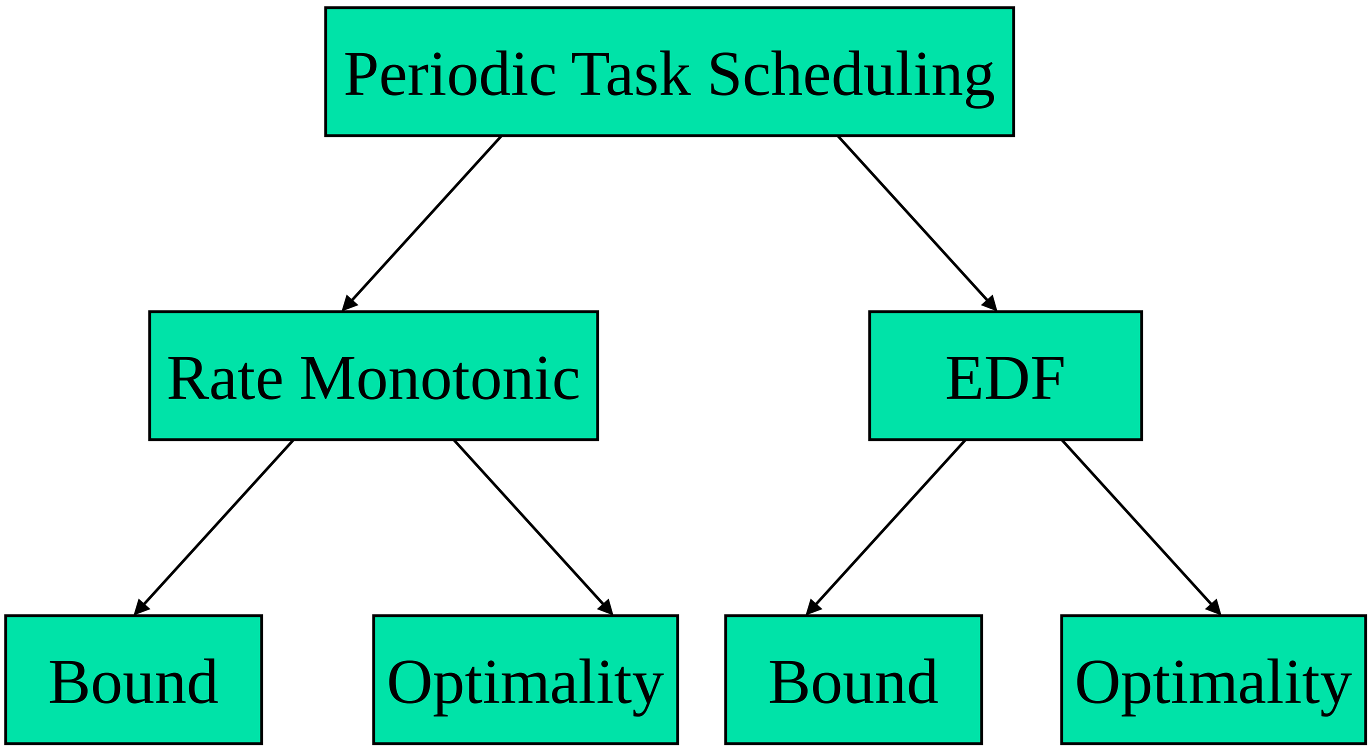
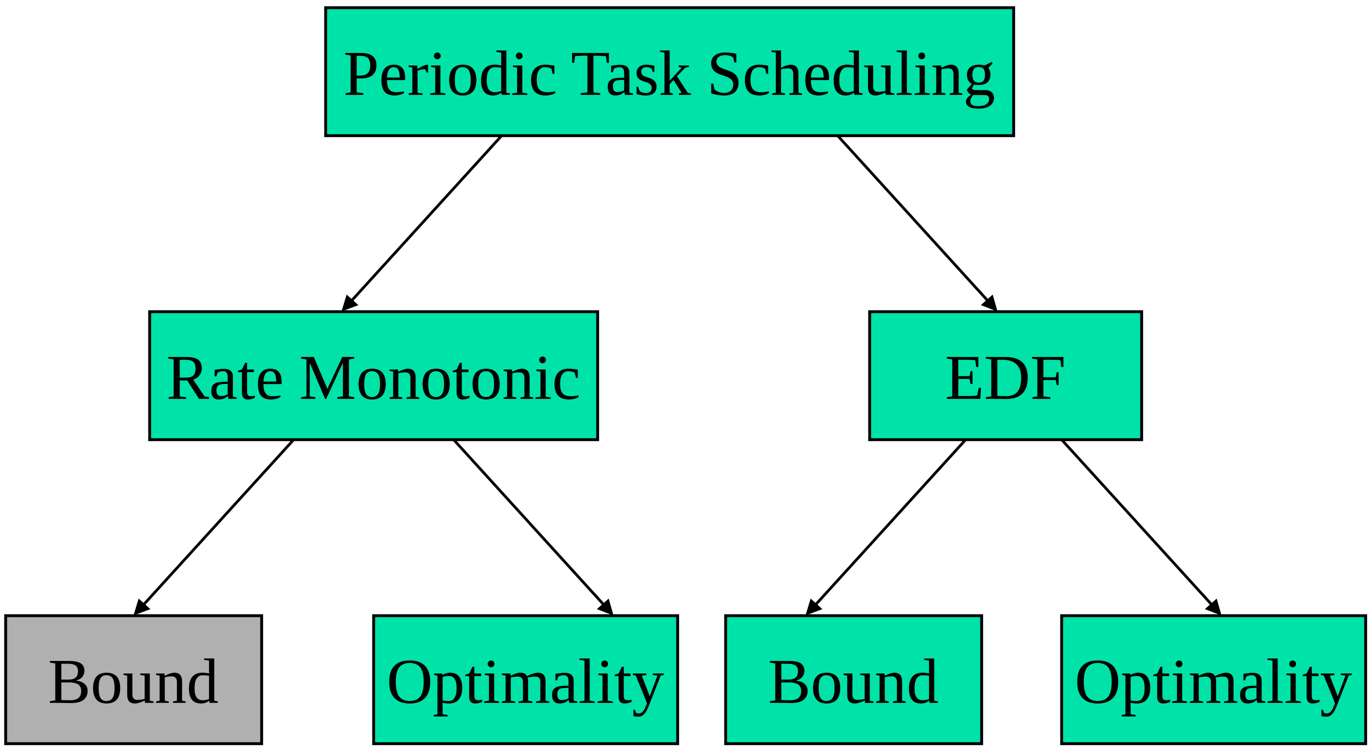
Real-Time Systems Optimality Results

Tarek Abdelzaher

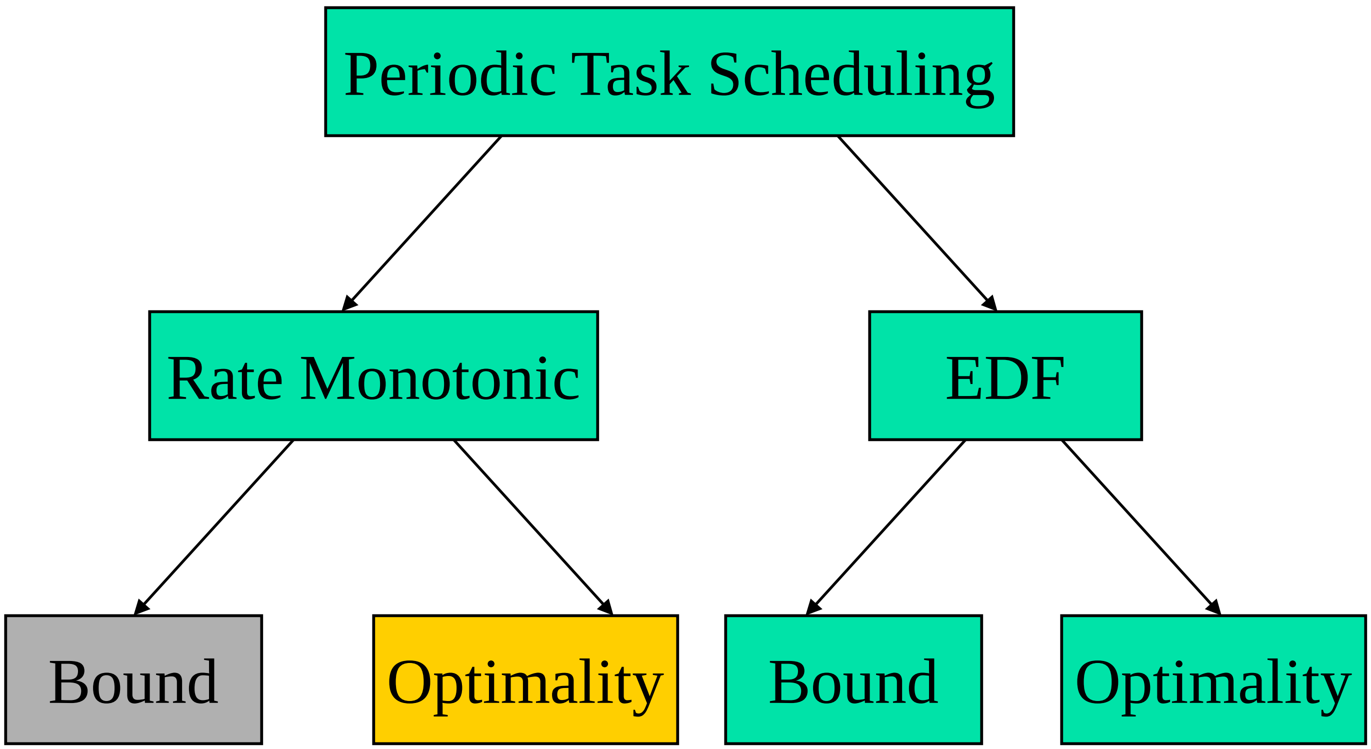
Coming Up Today



Coming Up Today



Coming Up Today



Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality: If any other fixed-priority scheduling policy can meet *all* deadlines, so can RM.
- How to prove it?

Rate Monotonic Continued

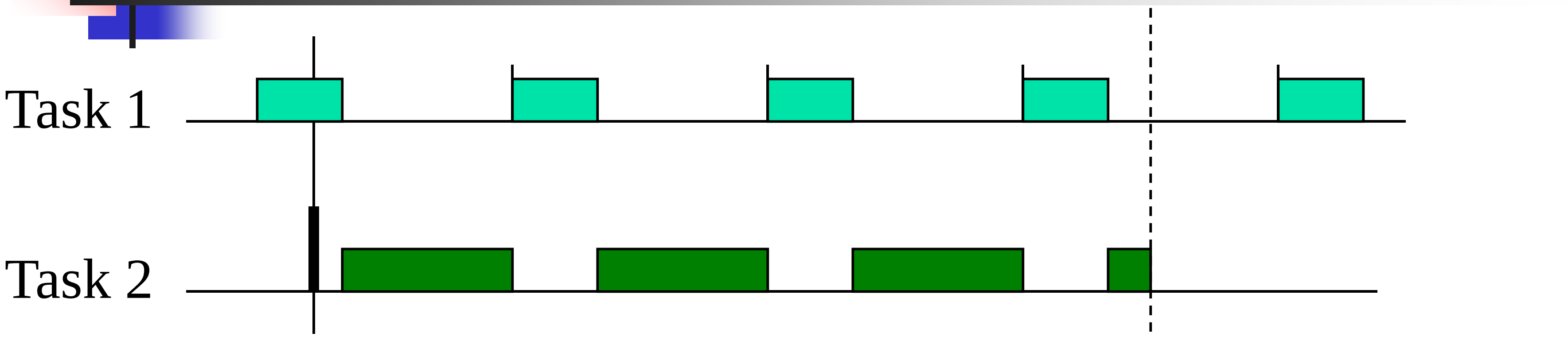
- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality: If any other fixed-priority scheduling policy can meet *all* deadlines, so can RM.
- How to prove it?
  - Consider the worst case task arrival times scenario
  - Show that if someone else can schedule it then RM can

The Worst-Case Scenario

- Q: When does a periodic task, *T*, experience the maximum delay?
- A: When it arrives together with all the higher-priority tasks (critical instance)

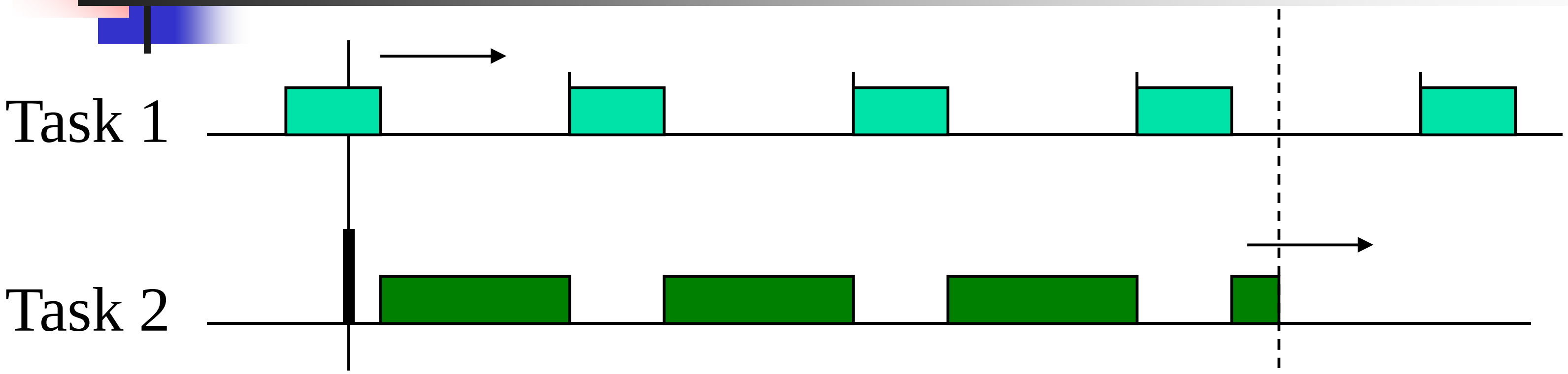
- Idea of Proof
  - If some higher-priority task does not arrive together with *T*, aligning the arrival times can only increase the completion time of *T*

Proof (Case 1)



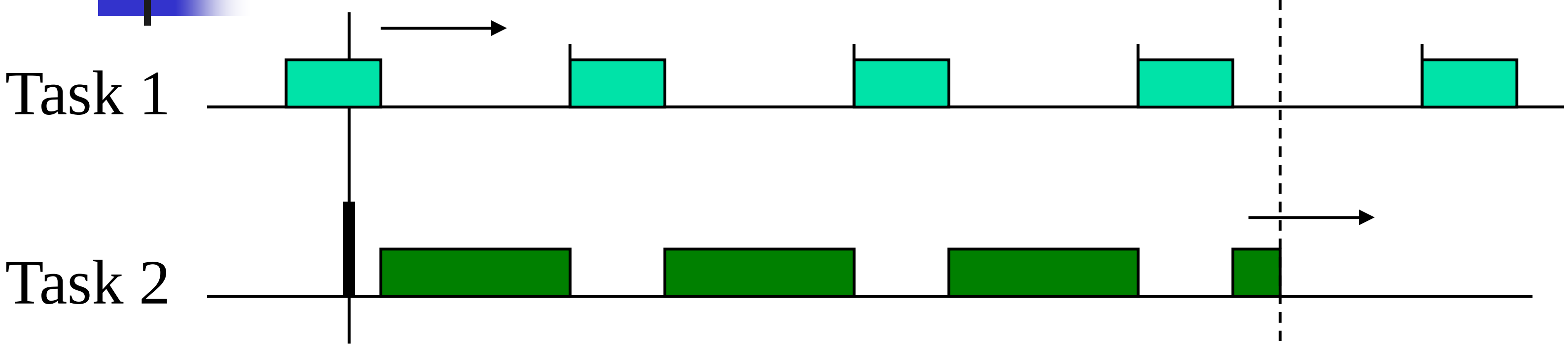
Case 1: higher priority task 1 is running when task 2 arrives

Proof

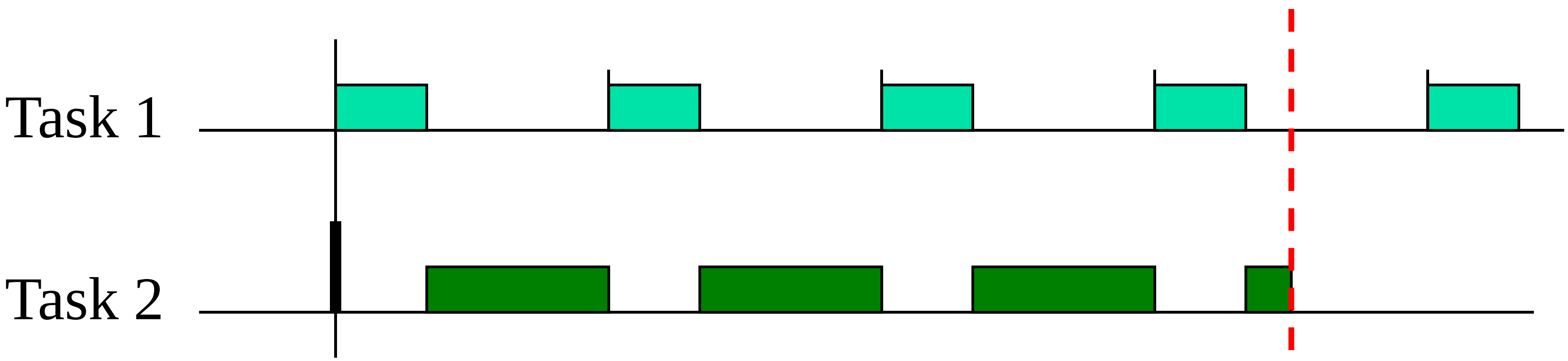


Case 1: higher priority task 1 is running when task 2 arrives
→ shifting task 1 right will increase completion time of 2

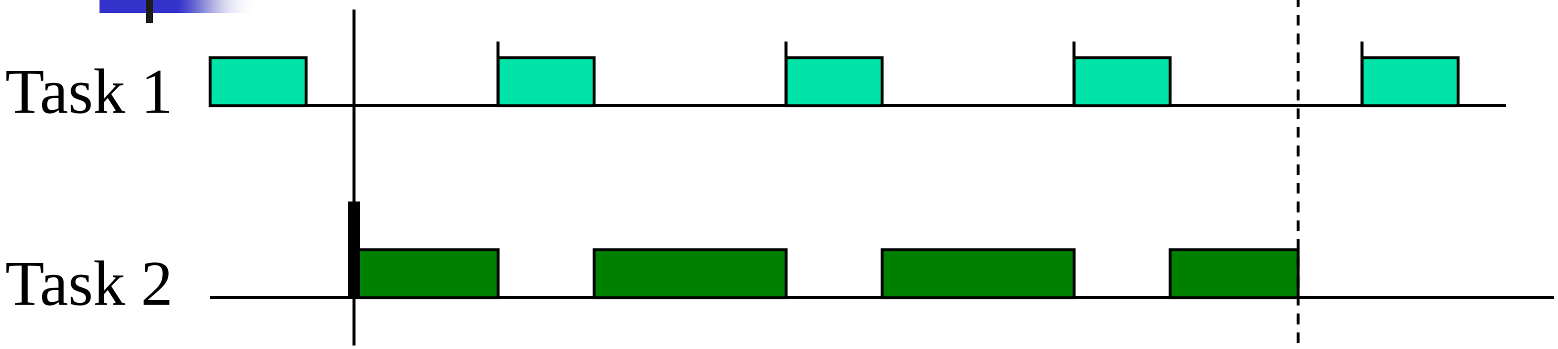
Proof



Case 1: higher priority task 1 is running when task 2 arrives
→ shifting task 1 right will increase completion time of 2

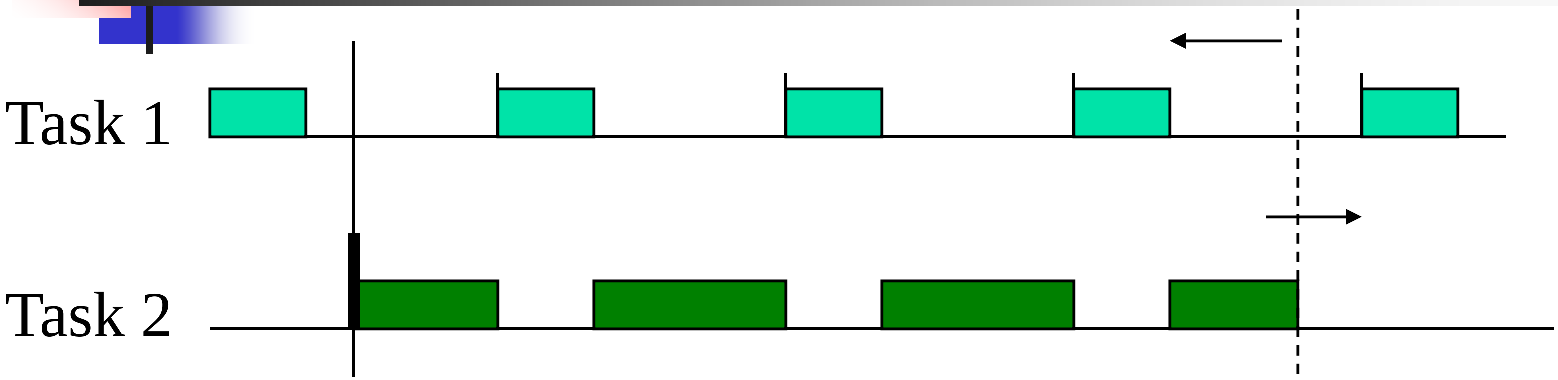


Proof (Case 2)



Case 2: processor is idle when task 2 arrives

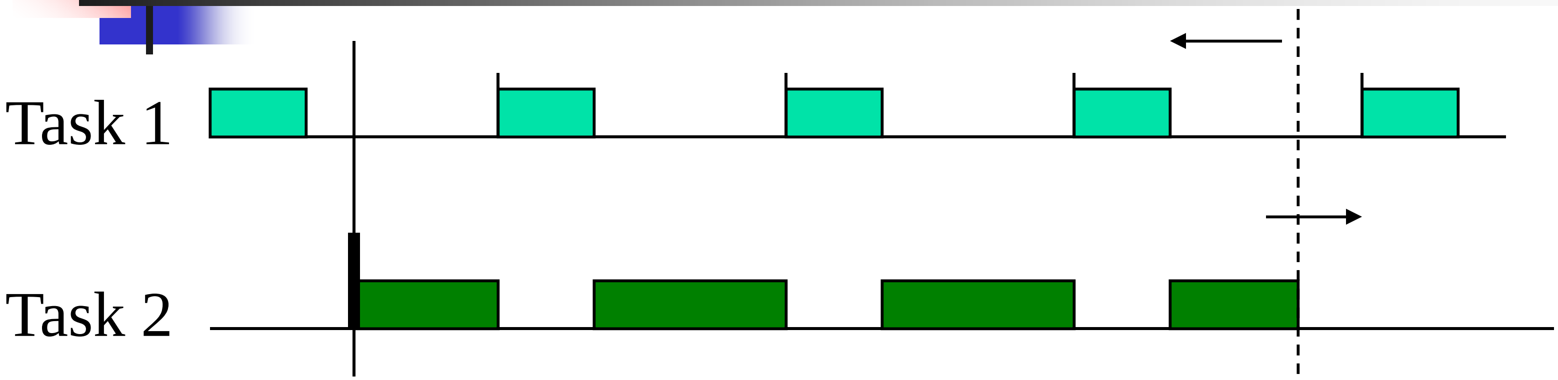
Proof (Case 2)



Case 2: processor is idle when task 2 arrives

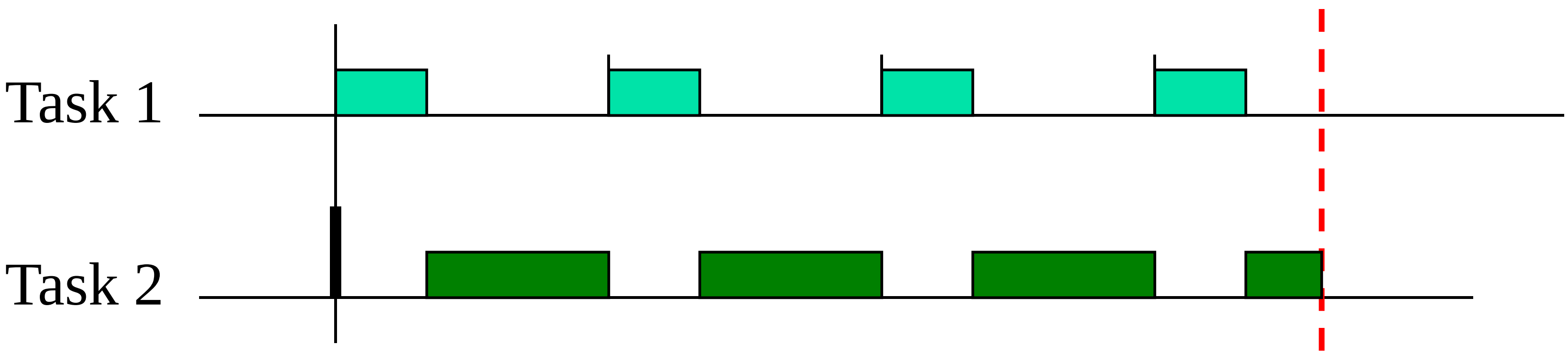
→ shifting task 1 left cannot decrease completion time of 2

Proof (Case 2)



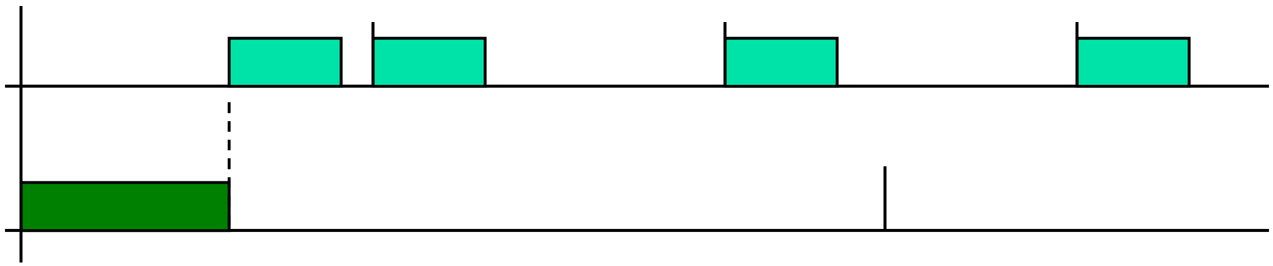
Case 2: processor is idle when task 2 arrives

→ shifting task 1 left cannot decrease completion time of 2



Optimality of Rate Monotonic

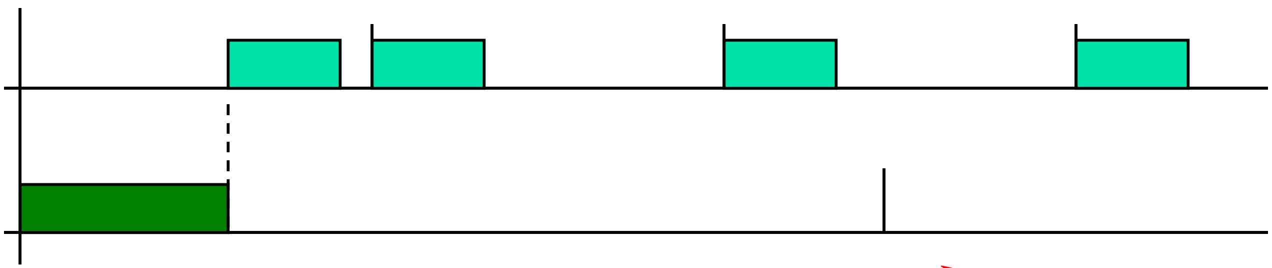
- If any other policy can meet deadlines so can RM



Policy X meets deadlines?

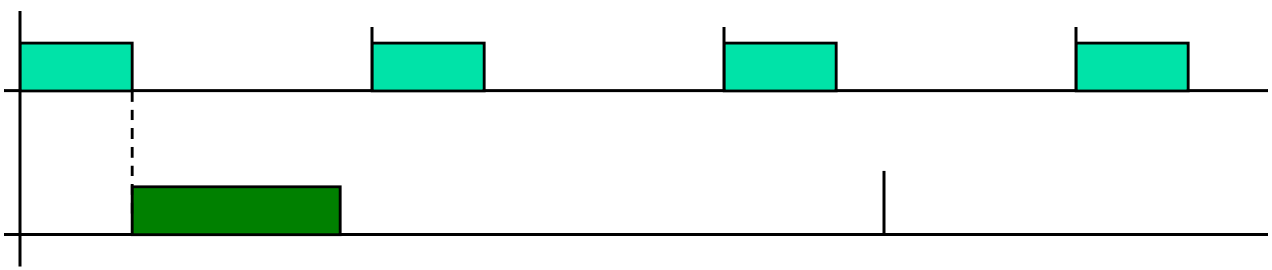
Optimality of Rate Monotonic

- If any other policy can meet deadlines so can RM

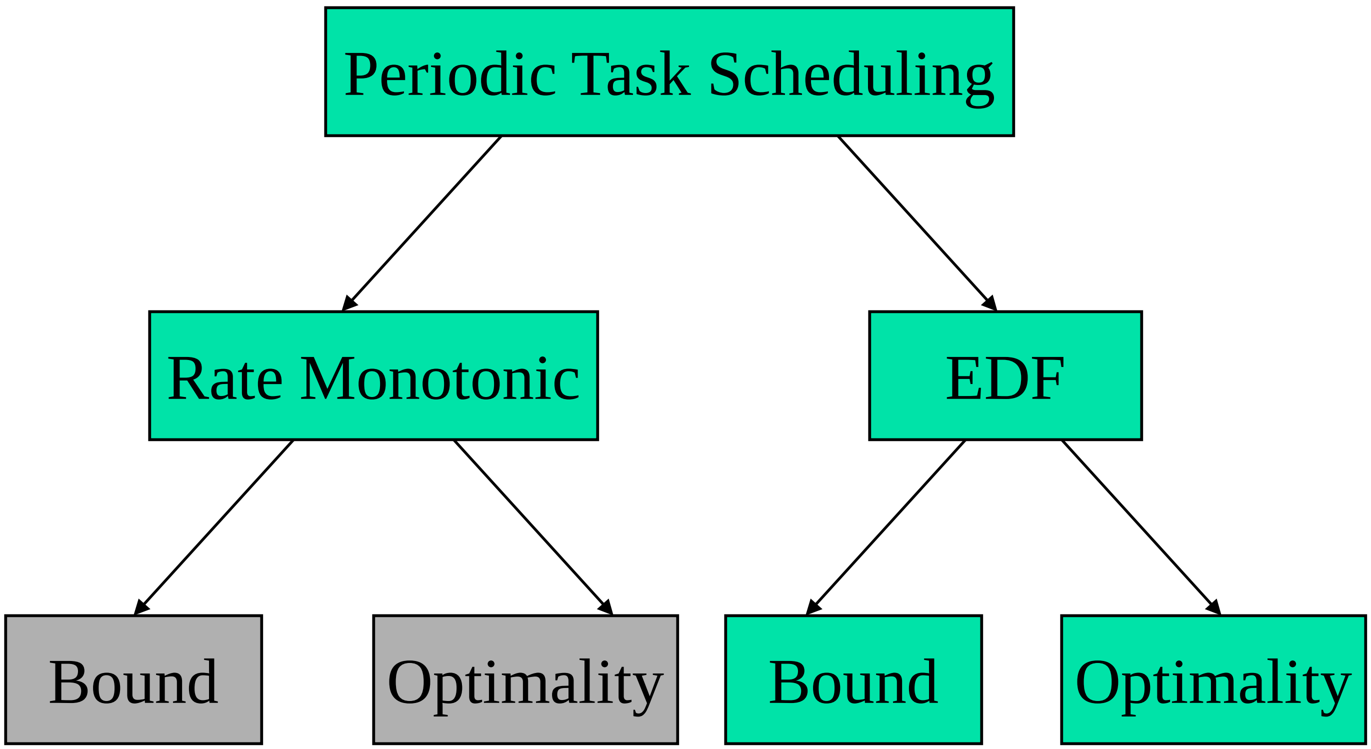


Policy X meets deadlines? *YES*

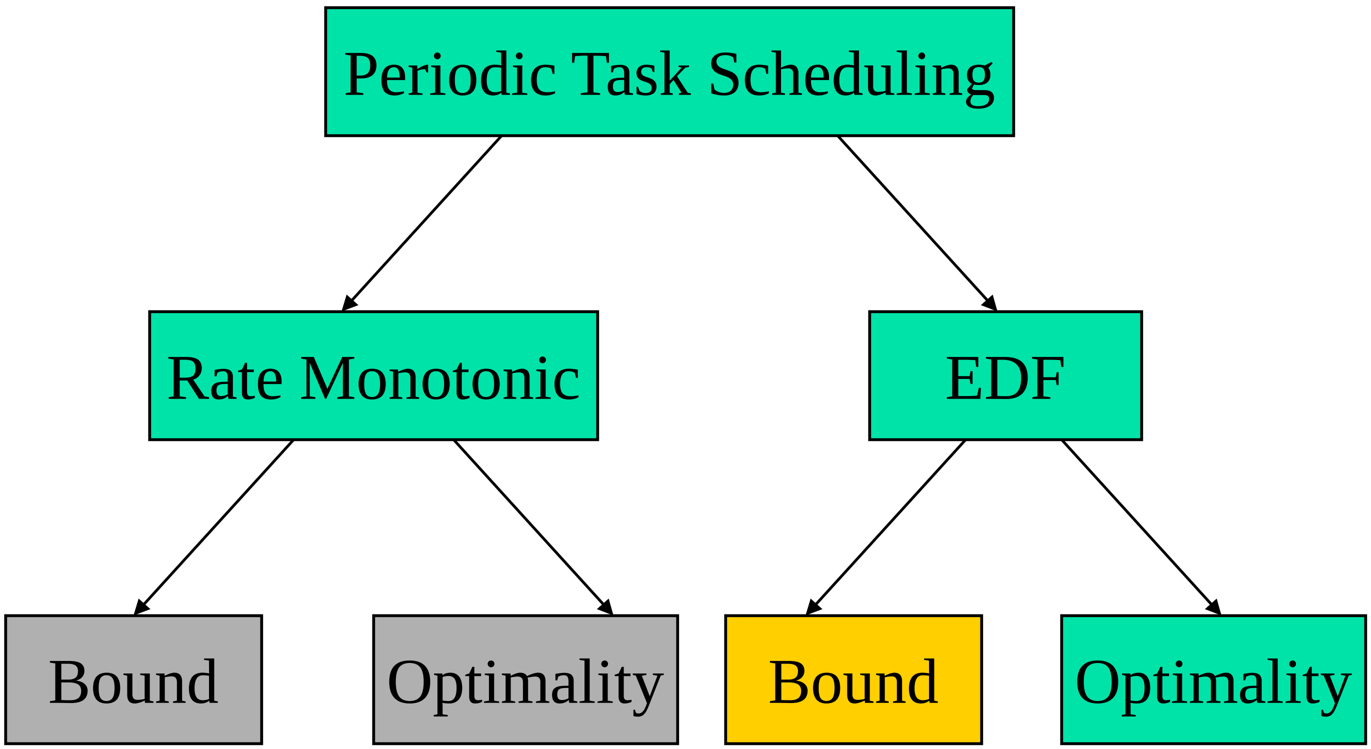
→ RM meets deadlines



Coming Up Today



Coming Up Today

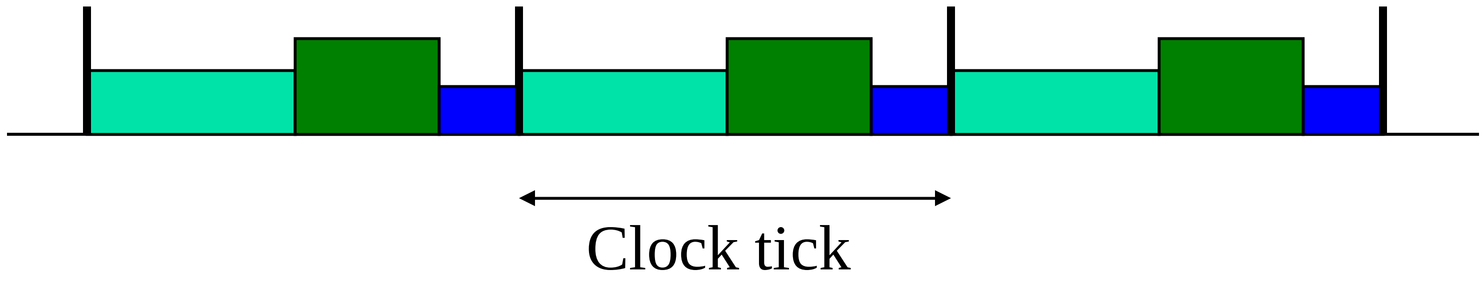


Utilization Bound of EDF

- Why is it 100%?
- Consider a task set where:

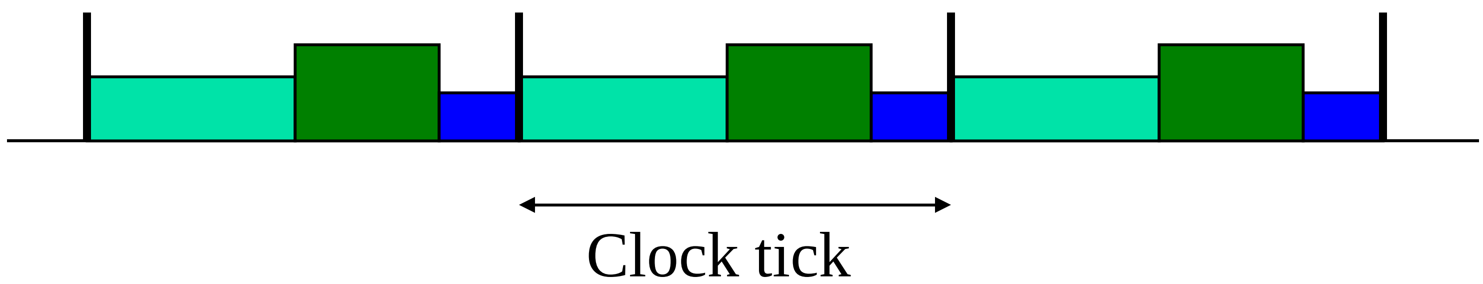
$$\sum_i \frac{C_i}{P_i} = 1$$

- 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

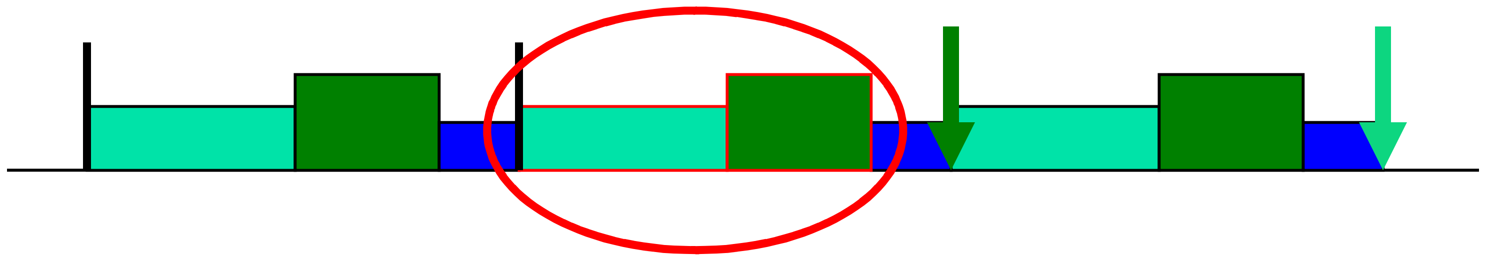
- Imagine a policy that reserves for each task *i* a fraction f_i of each time unit, where $f_i = C_i / P_i$



- 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)

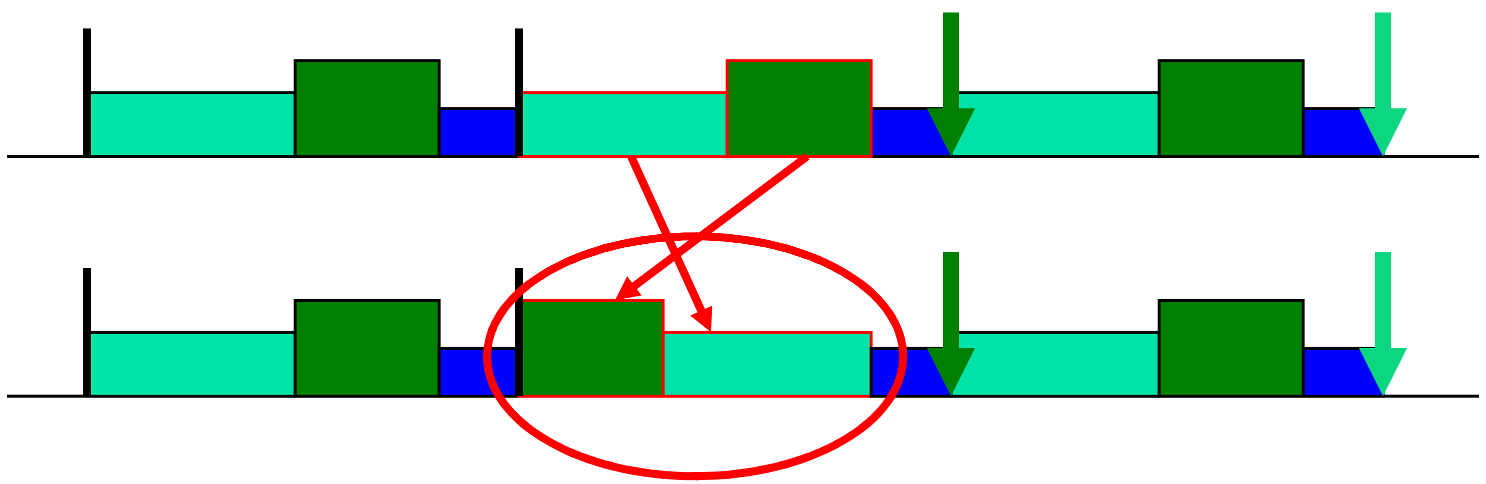
Utilization Bound of EDF

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



Utilization Bound of EDF

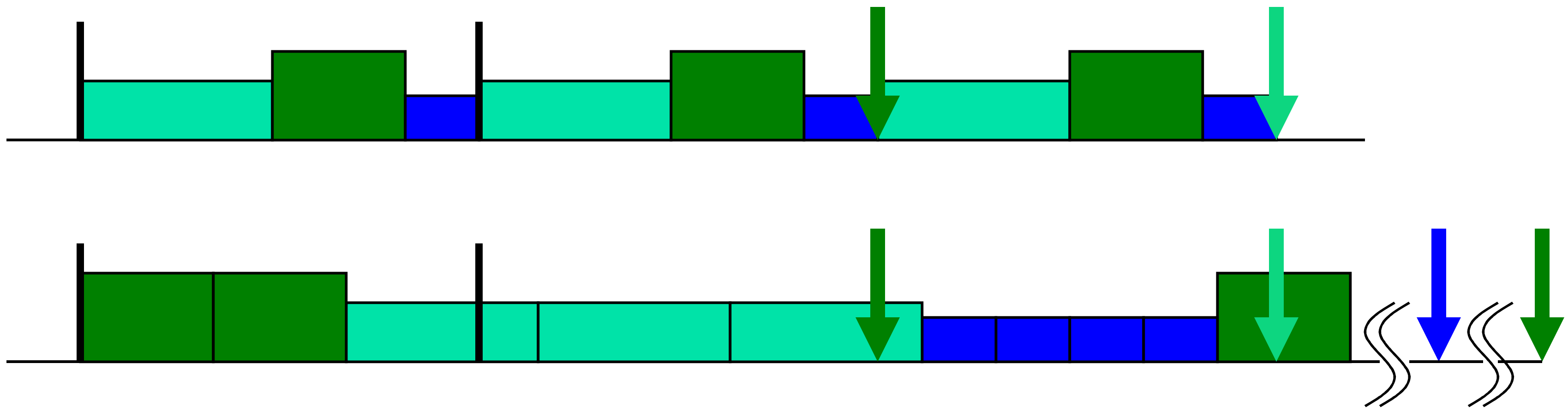
- Pick any two execution chunks that are not in EDF order and swap them



- Still meets deadlines!

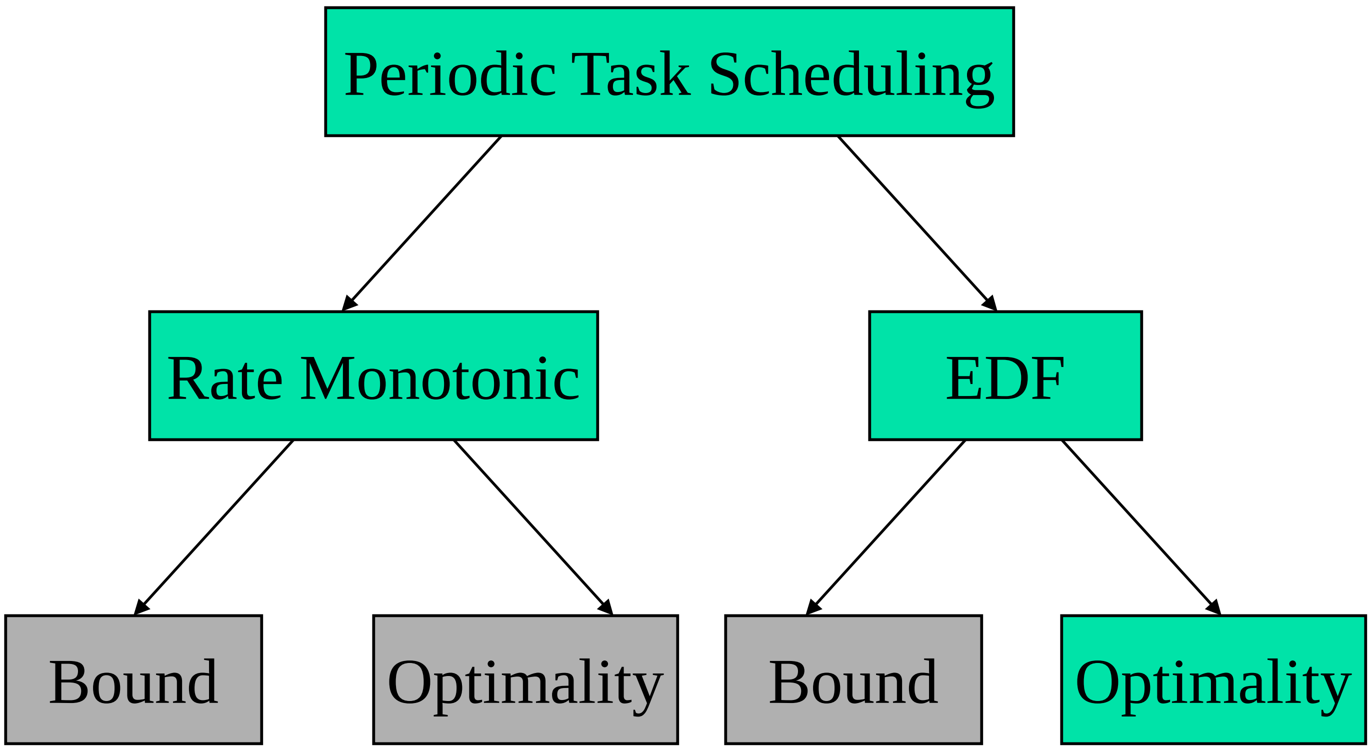
Utilization Bound of EDF

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

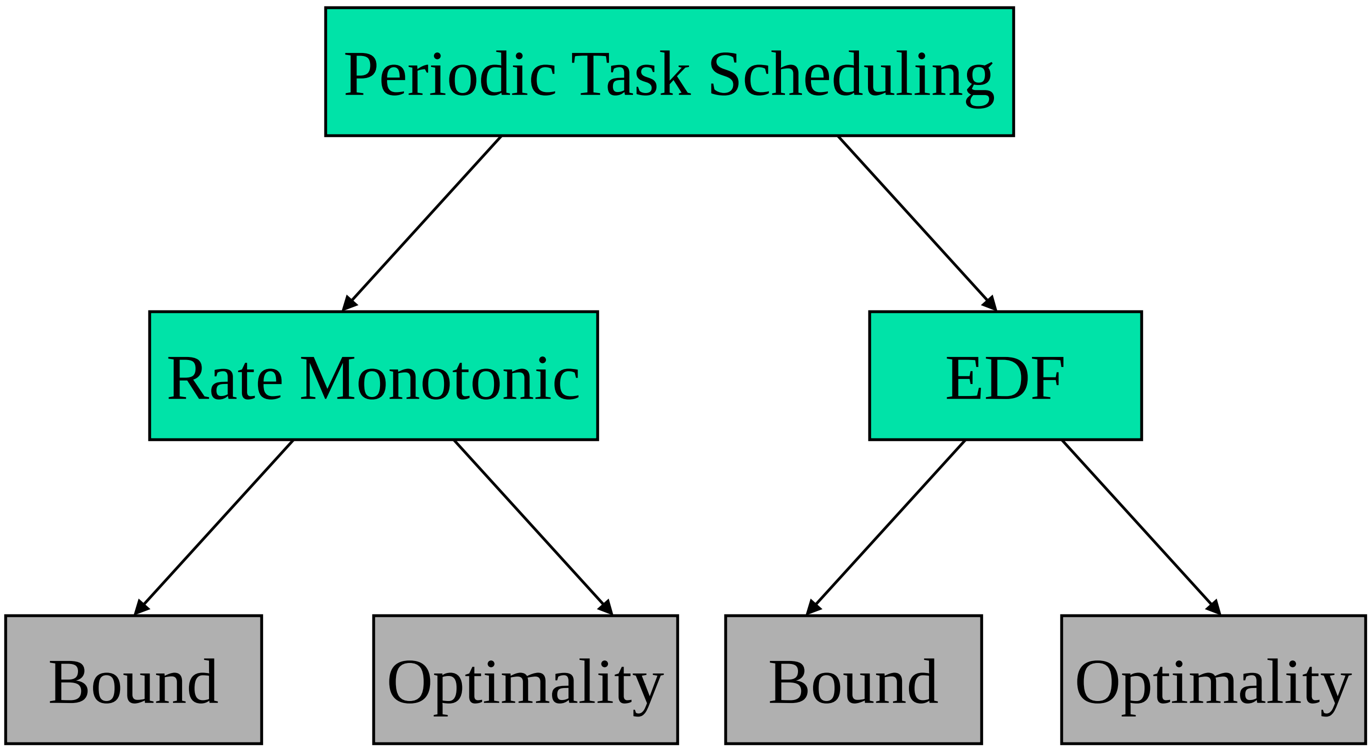


- Still meets deadlines!
- Repeat swap until all in EDF order
  → EDF meets deadlines

Coming Up Today



Done Today


Periodic Task Scheduling
Rate Monotonic
EDF
Bound
Optimality
Bound
Optimality

The Hyperbolic Bound for Rate Monotonic Scheduling (2001)

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \leq 2$$

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \le 2$$

- It's a better bound than $\sum_i U_i \le n\left(2^{1/n} - 1\right)$
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \leq 2$$

- It's a better bound!
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1
    - Liu and Layland bound: $U_1+U_2 = 0.9 > 0.83$ ✗

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

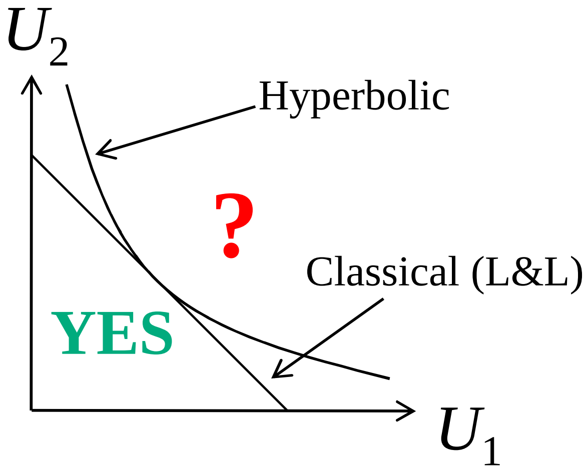
$$\prod_i (U_i + 1) \leq 2$$

- It's a better bound!
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1
    - Liu and Layland bound: $U_1+U_2 = 0.9 > 0.83$ ✗
    - Hyperbolic bound $(U_1+1)(U_2+1) = 1.8 \times 1.1 = 1.98 < 2$ ✓

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \le 2$$



- It's a better bound!
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1
    - Liu and Layland bound: U_1+U_2 = 0.9 > 0.83
    - Hyperbolic bound $(U_1+1)(U_2+1)$ =1.8 x 1.1=1.98 < 2

Exercise: Know Your Worst Case Scenario

- Consider a periodic system of two tasks
- Let $U_i = C_i / P_i$ (for $i = 1,2$)
- What is the maximum value of:

$$\Pi_i(1+U_i)$$

for a schedulable system?

Deriving the Utilization Bound for Rate Monotonic Scheduling

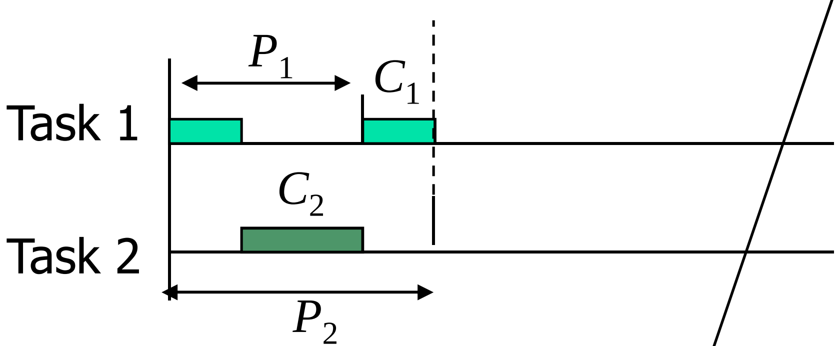
- The minimum utilization case:

$$C_1 = P_1\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor\right)$$

$$C_1 = P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$$

$$U = 1 + \frac{P_1}{P_2}\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor\right)\left[\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1\right]$$

$$\Rightarrow \left\lfloor \frac{P_2}{P_1} \right\rfloor = 1$$



$$C_1 + C_2 = P_1$$

$$P_2 - P_1 = C_1$$

$$U = 1 + \frac{C_1}{P_2}\left[\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1\right]$$

Solutions

$$C_1 = P_2 - P_1$$

$$C_2 = P_1 - C_1 = 2P_1 - P_2$$

Critically Schedulable

Schedulable

Solutions

$$\left\{\begin{array}{l} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \\ U_1 + 1 = \dfrac{C_1}{P_1} + 1 = \dfrac{C_1 + P_1}{P_1} = \dfrac{P_2}{P_1} \\ \\ \\ \end{array}\right.$$

Critically Schedulable

Schedulable

Solutions

Critically Schedulable

$$
\begin{cases}
C_1 = P_2 - P_1 \\
C_2 = P_1 - C_1 = 2P_1 - P_2 \\
U_1 + 1 = \dfrac{C_1}{P_1} + 1 = \dfrac{C_1 + P_1}{P_1} = \dfrac{P_2}{P_1} \\
U_2 + 1 = \dfrac{C_2}{P_2} + 1 = \dfrac{C_2 + P_2}{P_2} = \dfrac{2P_1}{P_2}
\end{cases}
$$

Schedulable

Solutions

Critically Schedulable

$$\begin{cases} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \\ U_1 + 1 = \dfrac{C_1}{P_1} + 1 = \dfrac{C_1 + P_1}{P_1} = \dfrac{P_2}{P_1} \\ U_2 + 1 = \dfrac{C_2}{P_2} + 1 = \dfrac{C_2 + P_2}{P_2} = \dfrac{2P_1}{P_2} \\ \prod_i (U_i + 1) = 2 \end{cases}$$

Schedulable

$$\prod_i (U_i + 1) \leq 2$$

The General Case

Critically Schedulable

$$\left\{ \begin{array}{l} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \end{array} \right.$$

Schedulable

The General Case

$$C_i = P_{i+1} - P_i$$

$$C_n = 2P_1 - P_n$$

$$U_i + 1 = \frac{C_i}{P_i} + 1 = \frac{C_i + P_i}{P_i} = \frac{P_{i+1}}{P_i}$$

Critically Schedulable

Schedulable

The General Case

Critically Schedulable

$$\begin{cases} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \\ U_i + 1 = \dfrac{C_i}{P_i} + 1 = \dfrac{C_i + P_i}{P_i} = \dfrac{P_{i+1}}{P_i} \\ U_n + 1 = \dfrac{C_n}{P_n} + 1 = \dfrac{C_n + P_n}{P_n} = \dfrac{2P_1}{P_n} \end{cases}$$

Schedulable

The General Case

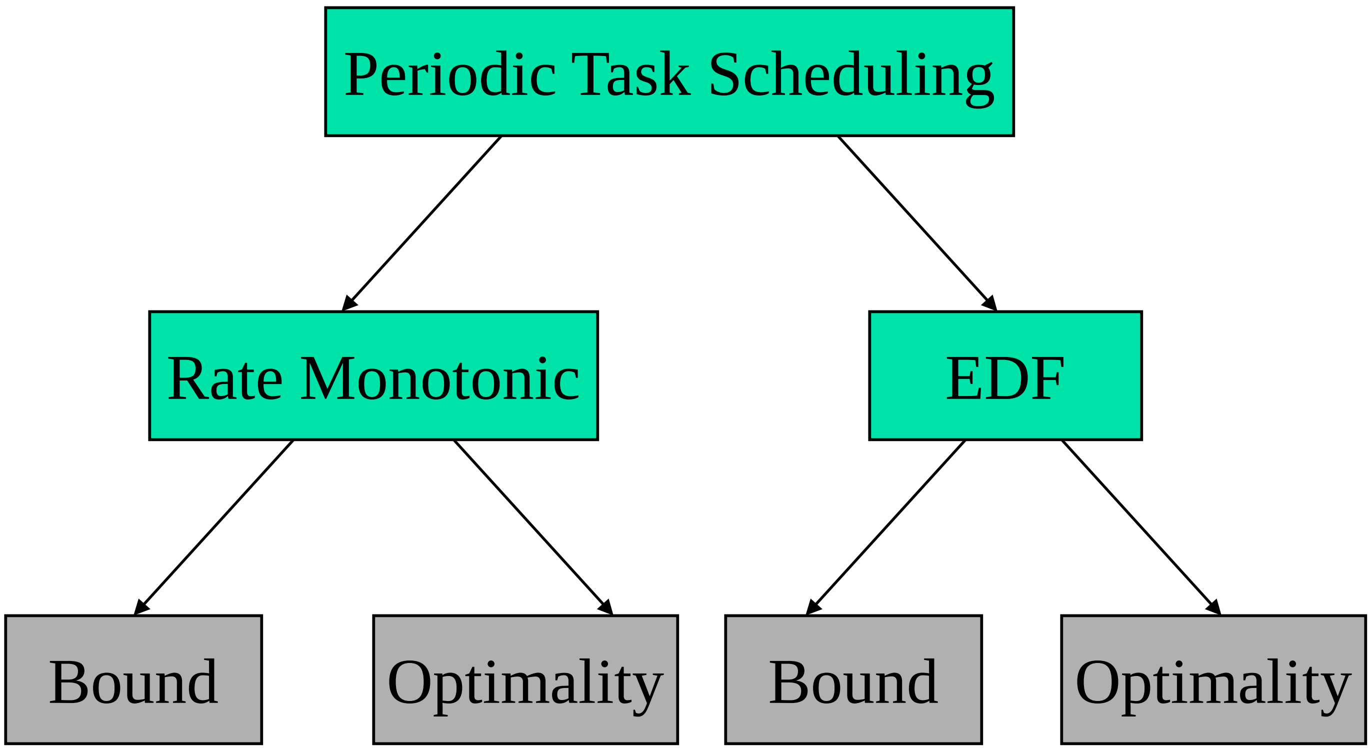
Critically Schedulable

$$\left\{\begin{aligned} &C_i = P_{i+1} - P_i \\ &C_n = 2P_1 - P_n \\ &U_i + 1 = \frac{C_i}{P_i} + 1 = \frac{C_i + P_i}{P_i} = \frac{P_{i+1}}{P_i} \\ &U_n + 1 = \frac{C_n}{P_n} + 1 = \frac{C_n + P_n}{P_n} = \frac{2P_1}{P_n} \\ &\prod_i (U_i + 1) = \frac{P_2}{P_1}\frac{P_3}{P_2}\cdots\frac{P_n}{P_{n-1}}\frac{2P_1}{P_n} = 2 \end{aligned}\right.$$

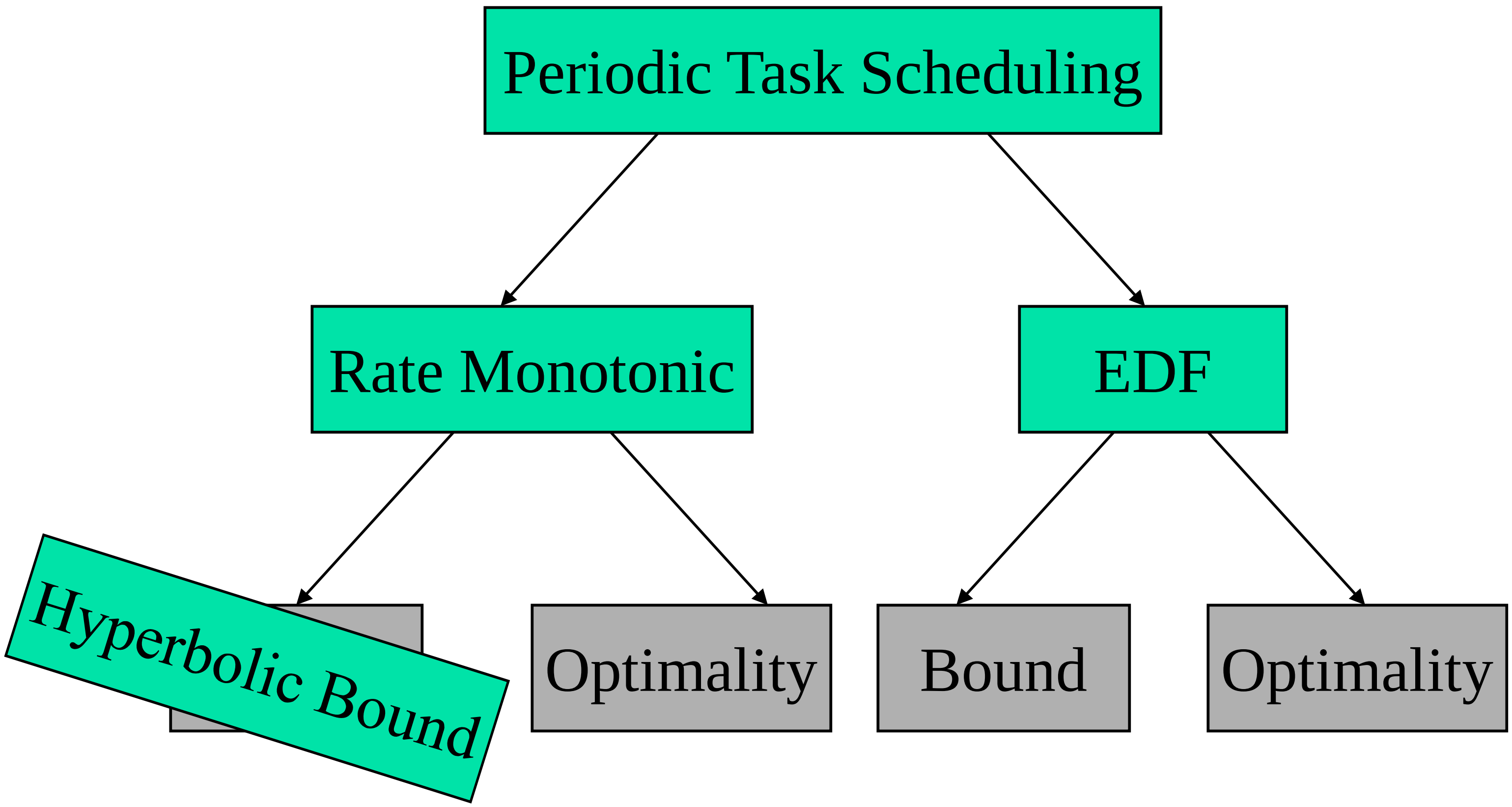
Schedulable

$$\prod_i (U_i + 1) \le 2$$

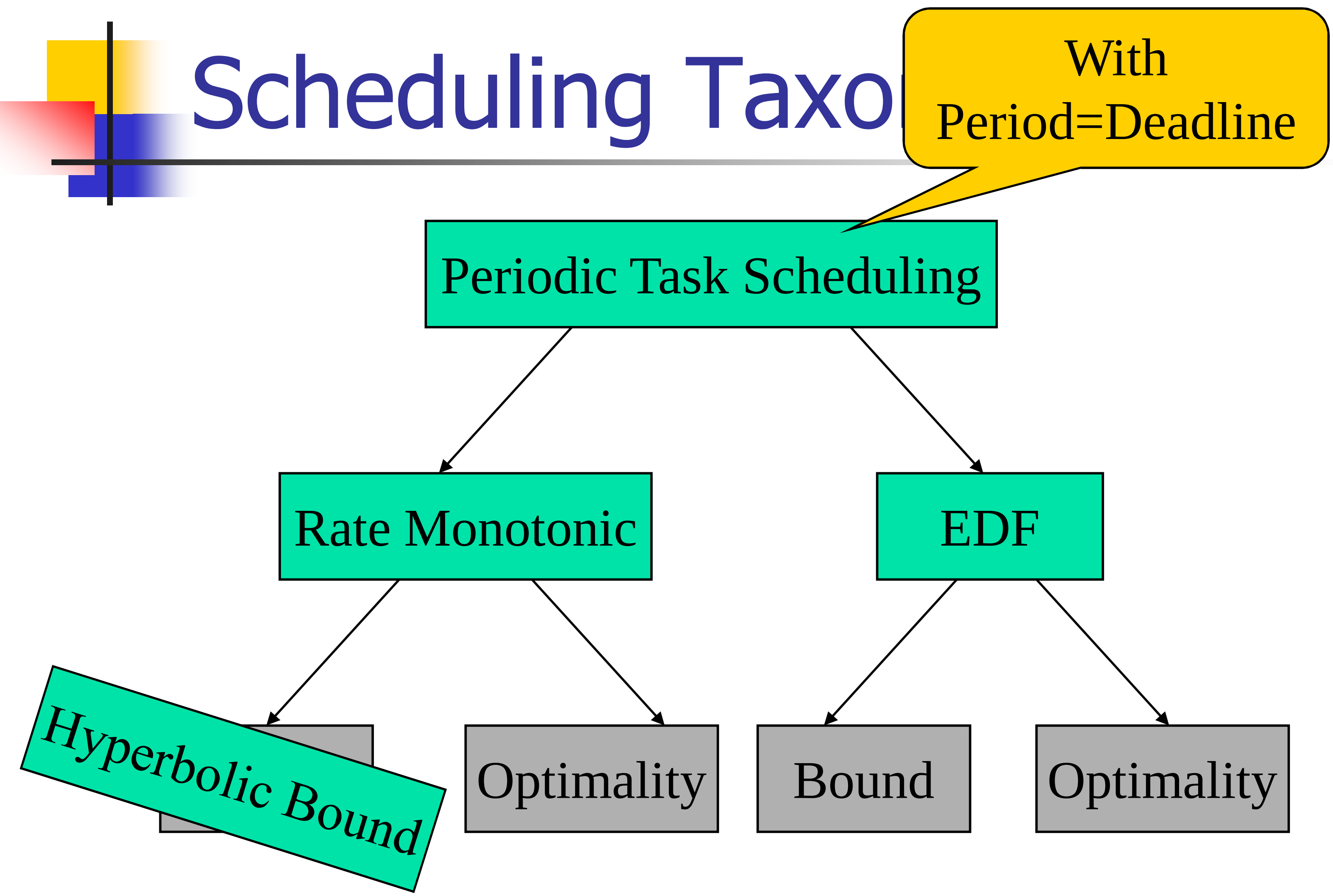
Scheduling Taxonomy



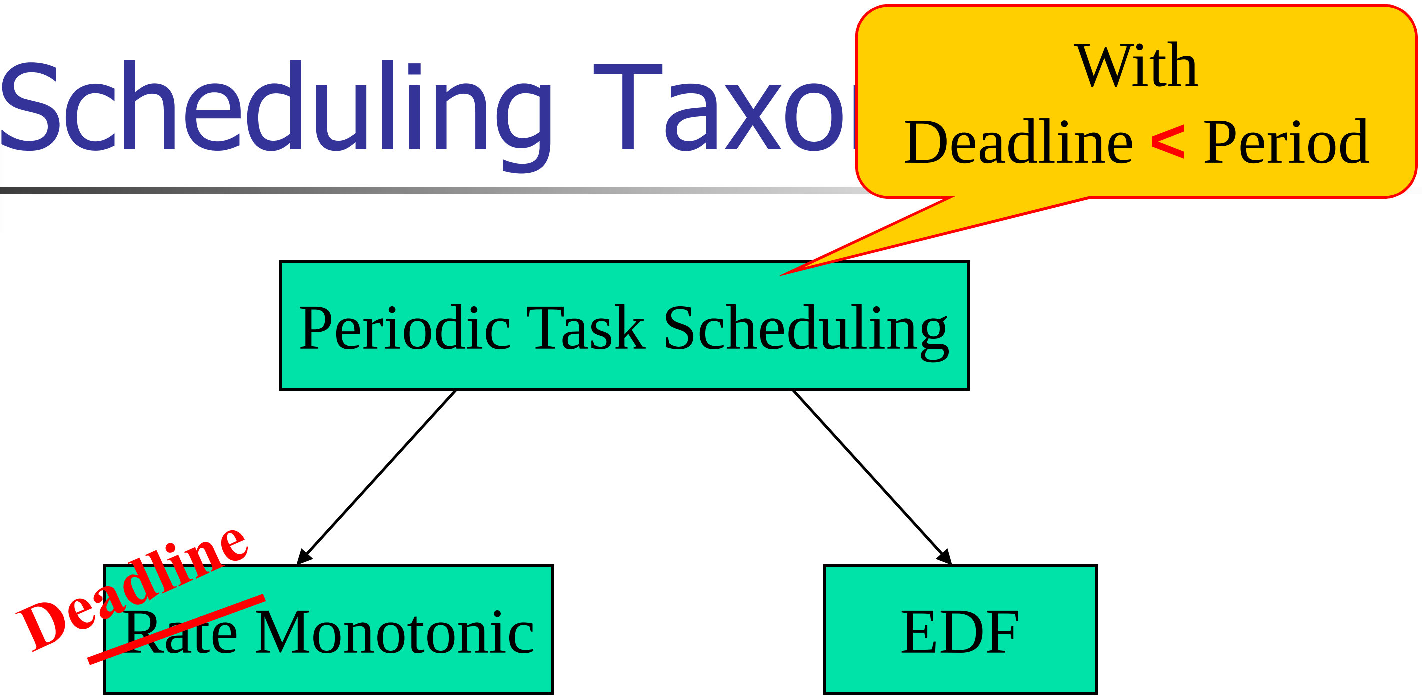
Scheduling Taxonomy



Scheduling Taxo

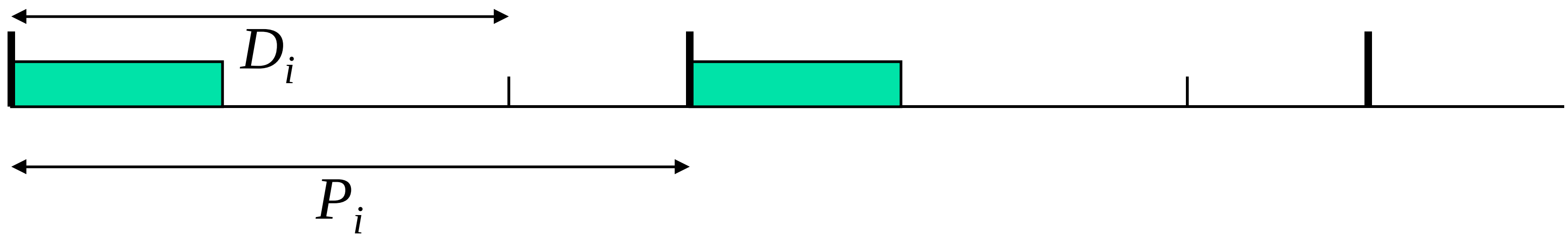


Scheduling Taxo



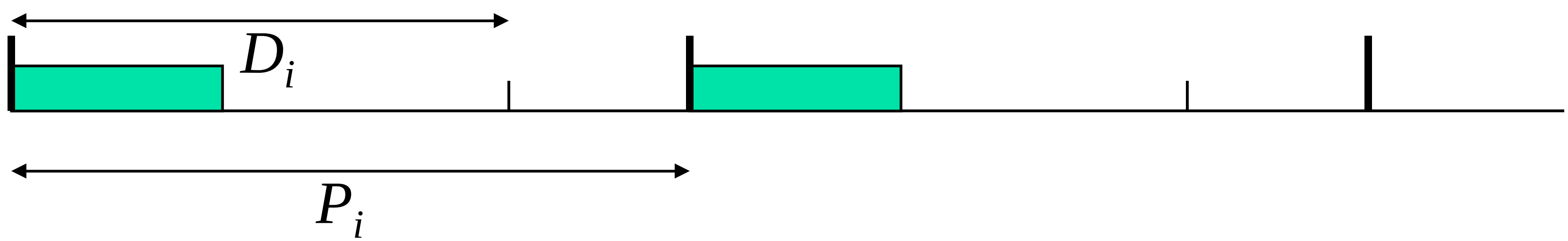
Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, i, has a computation time, C_i, a period, P_i, and a relative deadline $D_i < P_i$.



Deadline Monotonic Scheduling

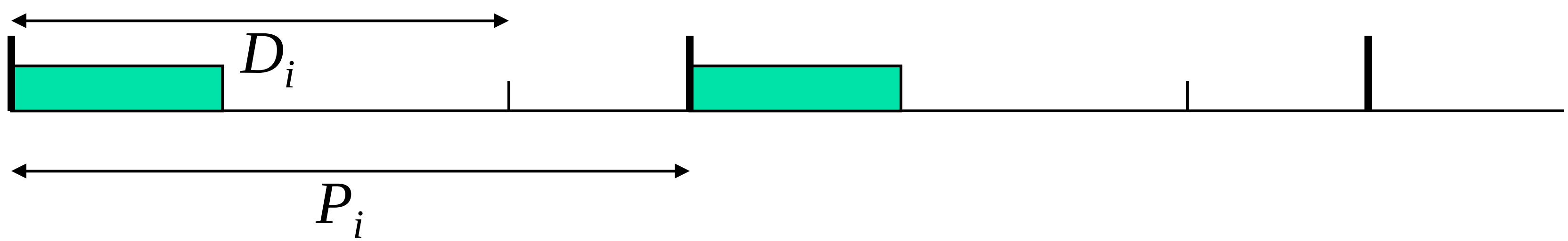
- Consider a set of periodic tasks where each task, i, has a computation time, C_i, a period, P_i, and a relative deadline $D_i < P_i$.



- What is the schedulability condition?

Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, i, has a computation time, C_i, a period, P_i, and a relative deadline $D_i < P_i$.

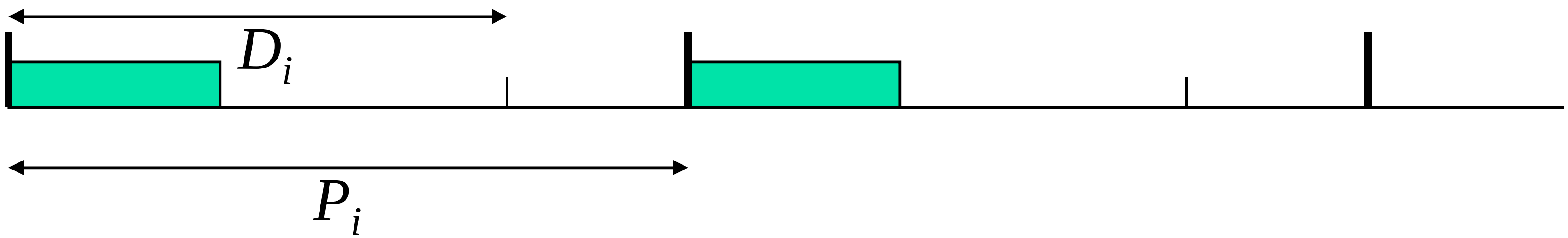


- Schedulability can't be worse than if P_i was reduced to D_i. Thus:

$$\sum_i \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, i, has a computation time, C_i, a period, P_i, and a relative deadline $D_i < P_i$.



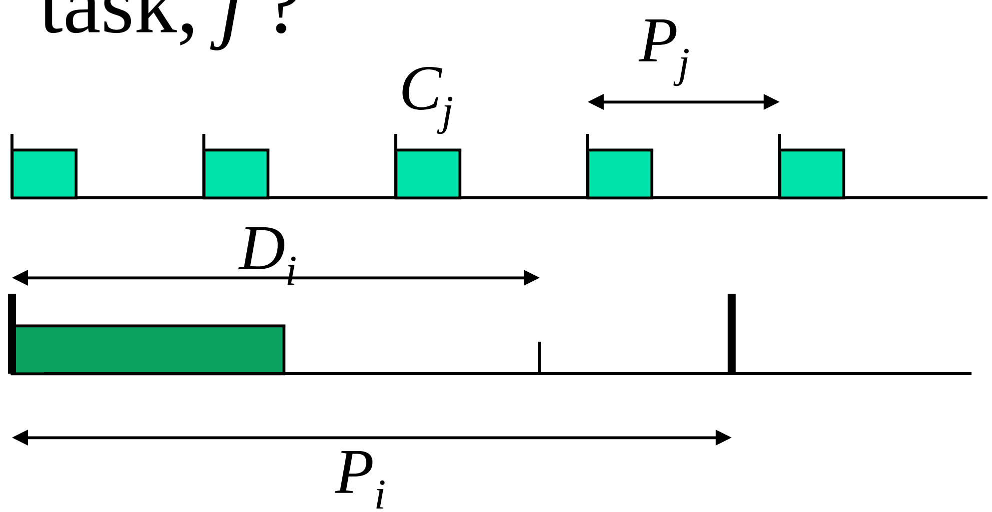
- Schedulability can't be worse than if P_i was reduced to D_i. Thus:

$$\sum_i \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

Problem?

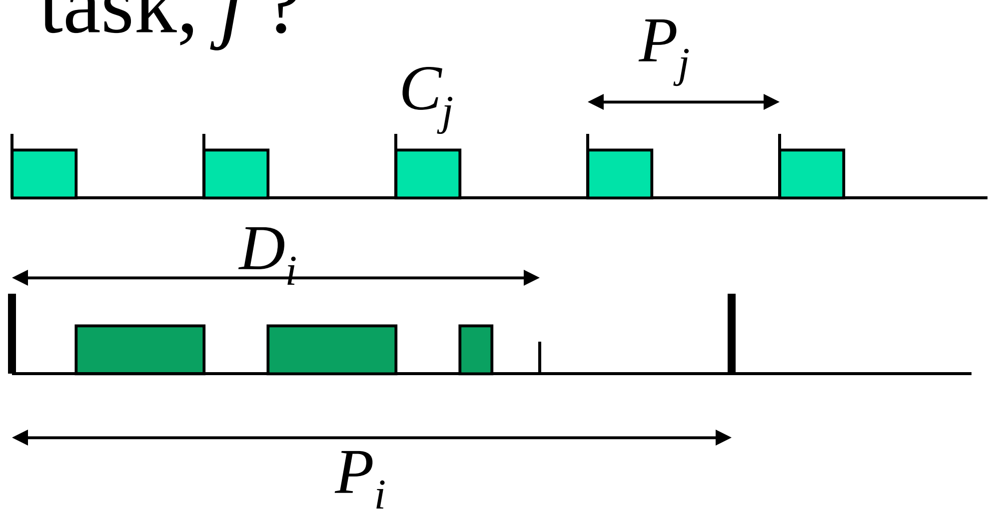
A Better Condition

- Worst case interference from a higher priority task, *j* ?



A Better Condition

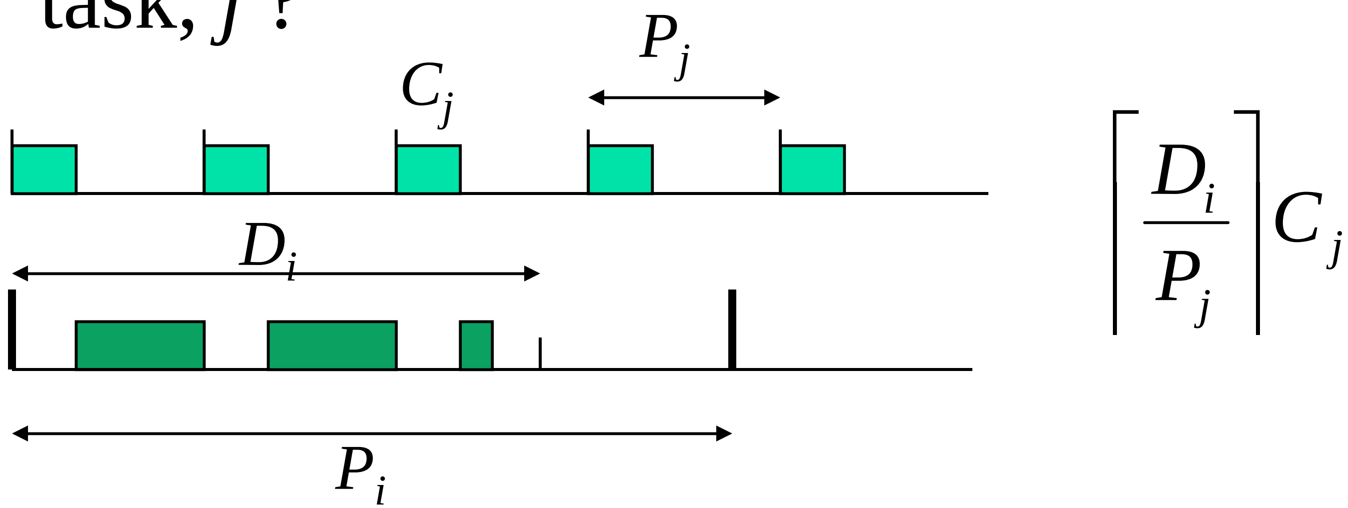
- Worst case interference from a higher priority task, *j* ?



$$\left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

A Better Condition

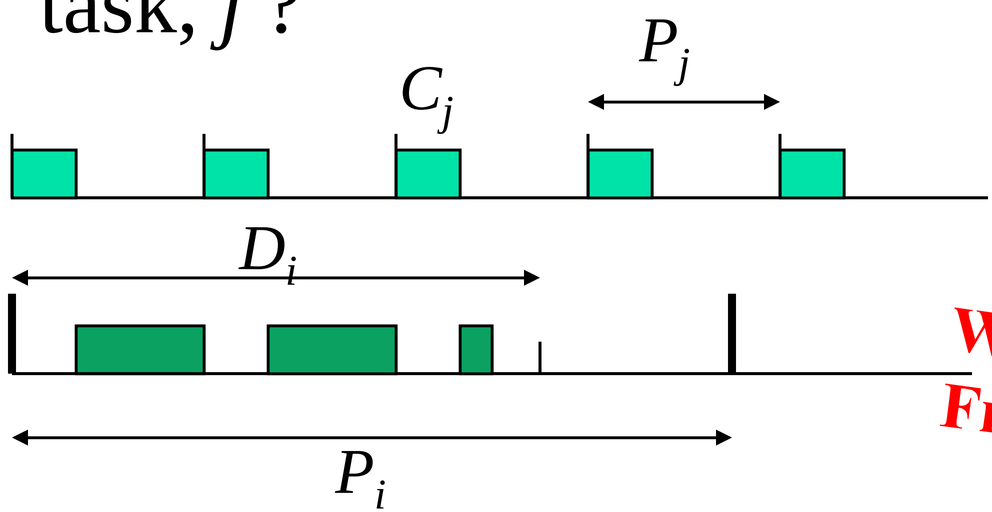
- Worst case interference from a higher priority task, j ?



- Schedulability condition: $C_i + \sum_j \left\lceil \frac{D_i}{P_j} \right\rceil C_j \le D_i$

A Better Condition

- Worst case interference from a higher priority task, *j* ?



$$\left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

Worst case interference, *I*, From higher priority tasks

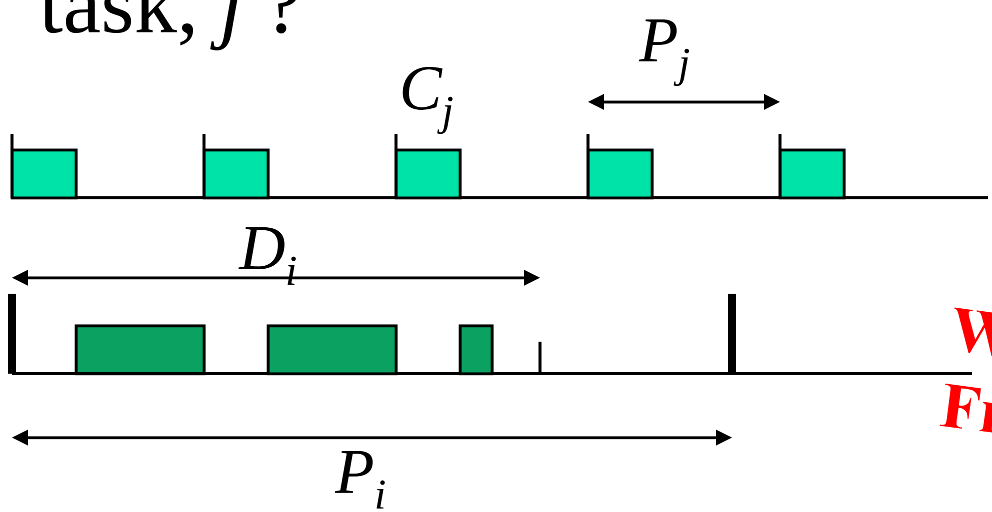
- Schedulability condition: $C_i + \sum_j \left\lceil \frac{D_i}{P_j} \right\rceil C_j \leq D_i$

My exec. time

My deadline

A Better Condition

- Worst case interference from a higher priority task, j ?



$$\left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

Problem?

Worst case interference, *I*,
From higher priority tasks

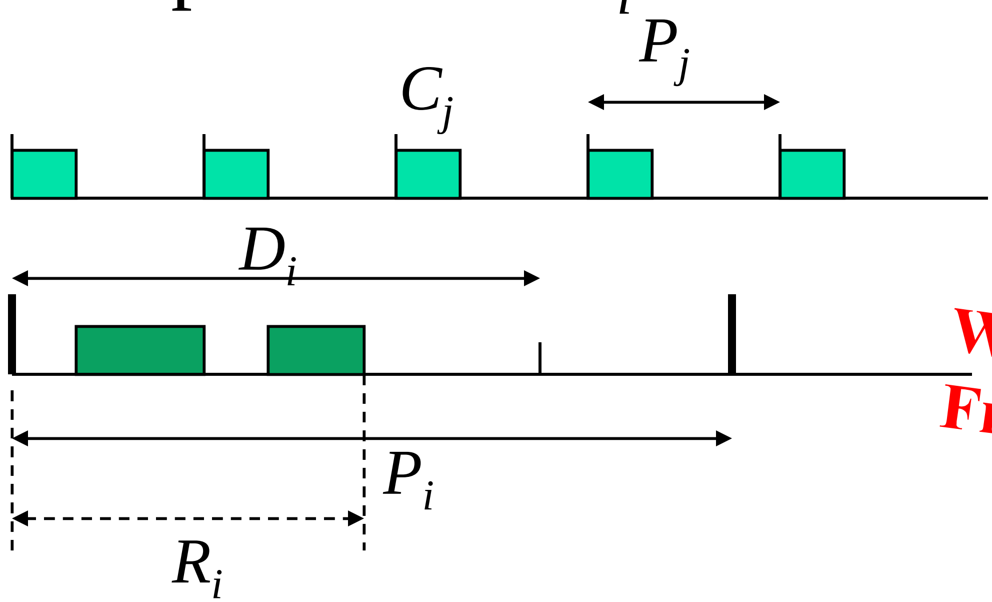
- Schedulability condition: $C_i + \sum_j \left\lceil \frac{D_i}{P_j} \right\rceil C_j \le D_i$

My exec. time

My deadline

An Exact Condition

- Note: Interference exists only during the response time R_i not the entire D_i

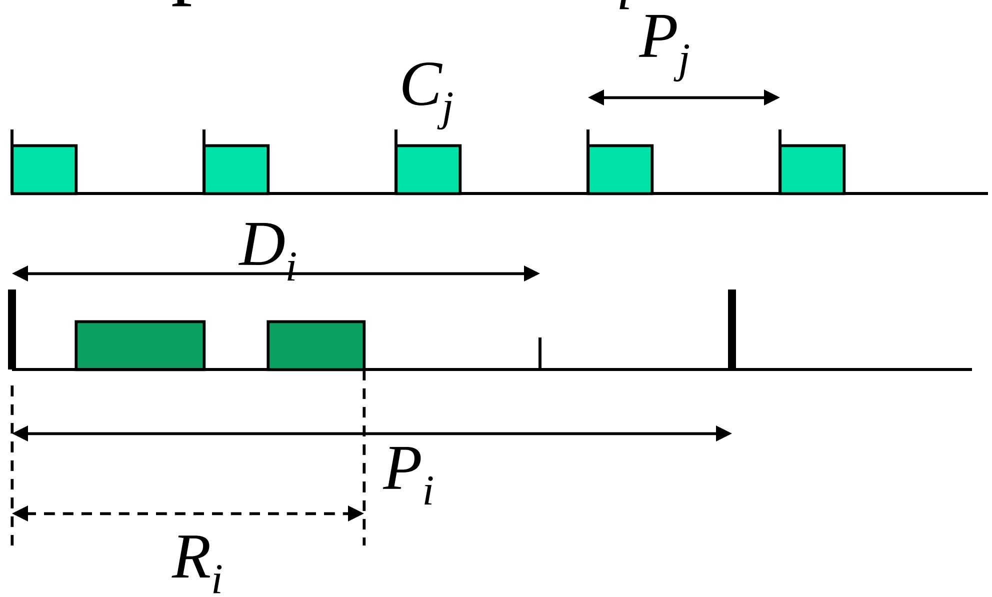


$$I = \sum_j \left\lceil \frac{\overset{R_i}{\cancel{D_i}}}{P_j} \right\rceil C_j$$

Worst case interference, *I*, From higher priority tasks

An Exact Condition

- Note: Interference exists only during the response time R_i not the entire D_i



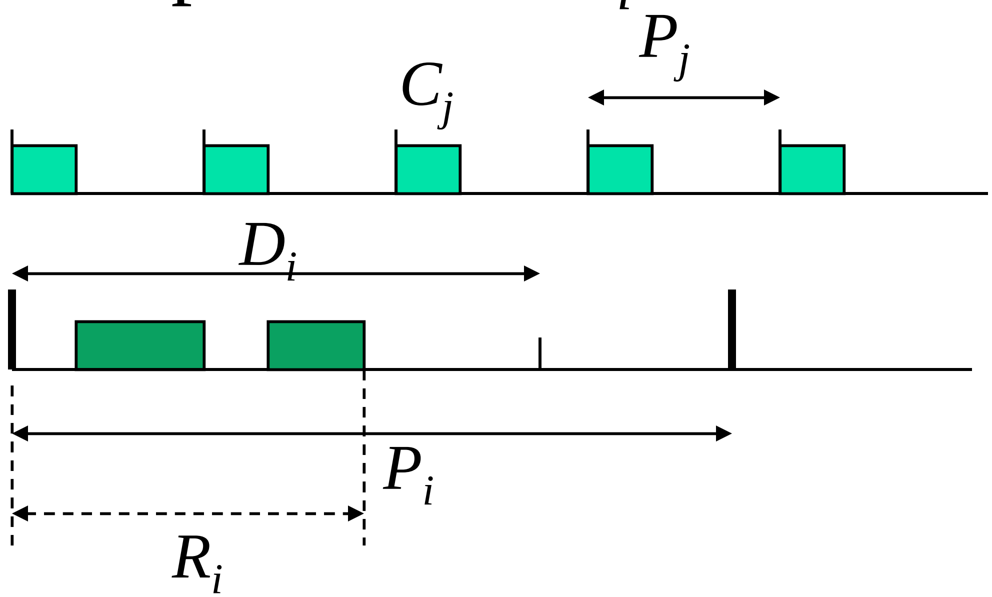
$$I = \sum_j \left\lceil \frac{\overset{R_i}{\cancel{D_i}}}{P_j} \right\rceil C_j$$

where

$$R_i = I + C_i$$

An Exact Condition

- Note: Interference exists only during the response time R_i not the entire D_i



$$I = \sum_j \left\lceil \frac{\overset{R_i}{\cancel{D_i}}}{P_j} \right\rceil C_j$$

where

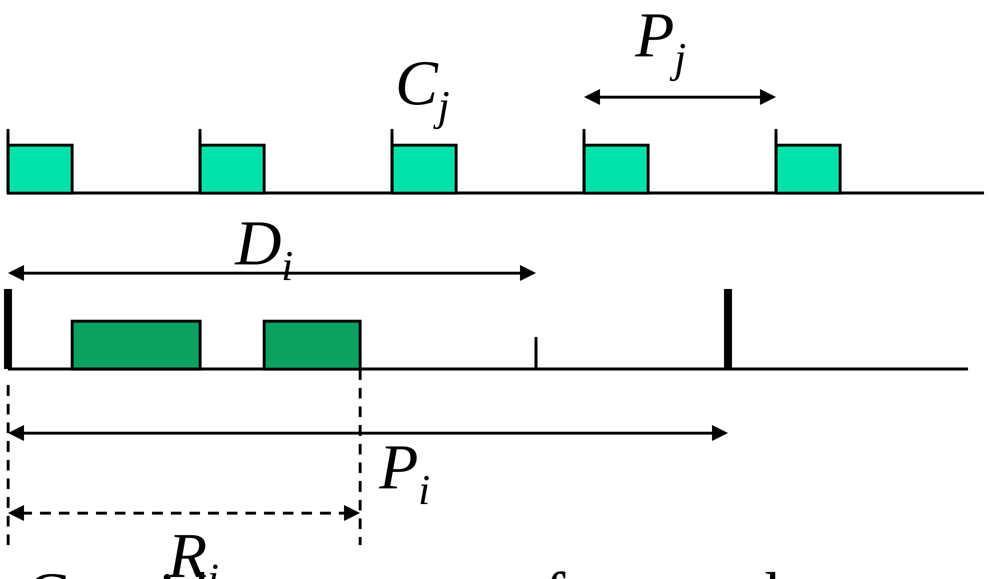
$$R_i = I + C_i$$

Solve iteratively for the smallest R_i that satisfies both equations

$$I = \sum_j \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

$$R_i = I + C_i$$

Example



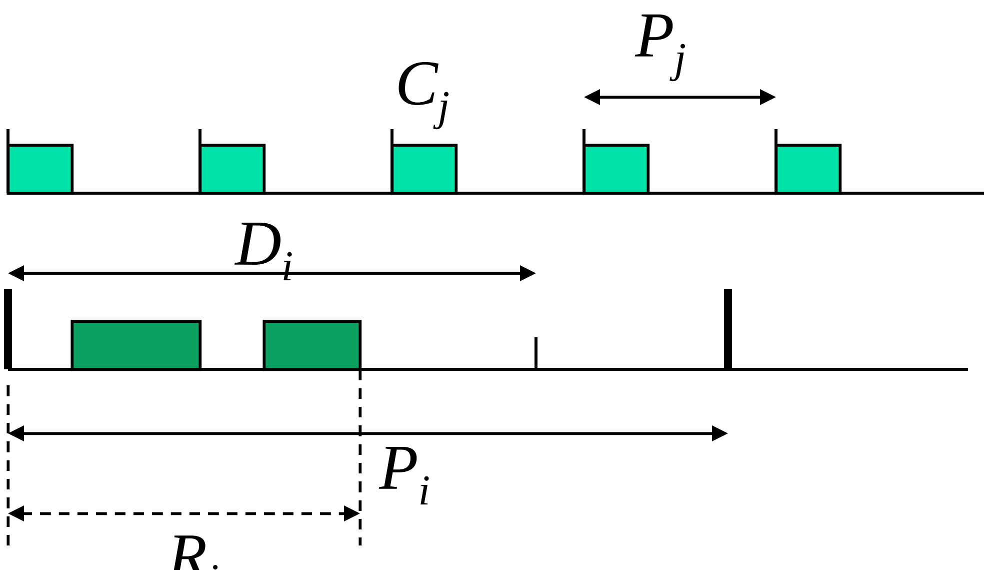
Consider a system of two tasks:

Task 1: P_1=1.7, D_1=0.5, C_1=0.5
Task 2: P_2=8, D_2=3.2, C_2=2

$$I = \sum_j \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

$$R_i = I + C_i$$

Example



$$I^{(0)} = C_1 = 0.5$$

$$R_2^{(0)} = I^{(0)} + C_2 = 2.5$$

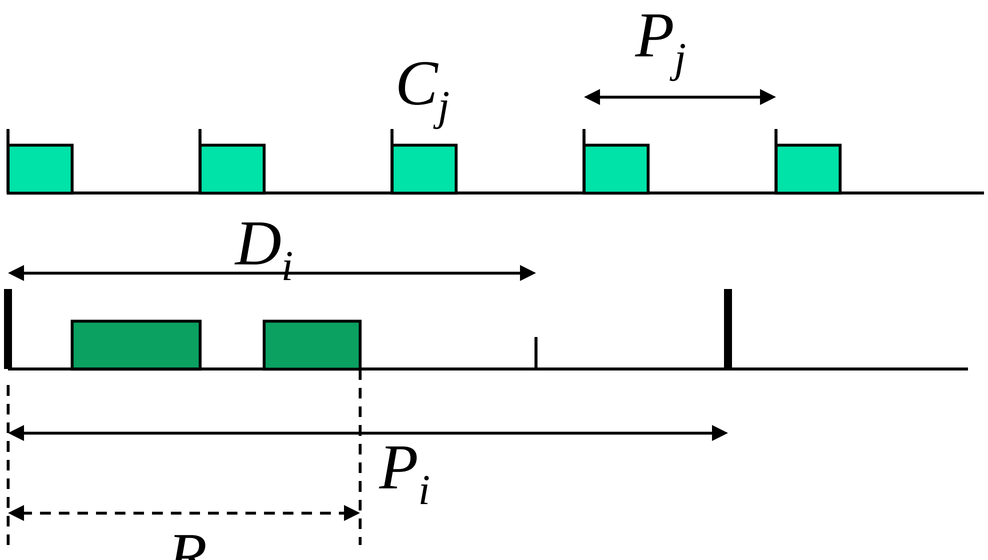
Consider a system of two tasks:

Task 1: P_1=1.7, D_1=0.5, C_1=0.5
Task 2: P_2=8, D_2=3.2, C_2=2

$$I = \sum_j \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

$$R_i = I + C_i$$

Example



$$I^{(0)} = C_1 = 0.5$$

$$R_2^{(0)} = I^{(0)} + C_2 = 2.5$$

$$I^{(1)} = \left\lceil \frac{R_2^{(0)}}{P_1} \right\rceil C_1 = \left\lceil \frac{2.5}{1.7} \right\rceil 0.5 = 1$$

$$R_2^{(1)} = I^{(1)} + C_2 = 3$$

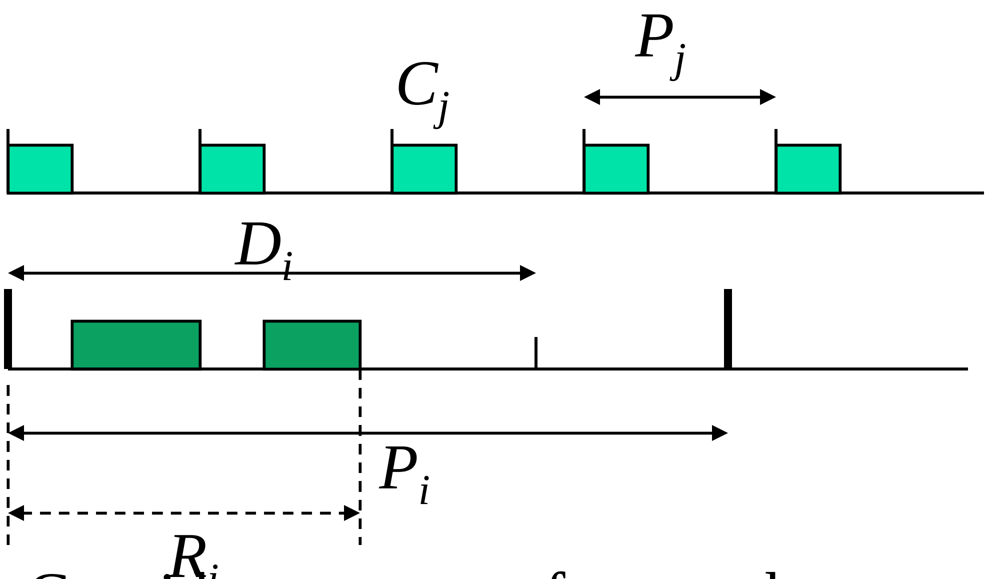
Consider a system of two tasks:

Task 1: P_1=1.7, D_1=0.5, C_1=0.5
Task 2: P_2=8, D_2=3.2, C_2=2

$$I = \sum_j \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

$$R_i = I + C_i$$

Example



Consider a system of two tasks:

Task 1: P_1=1.7, D_1=0.5, C_1=0.5
Task 2: P_2=8, D_2=3.2, C_2=2

$$I^{(0)} = C_1 = 0.5$$

$$R_2^{(0)} = I^{(0)} + C_2 = 2.5$$

$$I^{(1)} = \left\lceil \frac{R_2^{(0)}}{P_1} \right\rceil C_1 = \left\lceil \frac{2.5}{1.7} \right\rceil 0.5 = 1$$

$$R_2^{(1)} = I^{(1)} + C_2 = 3$$

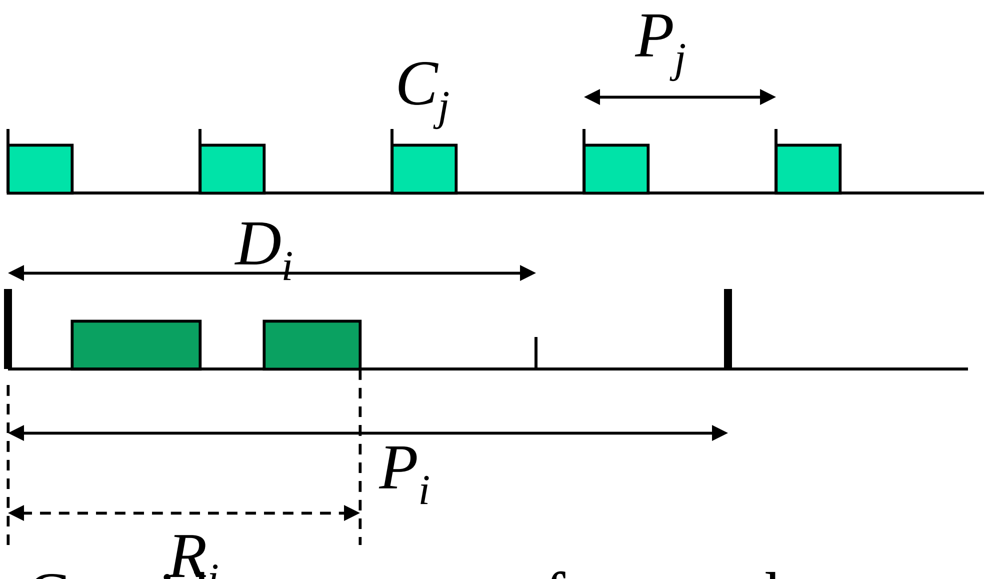
$$I^{(2)} = \left\lceil \frac{R_2^{(1)}}{P_1} \right\rceil C_1 = \left\lceil \frac{3}{1.7} \right\rceil 0.5 = 1$$

$$R_2^{(2)} = I^{(2)} + C_2 = 3$$

$$I = \sum_j \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

$$R_i = I + C_i$$

Example



Consider a system of two tasks:

Task 1: P_1=1.7, D_1=0.5, C_1=0.5
Task 2: P_2=8, D_2=3.2, C_2=2

$$I^{(0)} = C_1 = 0.5$$

$$R_2^{(0)} = I^{(0)} + C_2 = 2.5$$

$$I^{(1)} = \left\lceil \frac{R_2^{(0)}}{P_1} \right\rceil C_1 = \left\lceil \frac{2.5}{1.7} \right\rceil 0.5 = 1$$

$$R_2^{(1)} = I^{(1)} + C_2 = 3$$

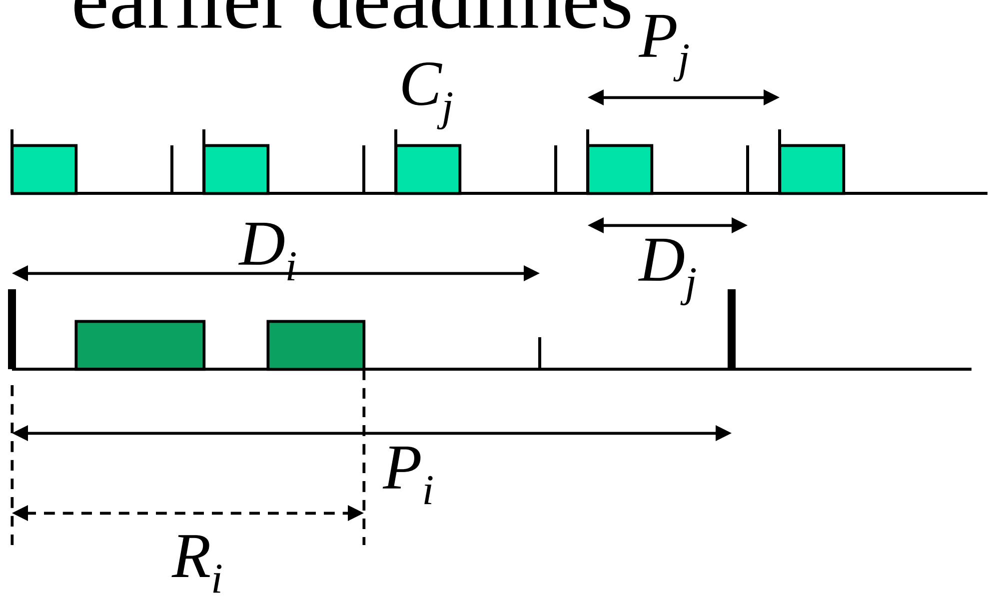
$$I^{(2)} = \left\lceil \frac{R_2^{(1)}}{P_1} \right\rceil C_1 = \left\lceil \frac{3}{1.7} \right\rceil 0.5 = 1$$

$$R_2^{(2)} = I^{(2)} + C_2 = 3$$

$3 < 3.2 \rightarrow$ Ok!

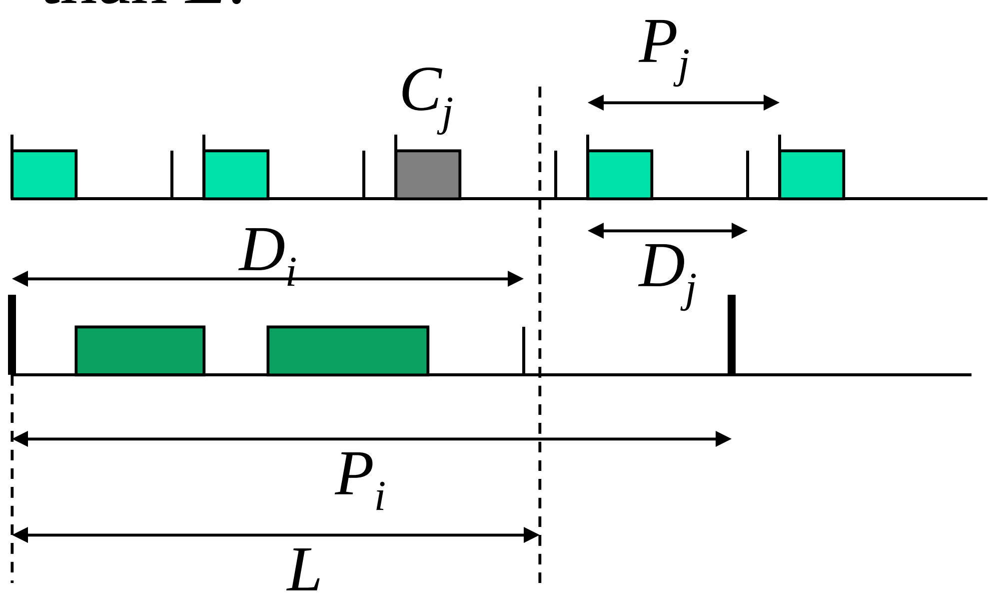
EDF and Processor Demand

- Interference is due to only those tasks with earlier deadlines



EDF and Processor Demand

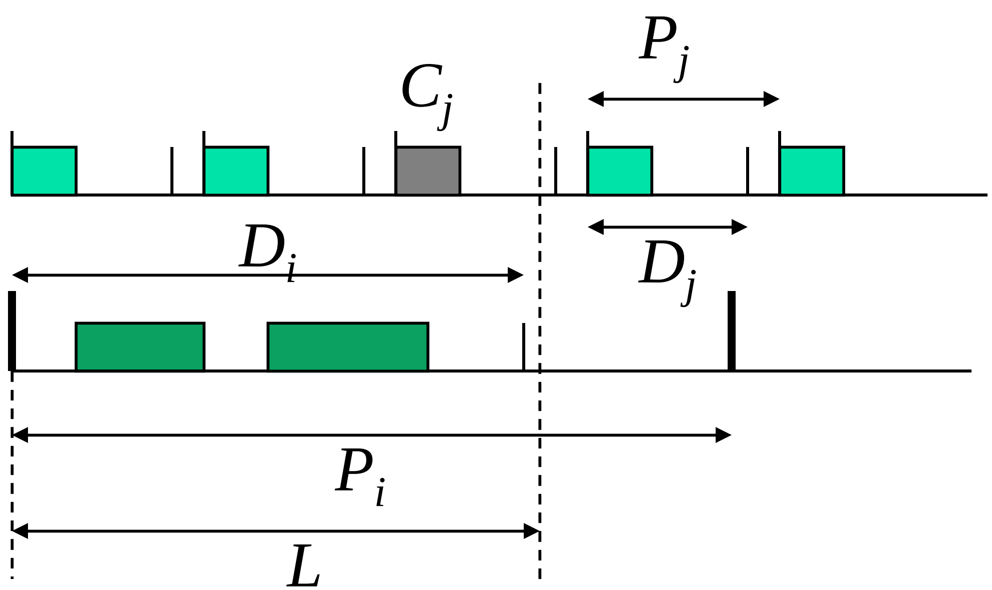
- Consider demand on the processor due to tasks whose deadlines have passed
- Within any time interval, *L*, the demand must be less than *L*.



$$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \le L$$

The Processor Demand Schedulability Test for EDF

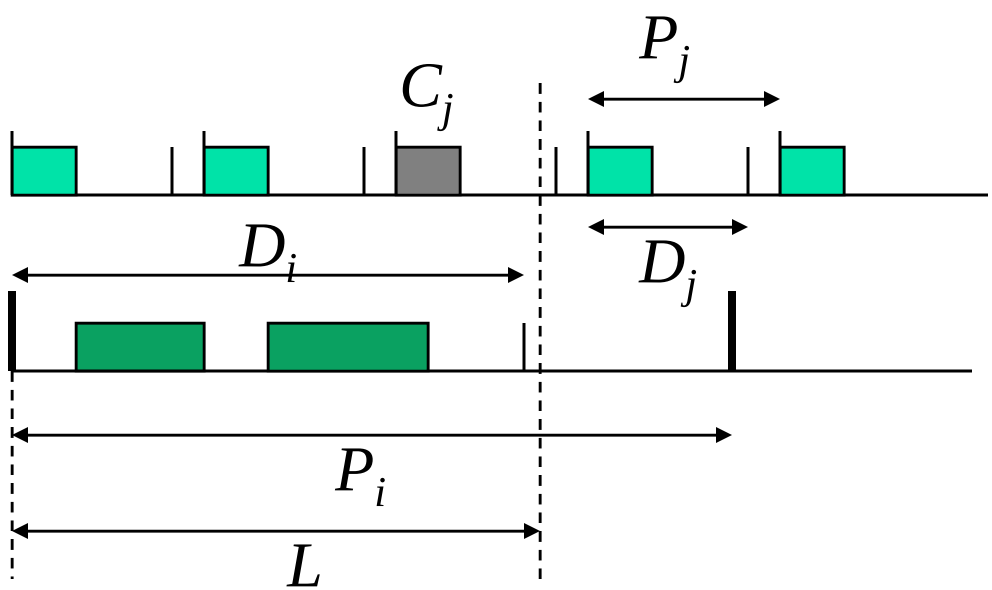
- Checking the schedulability conditions for all L is not possible.



$$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \le L$$

The Processor Demand Schedulability Test for EDF

- Checking the schedulability conditions for all L is not possible.
  - Observation 1: Check only within a hyper-period (schedule repeats itself)



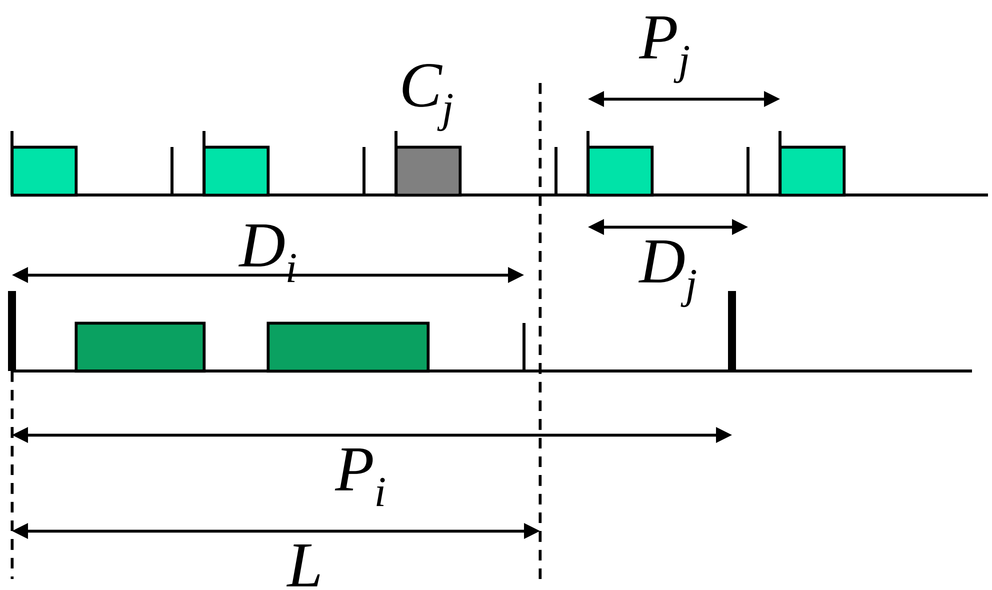
$$\sum_{i=1}^{n}\left\lceil \frac{L-D_i}{P_i} \right\rceil C_i \le L$$

The Processor Demand Schedulability Test for EDF

- Checking t for all L is not possible.

  Least common multiple of all task periods

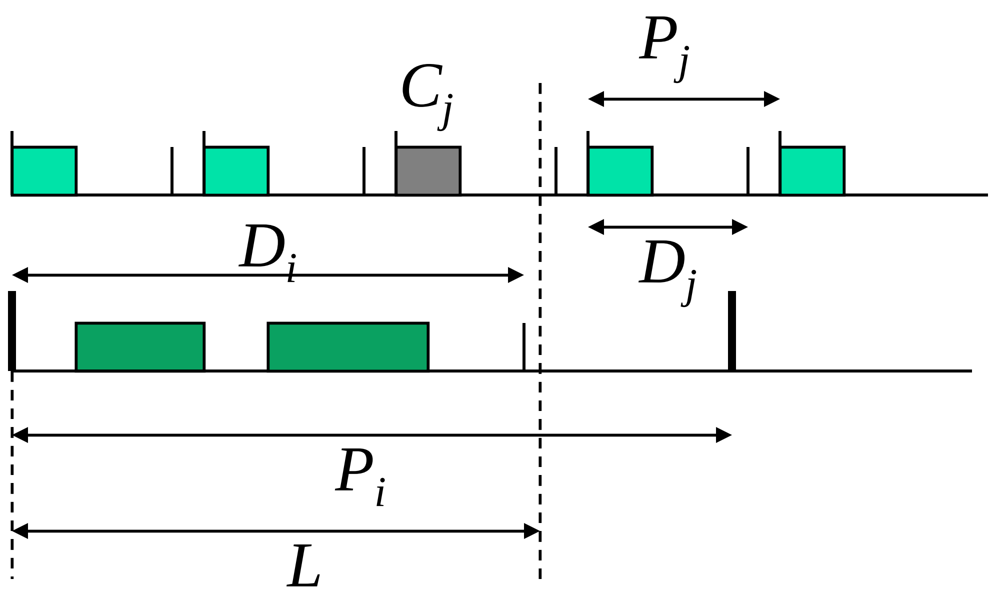
  - Observation 1: Check only within a hyper-period (schedule repeats itself)



$$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \le L$$

The Processor Demand Schedulability Test for EDF

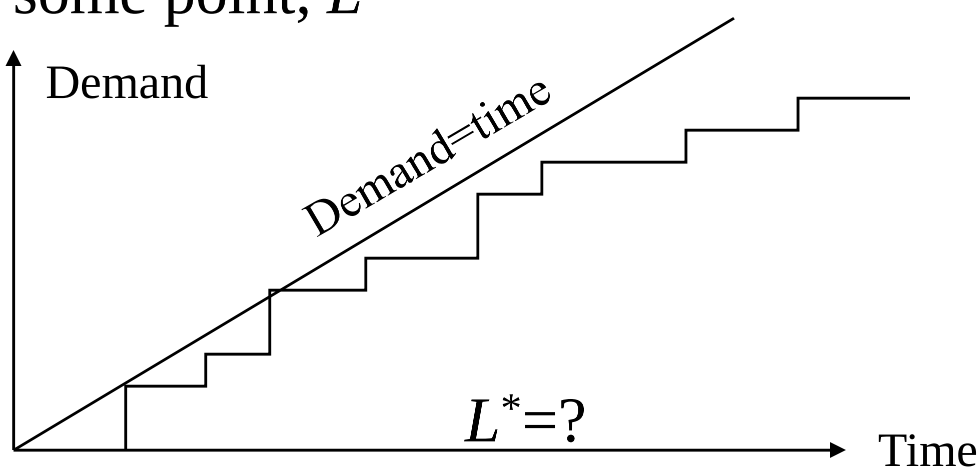
- Checking the schedulability conditions for all L is not possible.
  - Observation 1: Check only within a hyper-period (schedule repeats itself afterwards)
  - Observation 2: Check only on absolute deadlines



$$\sum_{i=1}^{n} \left\lfloor \frac{L - D_i}{P_i} \right\rfloor C_i \leq L$$

The Processor Demand Schedulability Test for EDF

- Checking the schedulability conditions for all L is not possible.
  - Observation 1: Check only within a hyper-period (schedule repeats itself afterwards)
  - Observation 2: Check only on absolute deadlines
  - Observation 3: If $U < 1$, Demand is trivially satisfied after some point, L^*



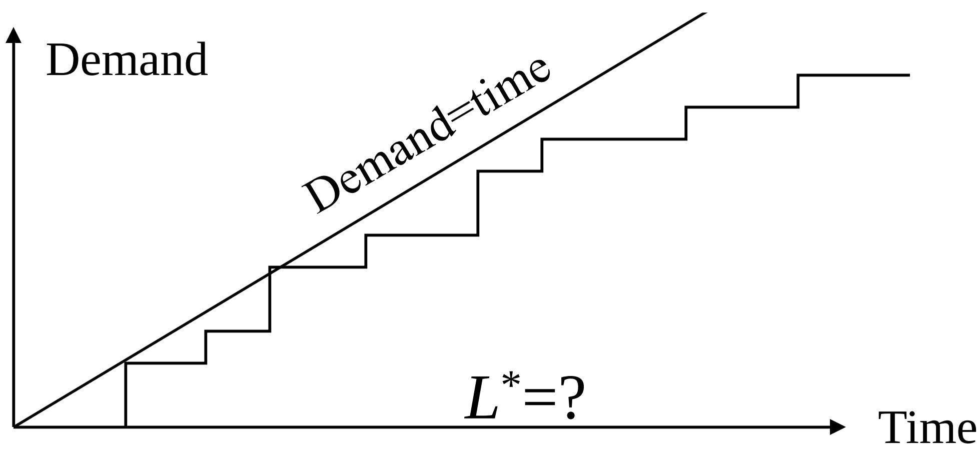
$$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \leq L$$

The Processor Demand Schedulability Test for EDF

$$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \le L$$

- Deriving L^*

$$\left\lceil \frac{t - D_i}{P_i} \right\rceil \le \frac{t - D_i}{P_i} + 1$$



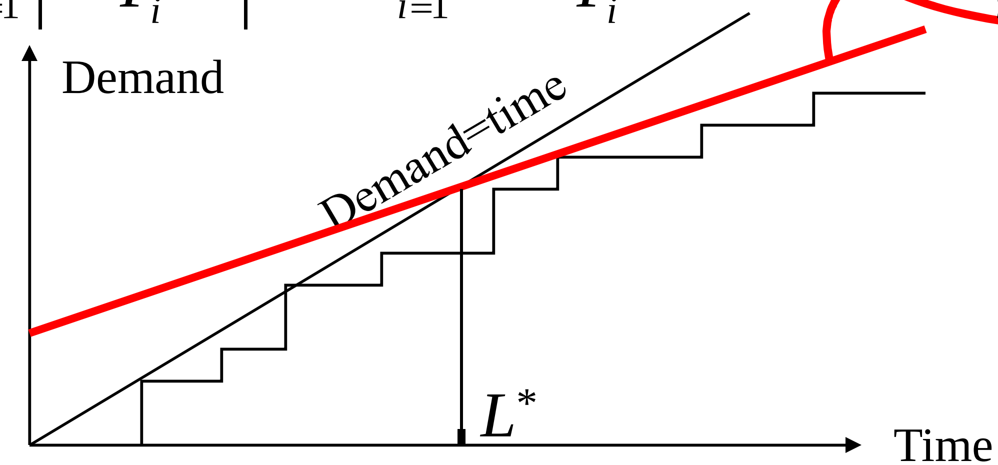
The Processor Demand Schedulability Test for EDF

$$\sum_{i=1}^{n}\left\lceil \frac{L-D_i}{P_i} \right\rceil C_i \le L$$

- Deriving L^*

$$\left\lceil \frac{t-D_i}{P_i} \right\rceil \le \frac{t-D_i}{P_i} + 1$$

$$\sum_{i=1}^{n}\left\lceil \frac{t-D_i}{P_i} \right\rceil C_i \le \sum_{i=1}^{n} \frac{t-D_i+P_i}{P_i} C_i = tU + \sum_{i=1}^{n}(P_i - D_i)U_i$$



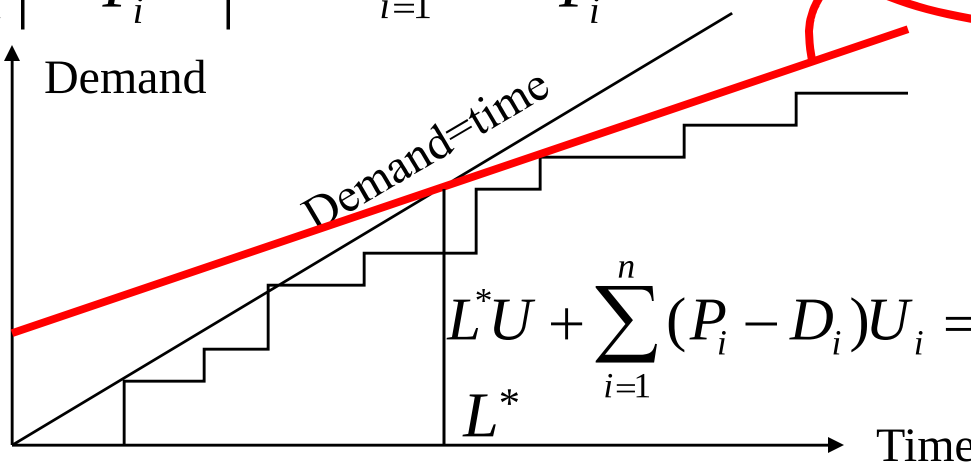
The Processor Demand Schedulability Test for EDF

$$\sum_{i=1}^{n}\left\lceil \frac{L-D_i}{P_i} \right\rceil C_i \le L$$

- Deriving L^*

$$\left\lceil \frac{t-D_i}{P_i} \right\rceil \le \frac{t-D_i}{P_i}+1$$

$$\sum_{i=1}^{n}\left\lceil \frac{t-D_i}{P_i} \right\rceil C_i \le \sum_{i=1}^{n} \frac{t-D_i+P_i}{P_i} C_i = tU + \sum_{i=1}^{n}(P_i-D_i)U_i$$



$$L^*U + \sum_{i=1}^{n}(P_i-D_i)U_i = L^*, \quad \Rightarrow L^* = \frac{\sum_{i=1}^{n}(P_i-D_i)U_i}{1-U}$$

The Processor Demand Schedulability Test for EDF

- Check if: $$\sum_{i=1}^{n} \left\lceil \frac{L - D_i}{P_i} \right\rceil C_i \le L$$

for all L = absolute deadlines in the interval $[0, L^*]$, where:

$$L^* = \frac{\sum_{i=1}^{n} (P_i - D_i) U_i}{1 - U}$$