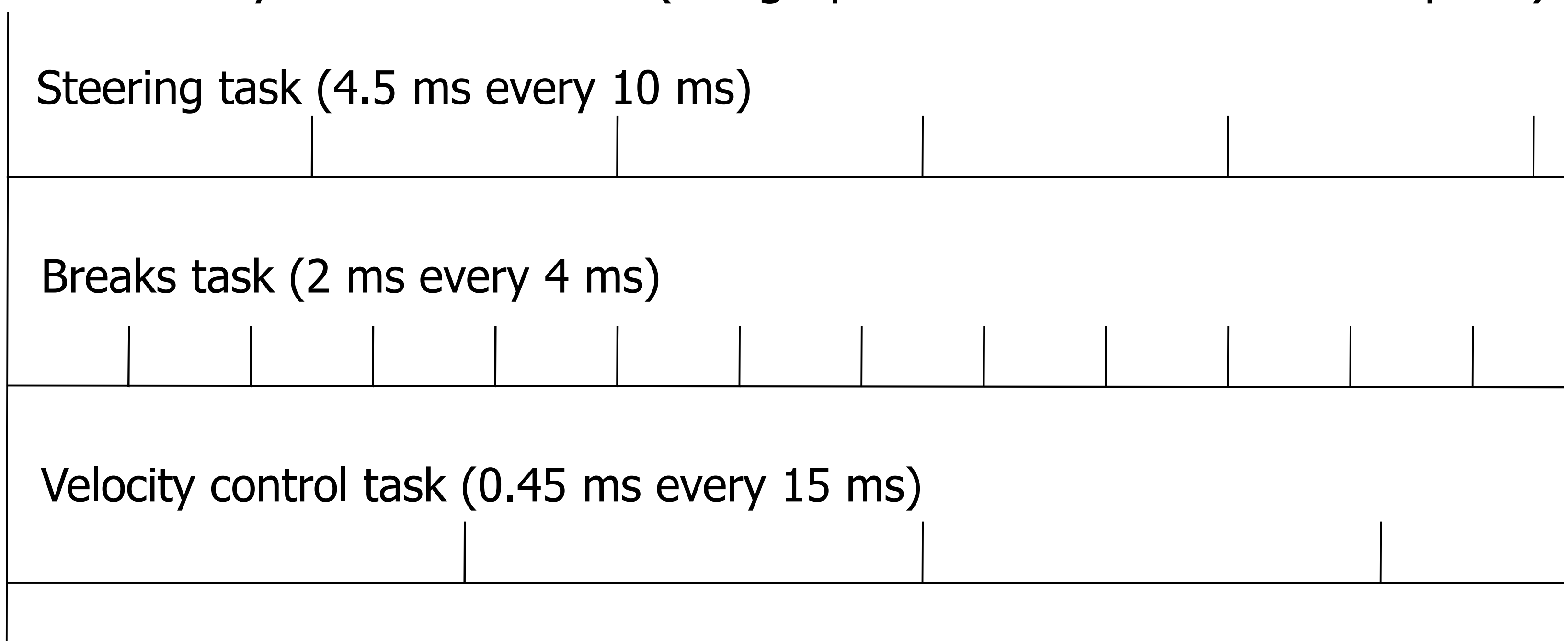
Back to
Drive-by-Wire Example

- Sanity check #1: Is the processor over-utilized? (e.g., if you have 5 homeworks due this time tomorrow, each takes 6 hours, then 5x6 = 30 > 24 → you are overutilized)
  - Hint: Check if processor utilization > 100%



Task Scheduling

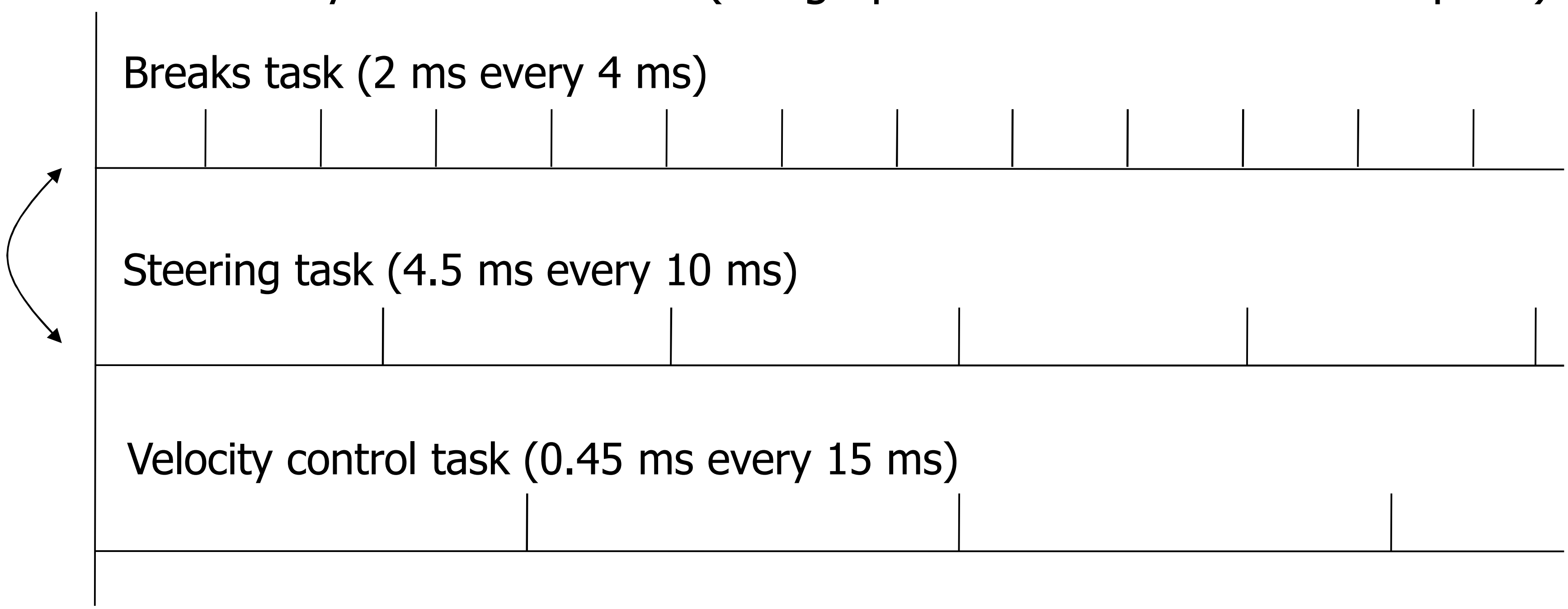
- Decision #1: In what order should tasks be executed?
  - Hand-crafted schedule (fill timeline by hand)
  - Priority based schedule (assign priorities → schedule is implied)



How to assign priorities to tasks?

Task Scheduling

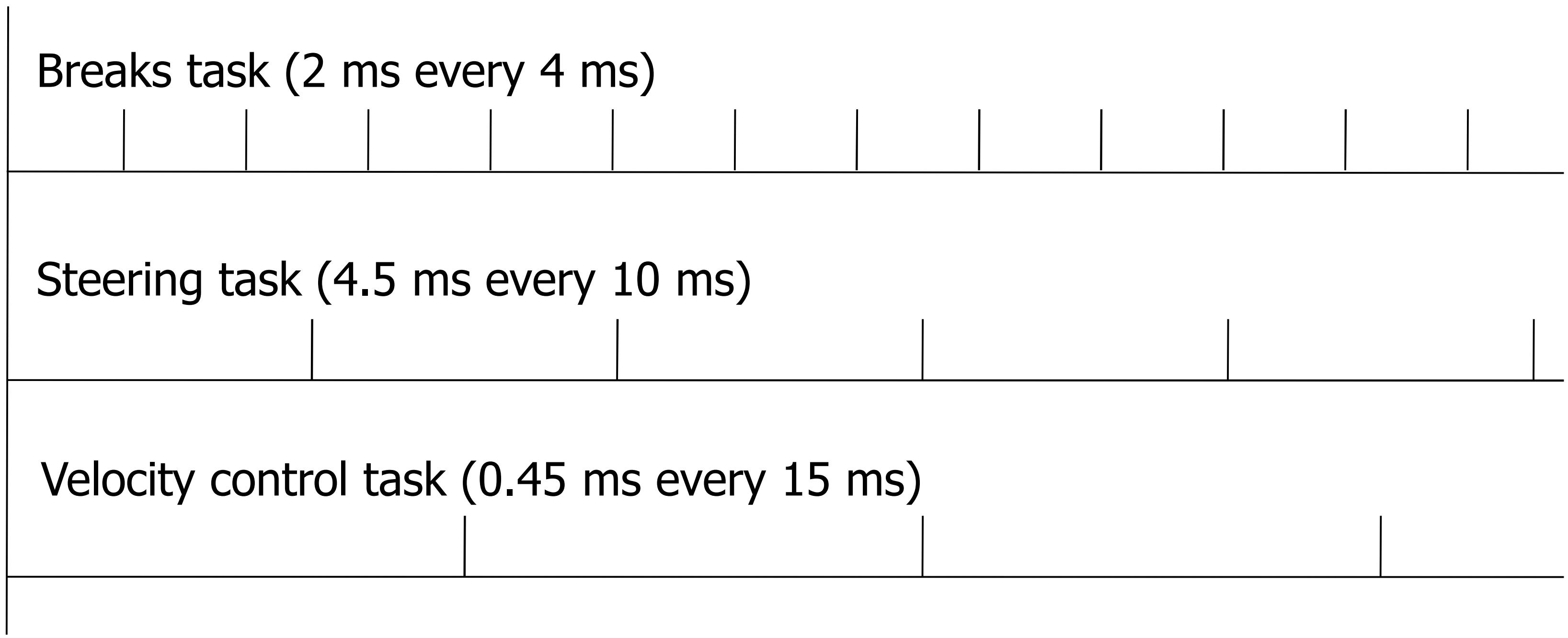
- Decision #1: In what order should tasks be executed?
  - Hand-crafted schedule (fill timeline by hand)
  - Priority based schedule (assign priorities → schedule is implied)



Intuition: Urgent tasks should be higher in priority

Task Scheduling

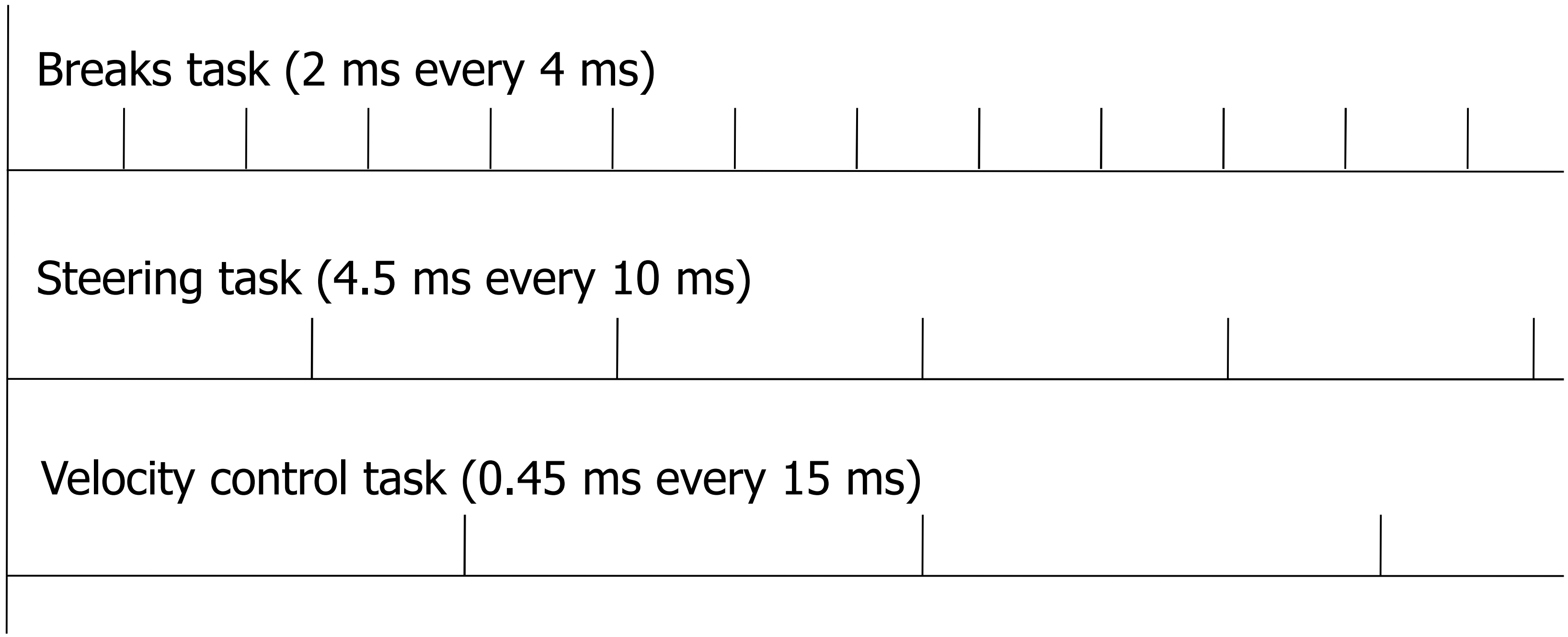
- Decision #2: Preemptive versus non-preemptive?
  - Preemptive: Higher-priority tasks can interrupt lower-priority ones
  - Non-preemptive: They can’t



In this example, will non-preemptive scheduling work?

Task Scheduling

- Decision #2: Preemptive versus non-preemptive
  - Preemptive: Higher-priority tasks can interrupt lower-priority ones
  - Non-preemptive: They can't

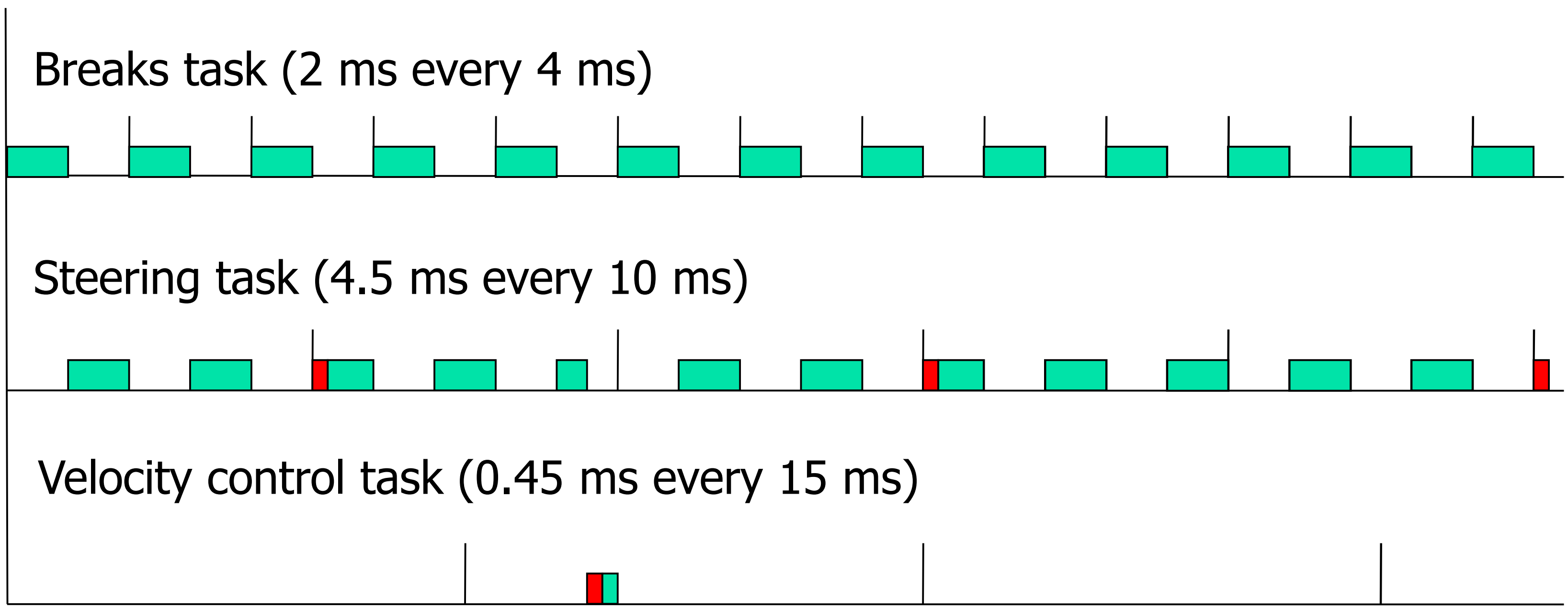


In this example, will non-preemptive scheduling work?

- Hint: Compare relative deadlines of tasks to execution times of others

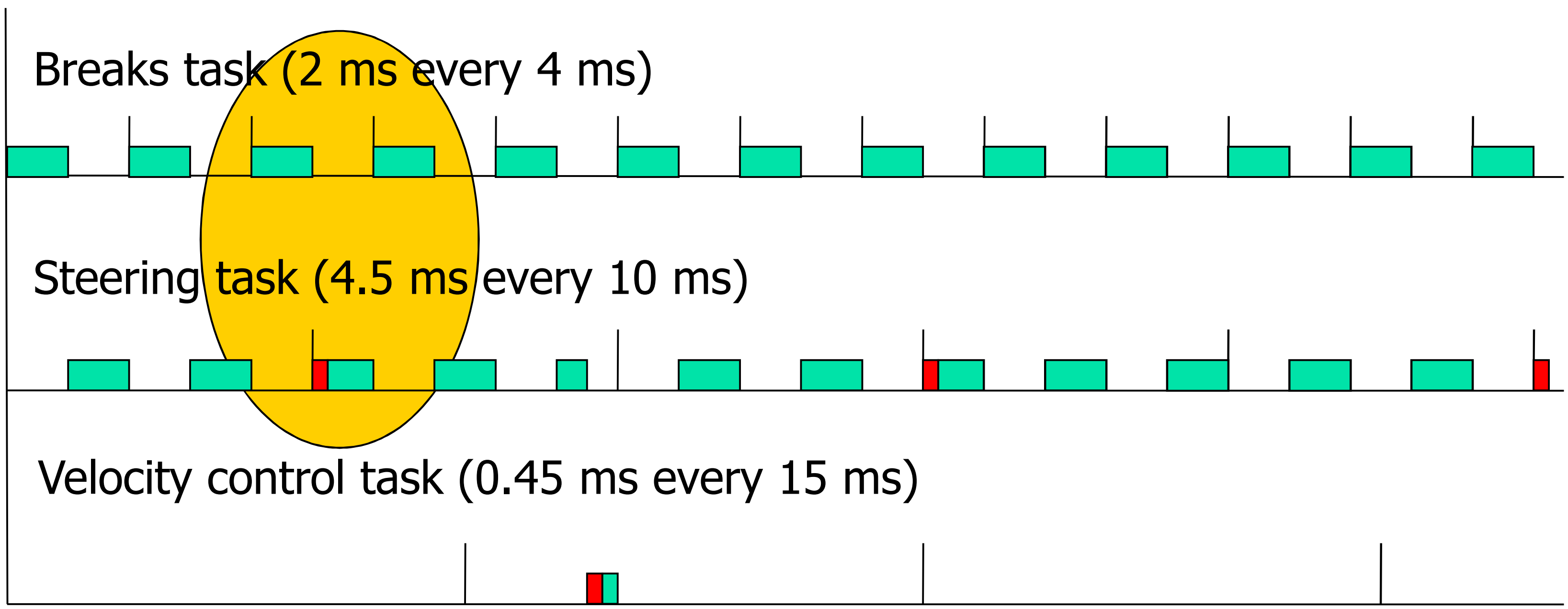
Timeline

- Deadlines are missed!
- Average Utilization < 100%



Timeline

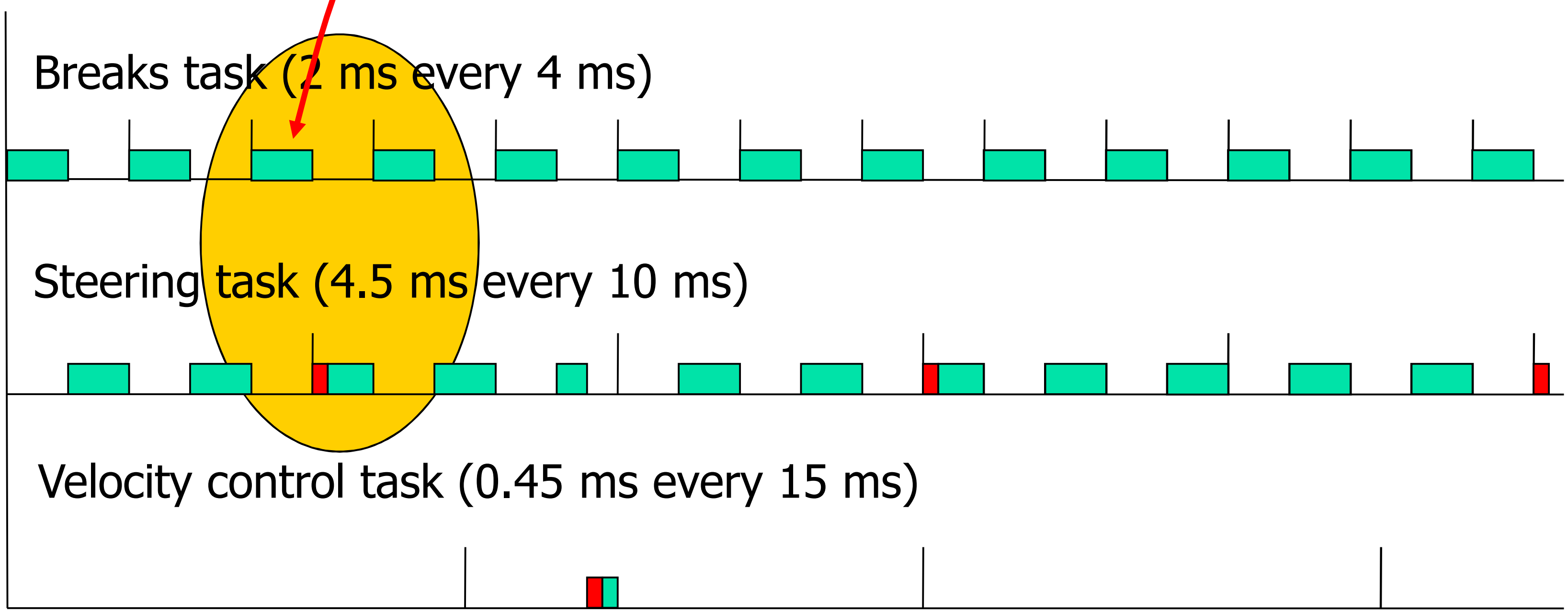
- Deadlines are missed!
- Average Utilization < 100%



Timeline

Fix:
Give this task invocation a lower priority

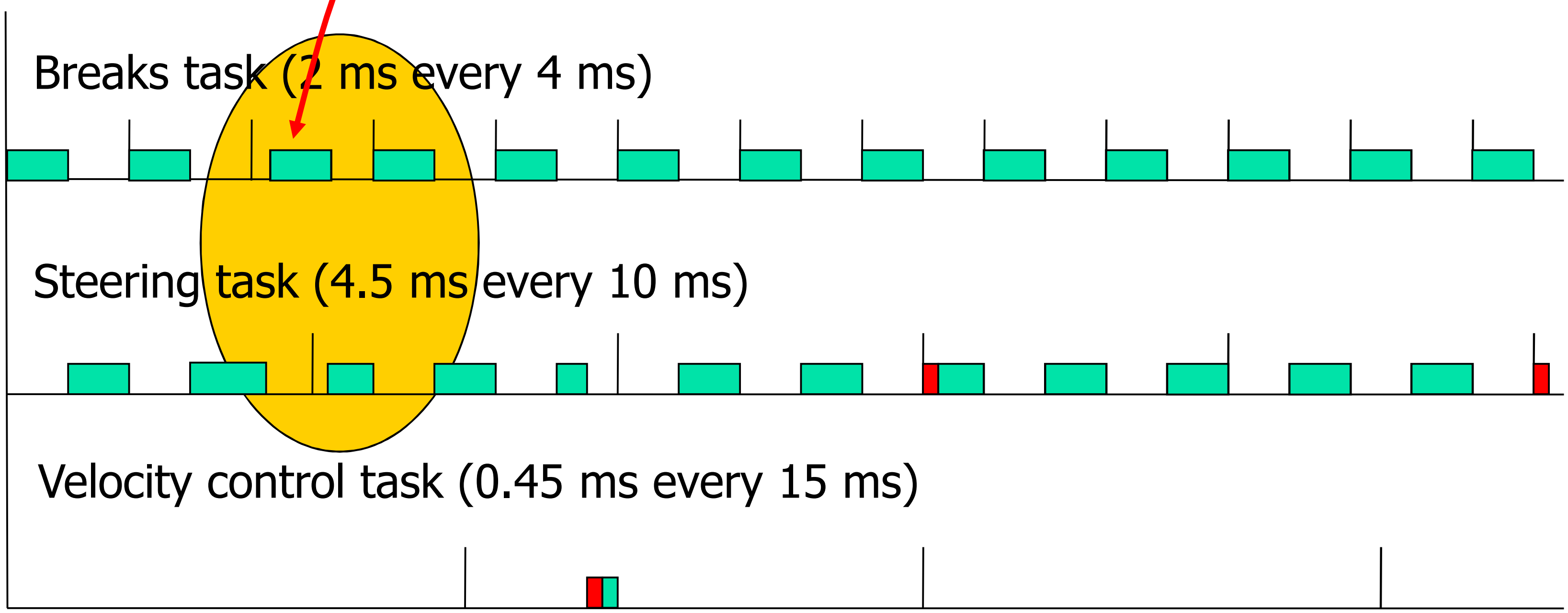
- Deadlines are missed!
- Average Utilization < 100%



Timeline

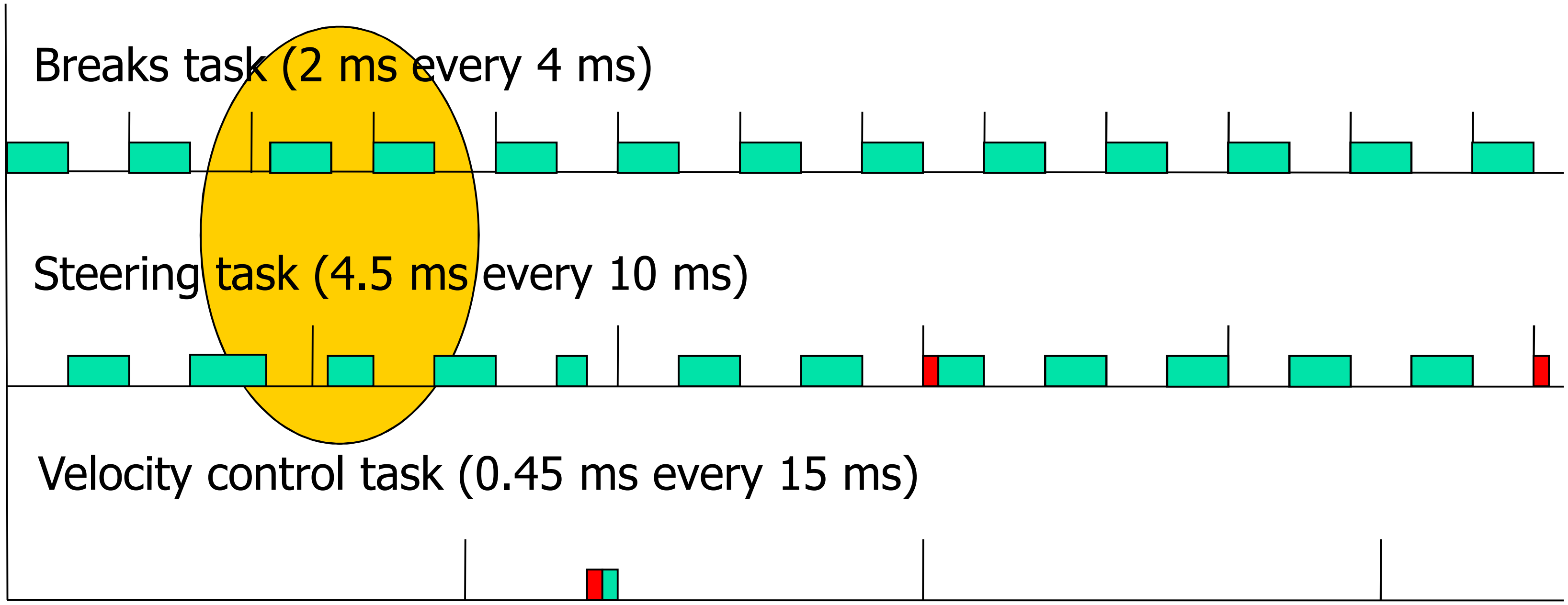
Fix:
Give this task invocation
a lower priority

- Deadlines are missed!
- Average Utilization < 100%



Task Scheduling

- Decision #3: 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

Interesting Questions

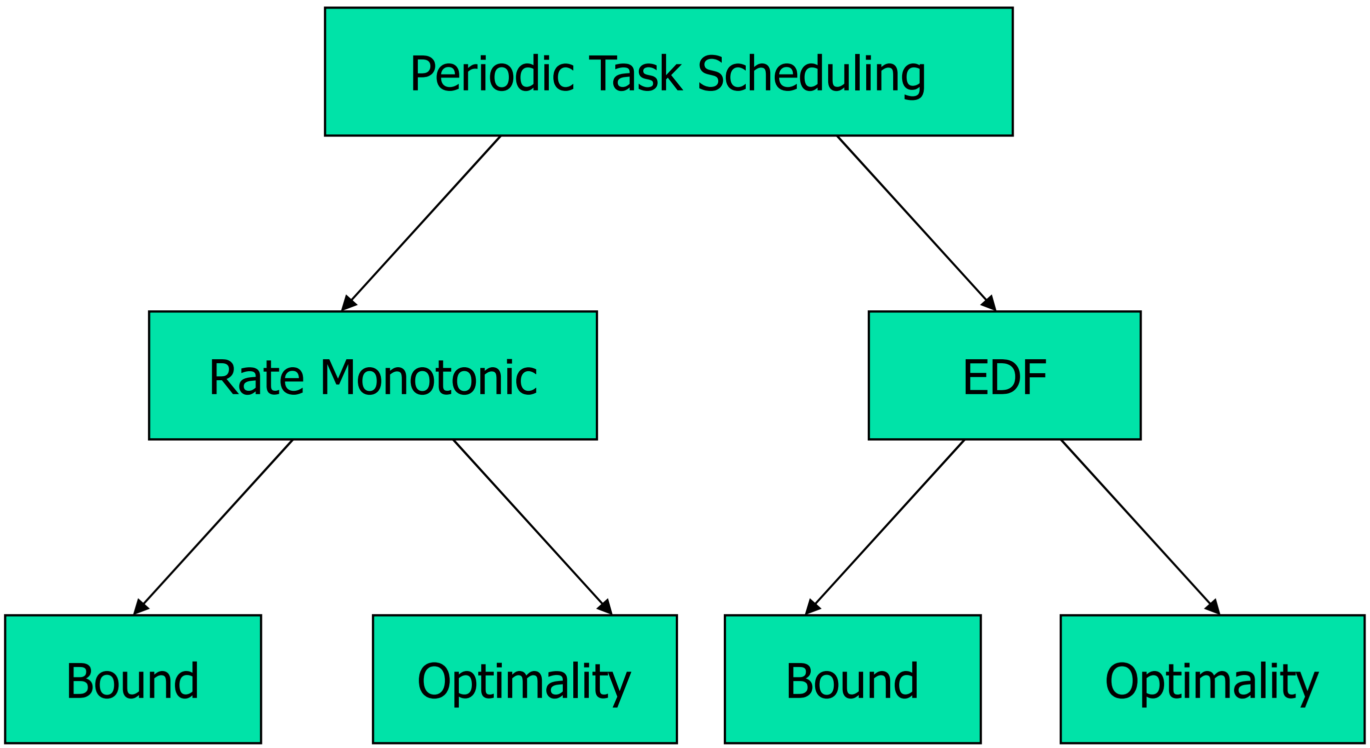
- What is the optimal dynamic priority scheduling policy? (Optimal: meets all deadlines as long as any other policy in its class can)
  - Can it meet all deadlines as long as the processor is not over-utilized?
- What is the optimal static priority scheduling policy?
  - When can it meet all deadlines?
  - Can it meet all deadline as long as the processor is not over-utilized?

Interesting Questions

- What is the optimal dynamic priority scheduling policy? (Optimal: meets all deadlines as long as any other policy in its class can)
  - Can it meet all deadlines as long as the processor is not over-utilized?
- What is the optimal static priority scheduling policy?
  - When can it meet all deadlines?
  - Can it meet ~~all~~ deadline as long as the processor is not over-utilized?

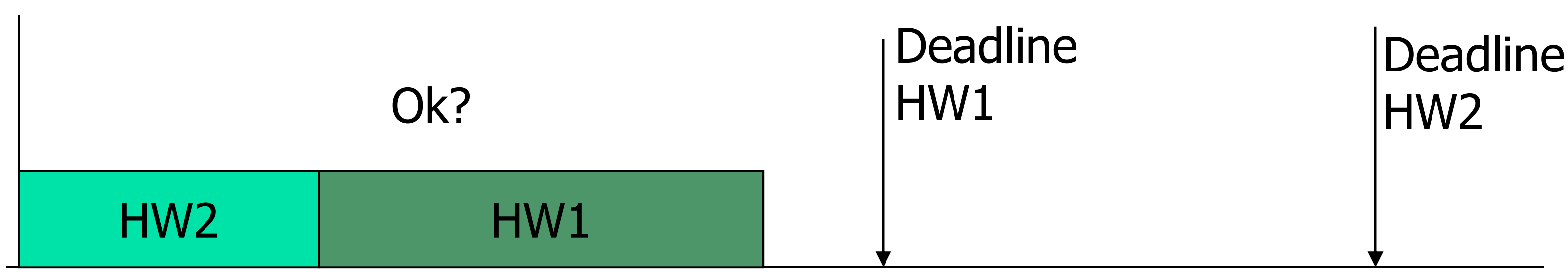
Utilization Bounds

Main Results in Real-time Scheduling of Periodic Tasks



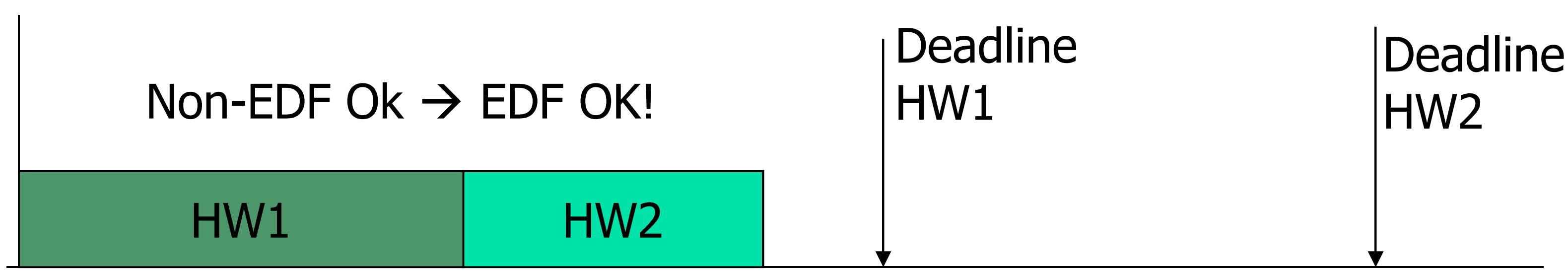
Advanced: Earliest Deadline First (EDF) Optimality Result

- EDF is the optimal dynamic priority scheduling policy
  - It can meet all deadlines whenever the processor utilization is less than 100%
  - Intuition:
    - You have HW1 due tomorrow and HW2 due the day after, which one do you do first?
    - If you started with HW2 and met both deadlines you could have started with HW1 (in EDF order) and still met both deadlines
    - EDF can meet deadlines whenever anyone else can



Earliest Deadline First (EDF) Optimality Result

- EDF is the optimal dynamic priority scheduling policy
  - It can meet all deadlines whenever the processor utilization is less than 100%
  - Intuition:
    - You have HW1 due tomorrow and HW2 due the day after, which one do you do first?
    - If you started with HW2 and met both deadlines you could have started with HW1 (in EDF order) and still met both deadlines
    - EDF can meet deadlines whenever anyone else can

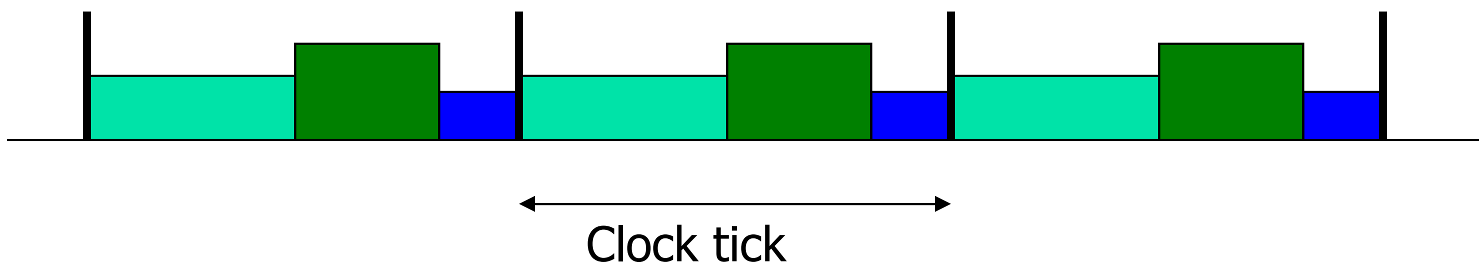


When can EDF Meet Deadlines?

- Consider a task set where:

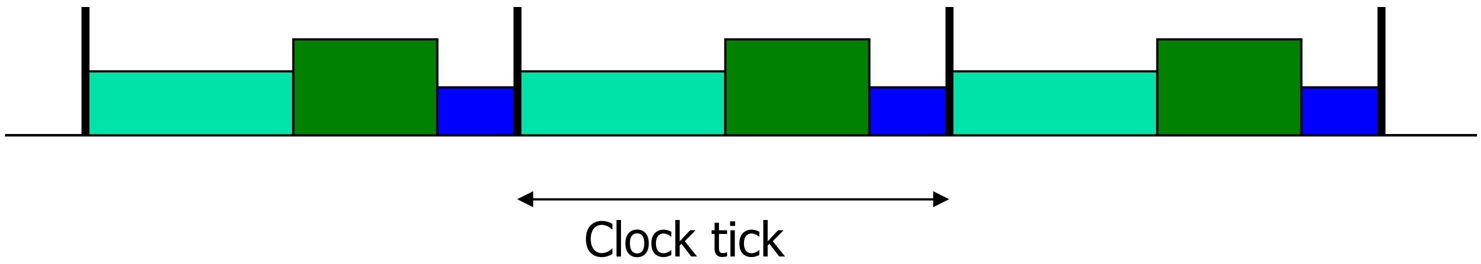
$$\sum_i \frac{C_i}{P_i} = 1$$

- Imagine a policy that reserves for each task i a fraction f_i of each clock tick, where $f_i = C_i / P_i$



Utilization Bound of EDF

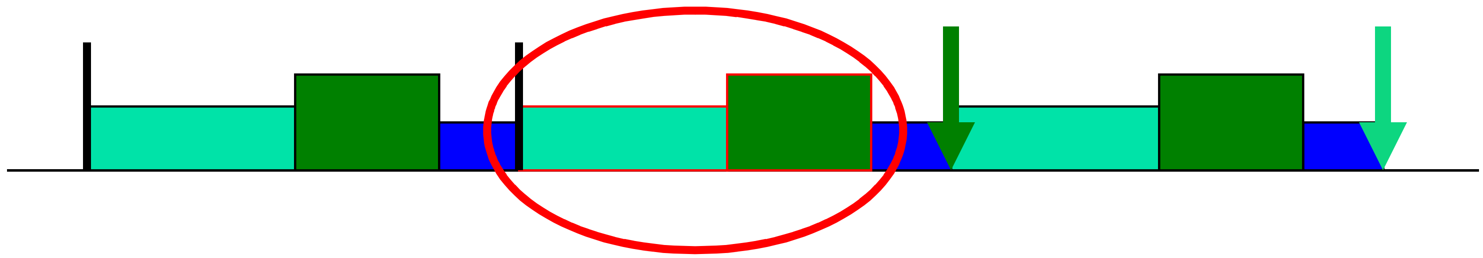
- Imagine a policy that reserves for each task i a fraction f_i of each time unit, where $f_i = C_i / P_i$



- This policy meets all deadlines, because within each period P_i it reserves for task i a total time
  - Time $= f_i \, P_i = (C_i / P_i) \, P_i = C_i$ (i.e., enough to finish)

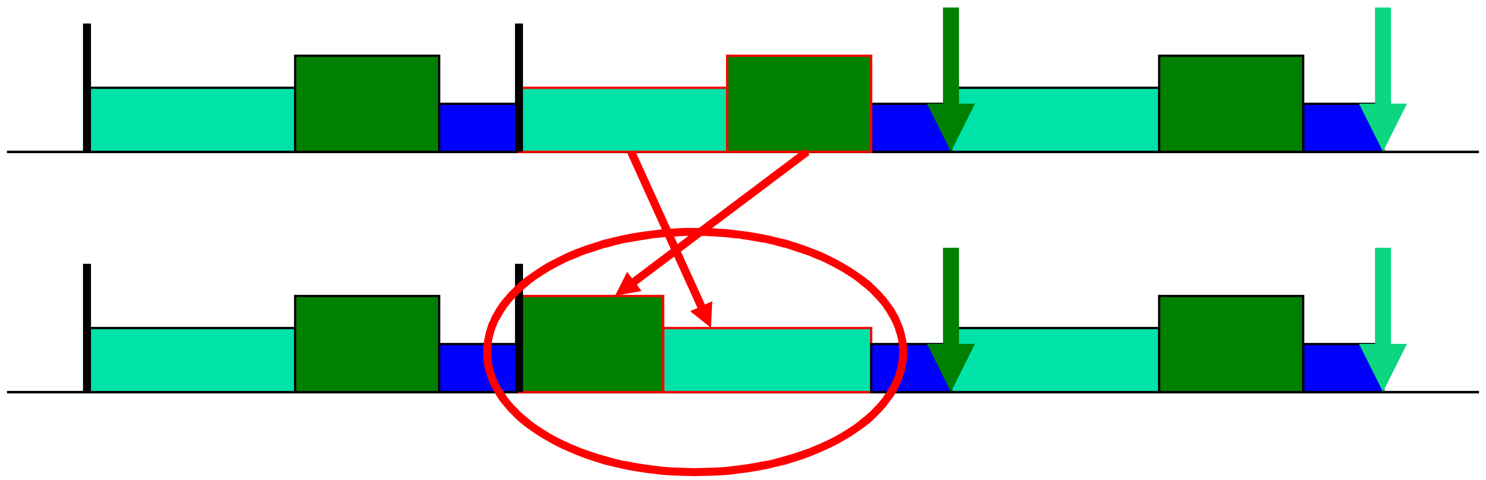
Utilization Bound of EDF

- Pick any two execution chunks that are not in EDF order and swap them



Utilization Bound of EDF

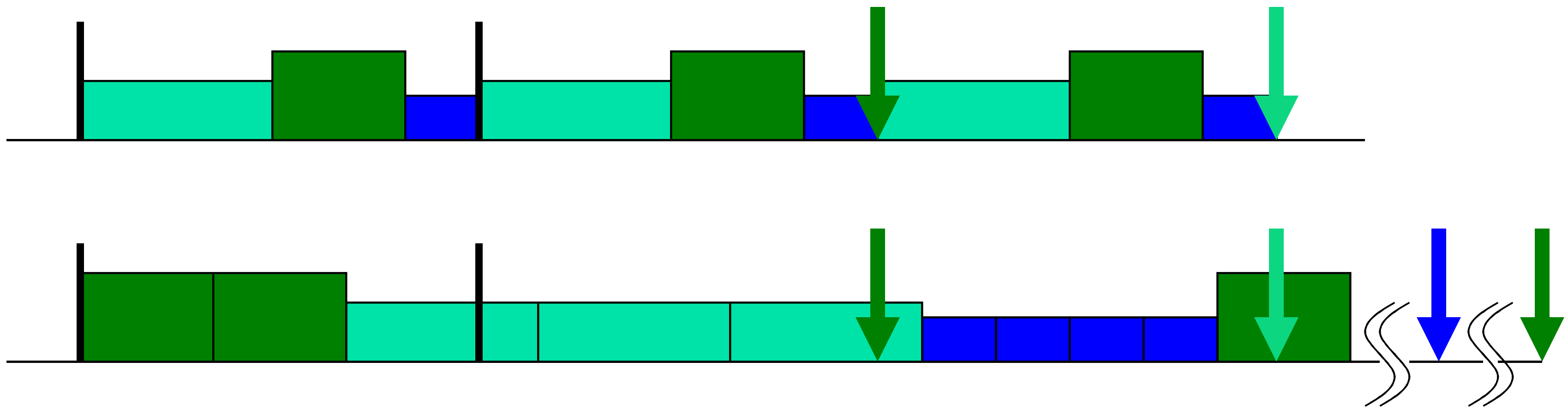
- Pick any two execution chunks that are not in EDF order and swap them



- Still meets deadlines!

Utilization Bound of EDF

- Pick any two execution chunks that are not in EDF order and swap them



- Still meets deadlines!
- Repeat swap until all in EDF order
  → EDF meets deadlines

Rate Monotonic Scheduling

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks (with period = deadline).

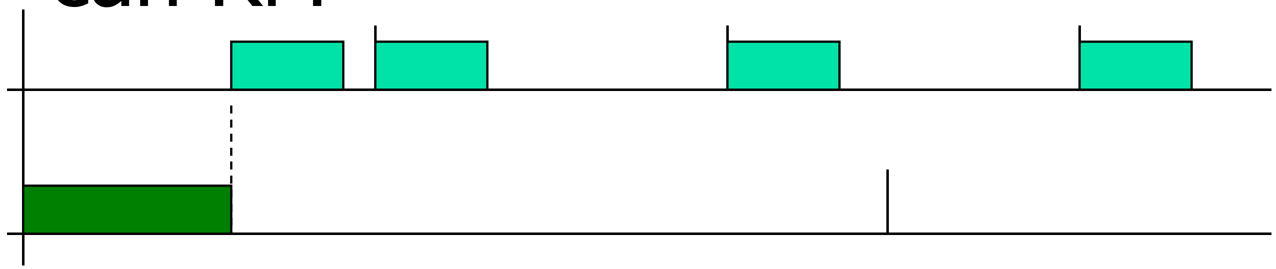
The Worst-Case Scenario

- Consider the worst case where all tasks arrive at the same time.

- If any fixed priority scheduling policy can meet deadline, rate monotonic can!

Optimality of Rate Monotonic

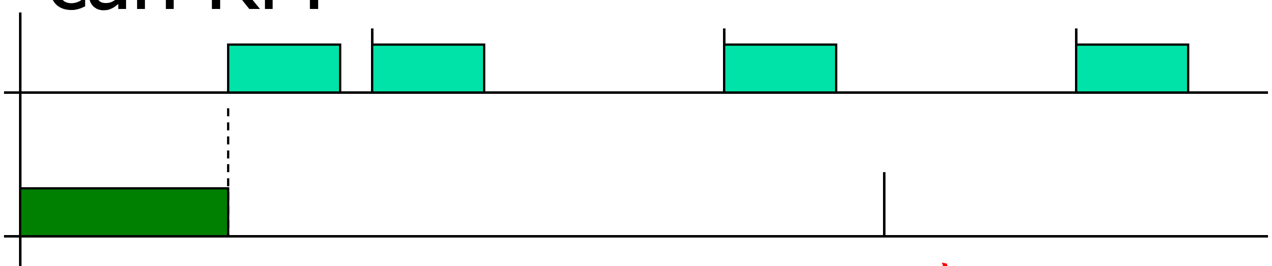
- If any other policy can meet deadlines so can RM



Policy X meets deadlines?

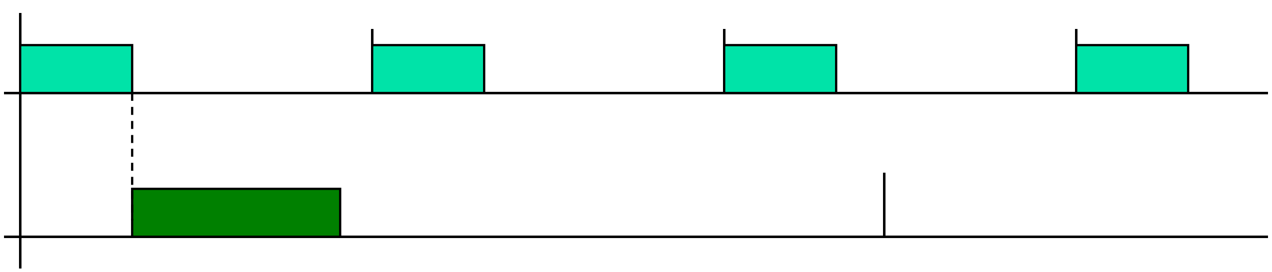
Optimality of Rate Monotonic

- If any other policy can meet deadlines so can RM



Policy X meets deadlines? YES

→ RM meets deadlines



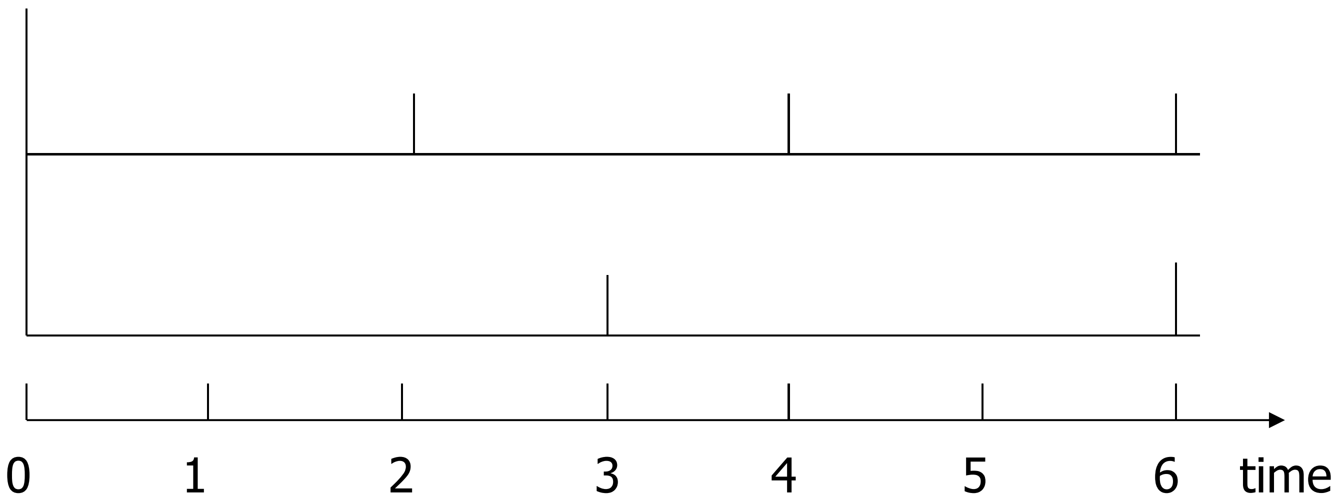
Utilization Bounds

- Intuitively:
  - The lower the processor utilization, U, the easier it is to meet deadlines.
  - The higher the processor utilization, U, the more difficult it is to meet deadlines.
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

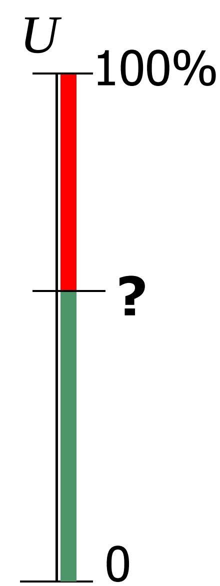
Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01



$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

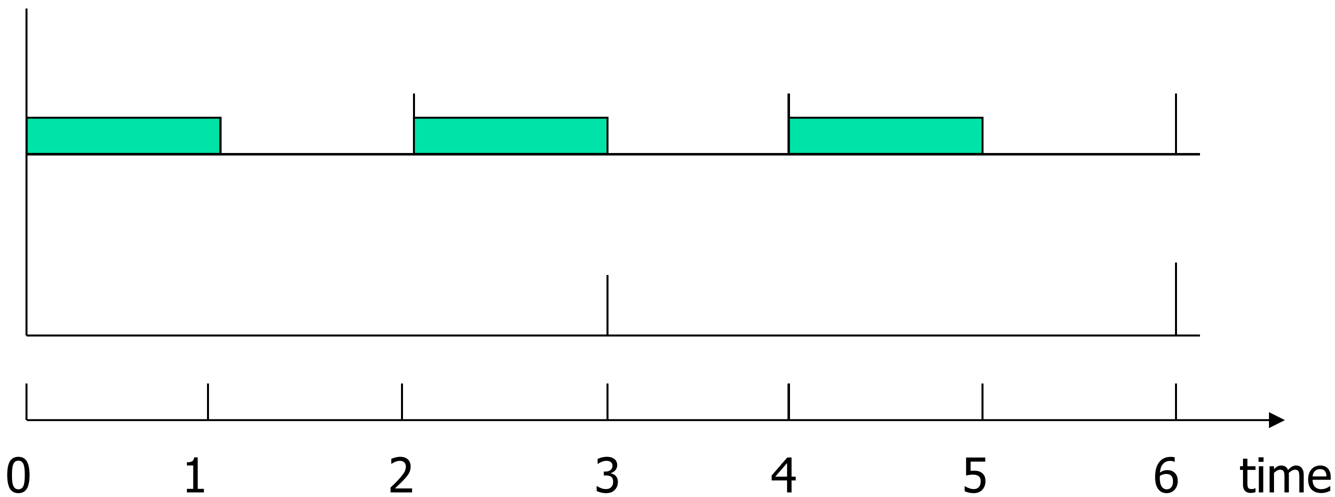


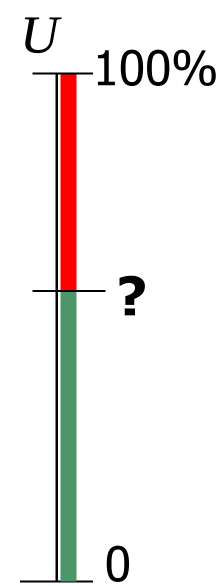
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01

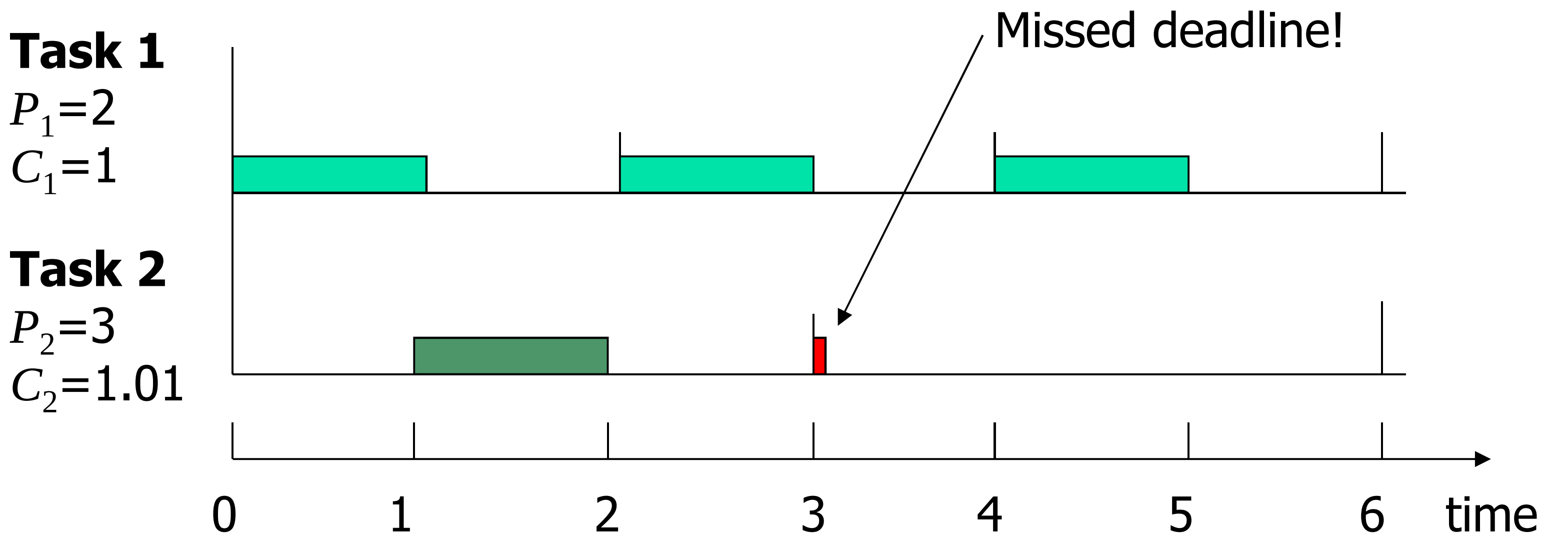




$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

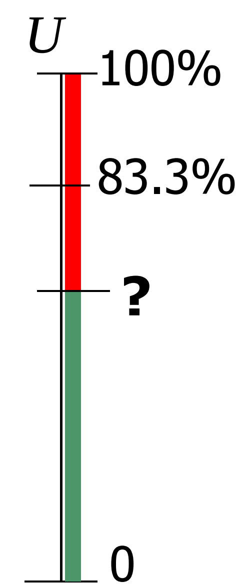
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Example (Rate-Monotonic Scheduling)



$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

Unschedulable

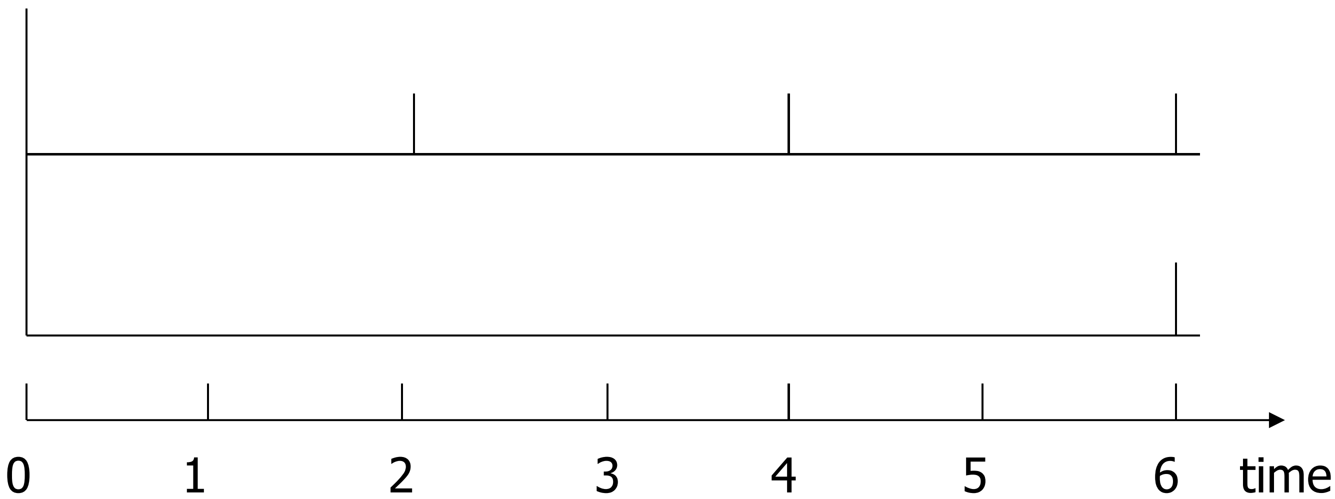


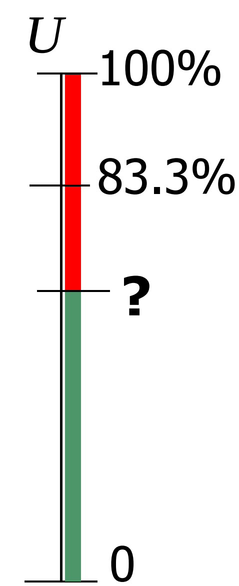
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4





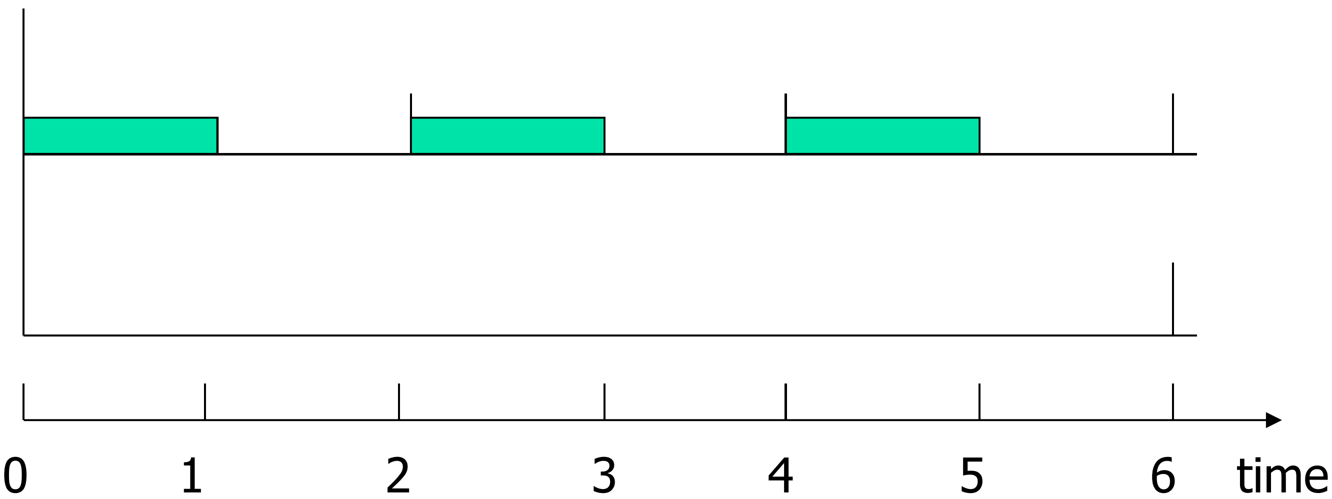
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

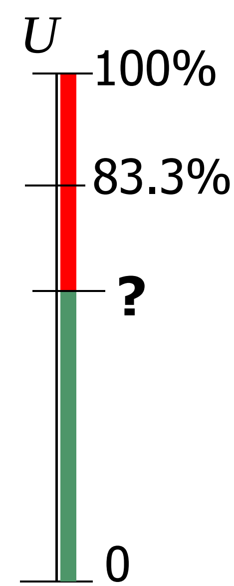
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4

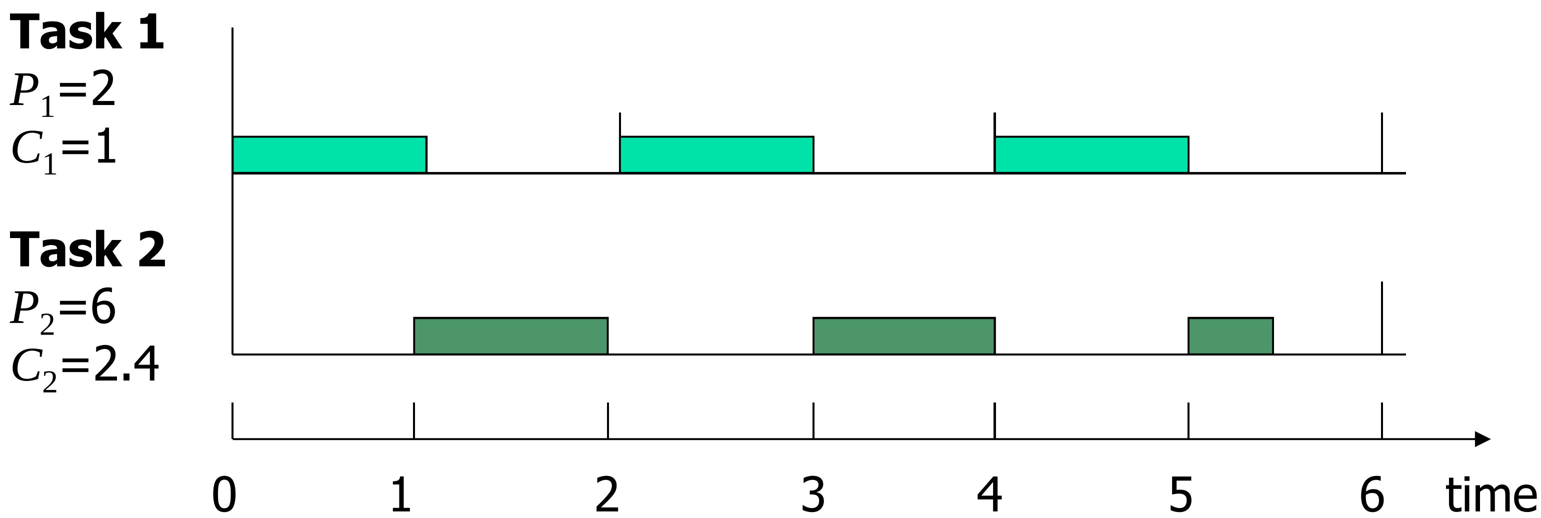


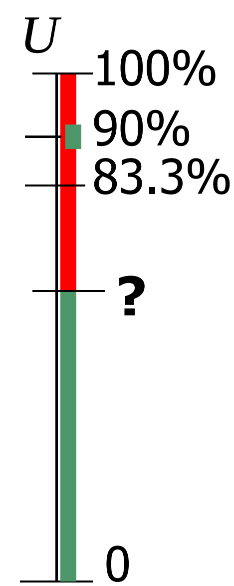


$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)





$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

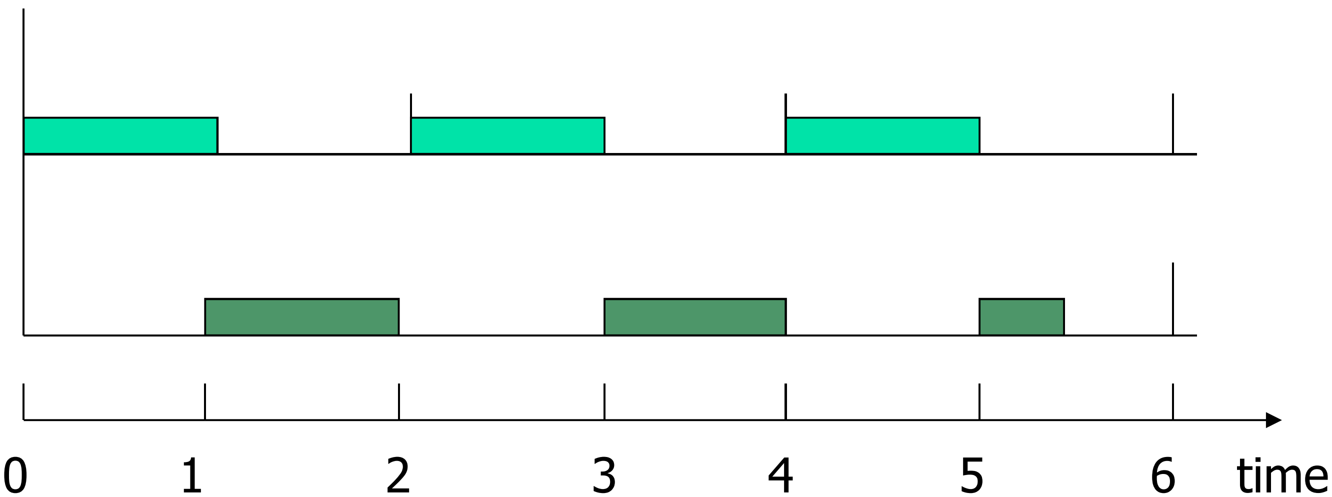
Schedulable!

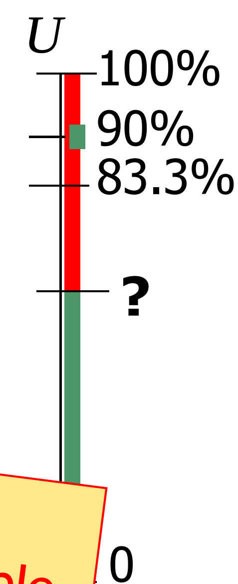
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4



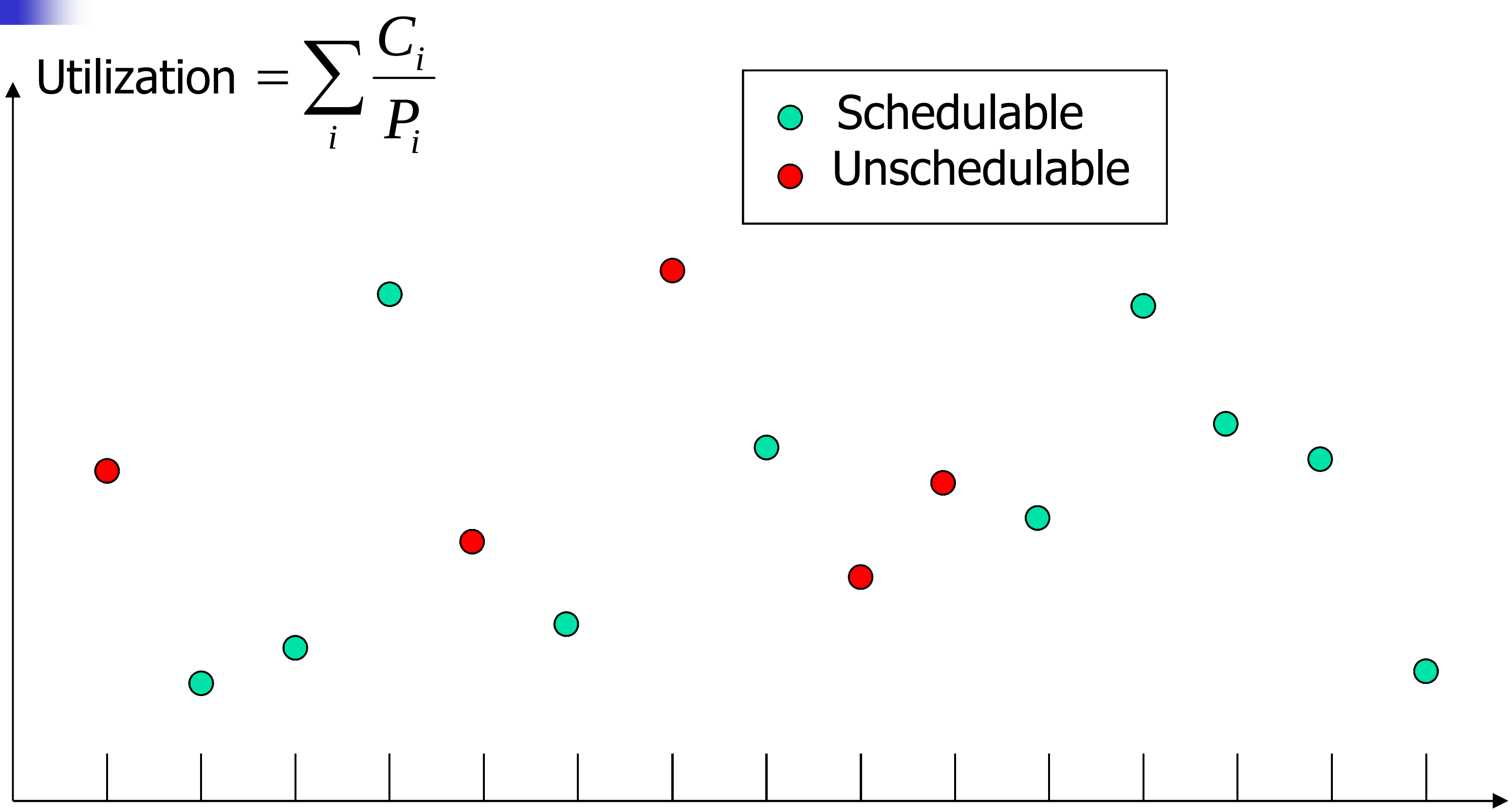


$$\frac{C_1}{P_1}+\frac{C_2}{P_2}=\frac{1}{2}+\frac{2.4}{6}=90\%$$

Schedulability depends on task set!
No clean utilization threshold between schedulable and unschedulable task sets!

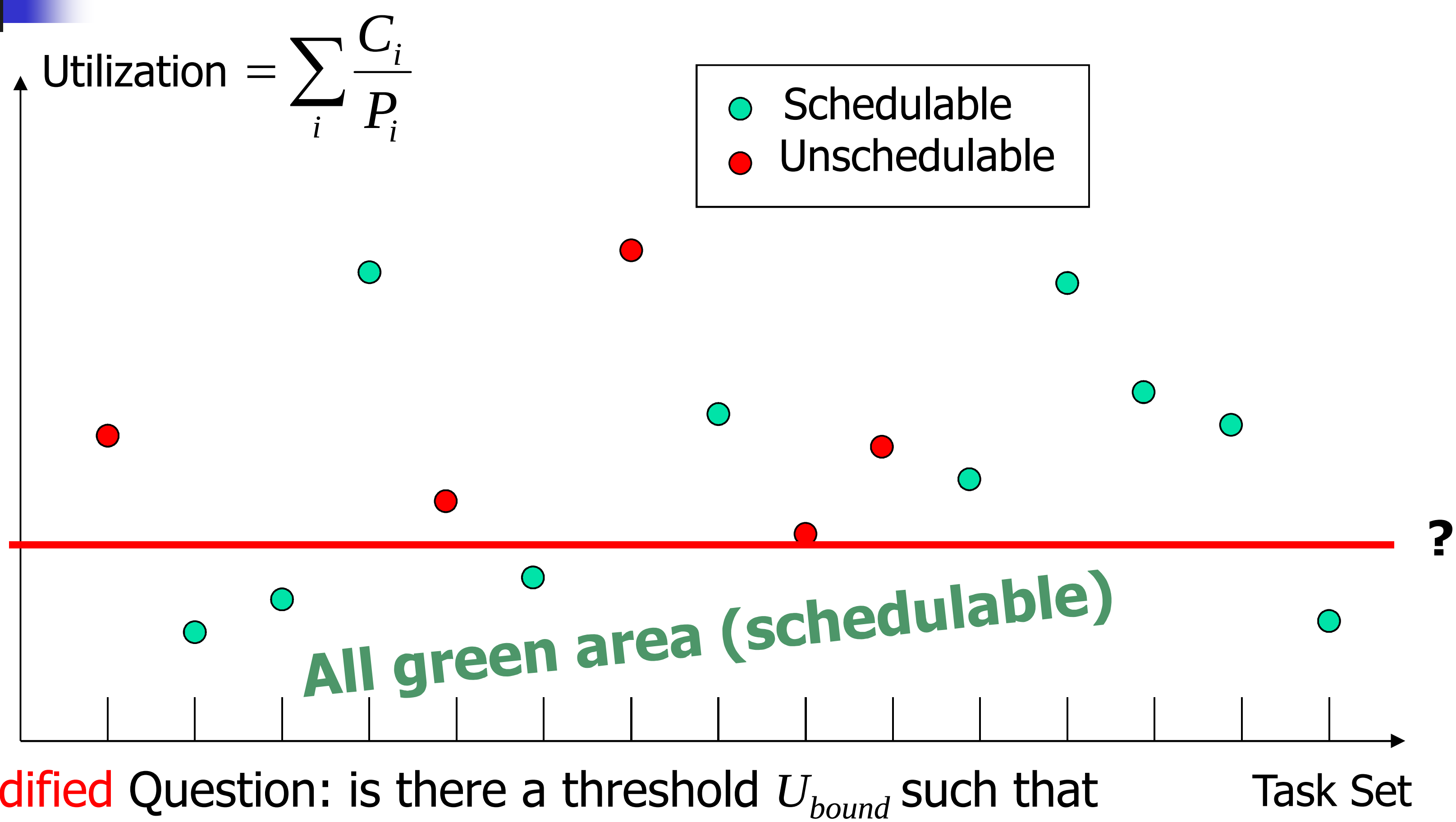
- Question: is there a th
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

A Conceptual View of Schedulability



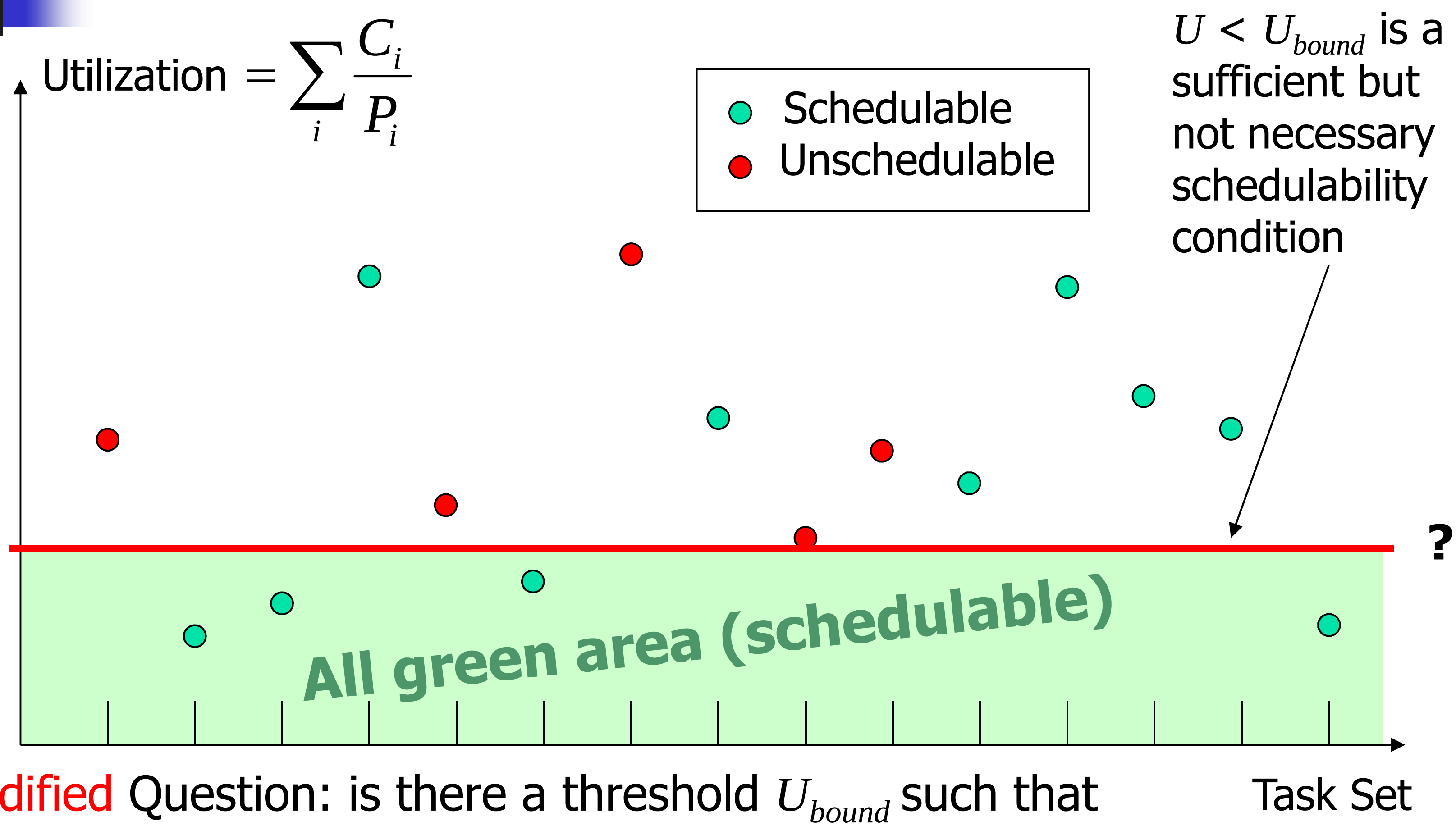
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

A Conceptual View of Schedulability



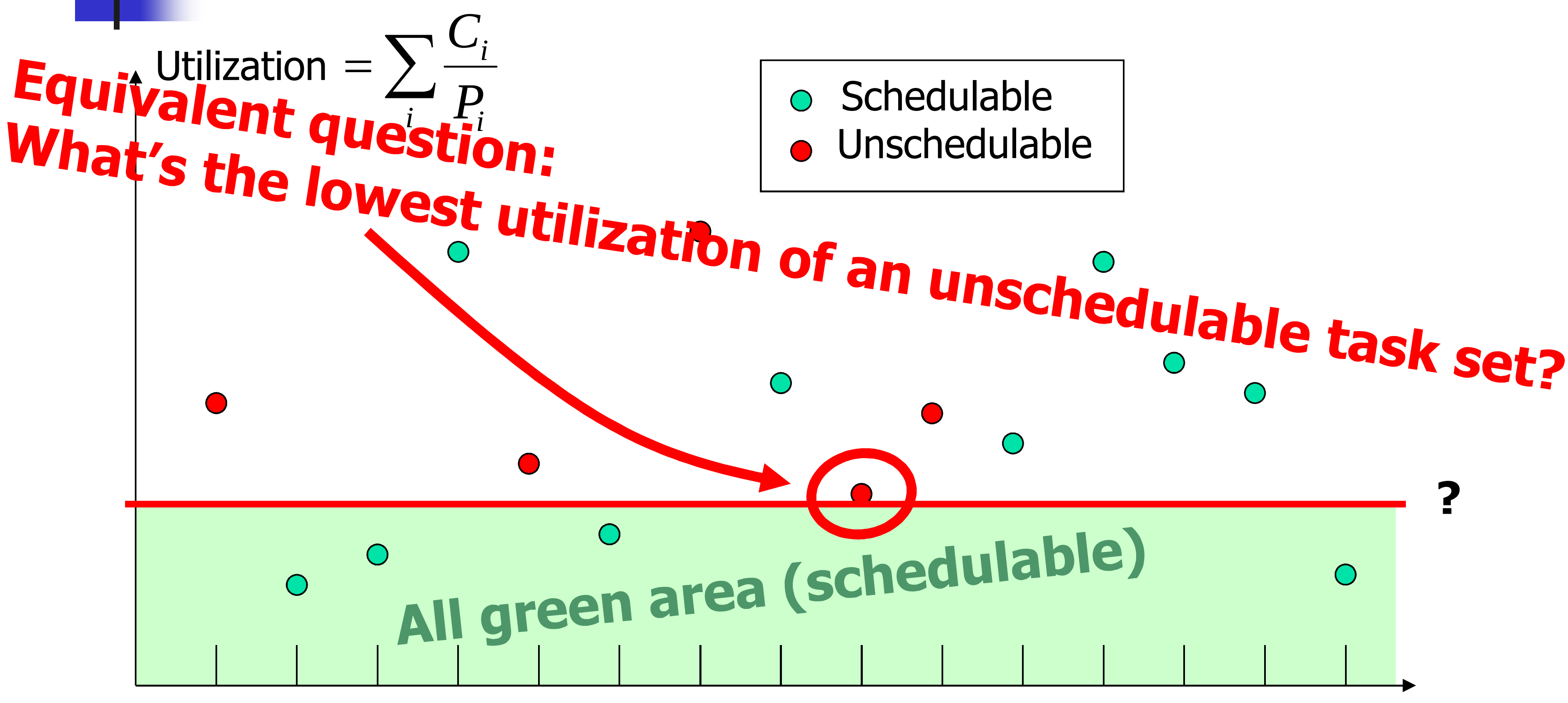
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



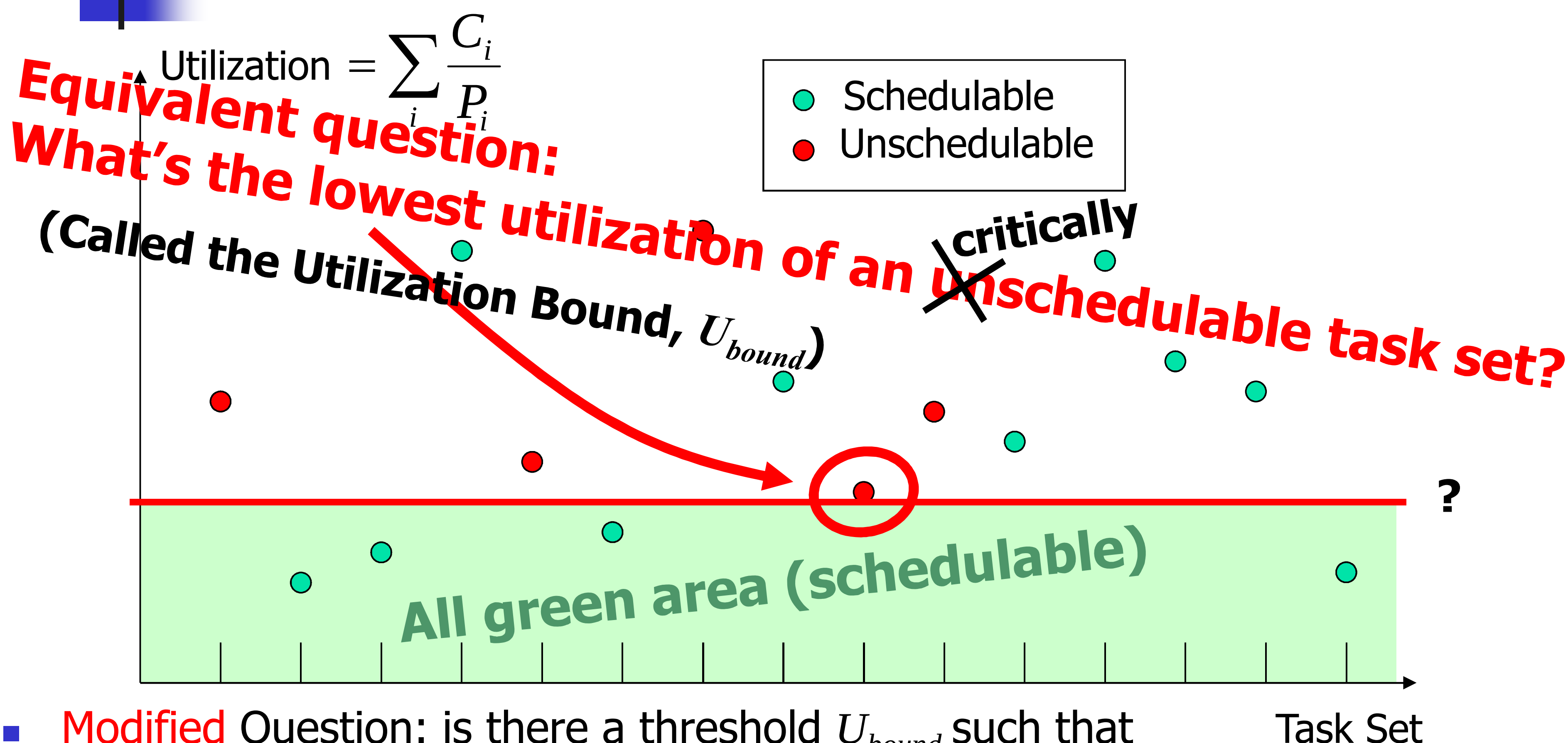
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

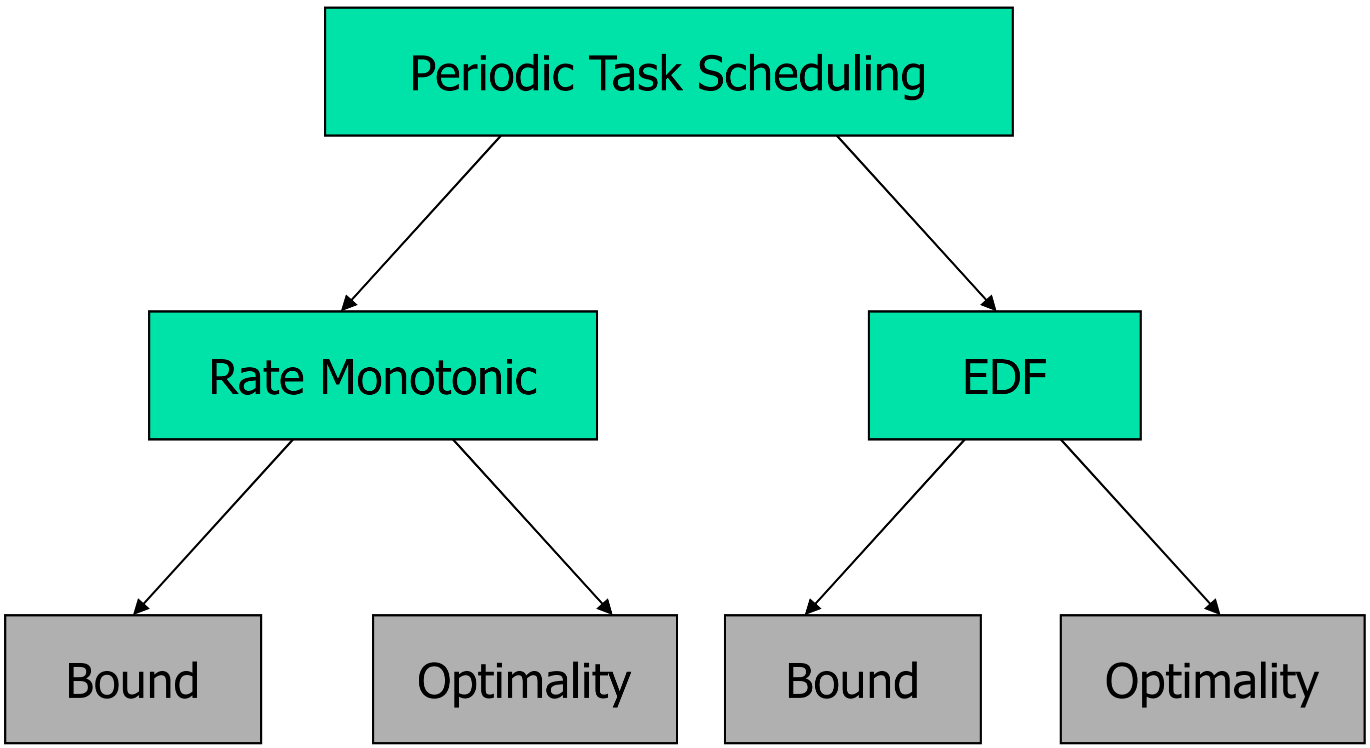
The Schedulability Condition

For n independent periodic tasks with periods equal to deadlines, the utilization bound is:

$$U = n\left(2^{1/n} - 1\right)$$

$$n \to \infty \quad U \to \ln 2$$

Done Today


Periodic Task Scheduling
Rate Monotonic
EDF
Bound
Optimality
Bound
Optimality