Periodic task scheduling

Static priorities

- ★Better utilization bounds
- ★Deadlines less than periods
- ★Exact test for schedulability

Quick review

- Why is rate monotonic scheduling optimal (among static priority policies)?
  - Critical instant theorem: The worst-case execution time of a job when tasks are scheduled with fixed priorities occurs when jobs belonging to all tasks release at the same instant
  - It is sufficient, then, to verify that the job that is released at the critical instant meets its deadline
  - In this worst case, rate monotonic scheduling is optimal (easy to see; if tasks are feasibly scheduled in any other order, swap based on deadlines)
- Utilization bound and optimality of EDF
  - The utilization bound is 1 (or 100%)
  - EDF is optimal because no policy can do better (may do as well but not better)

Exercise

Know Your Worst Case Scenario

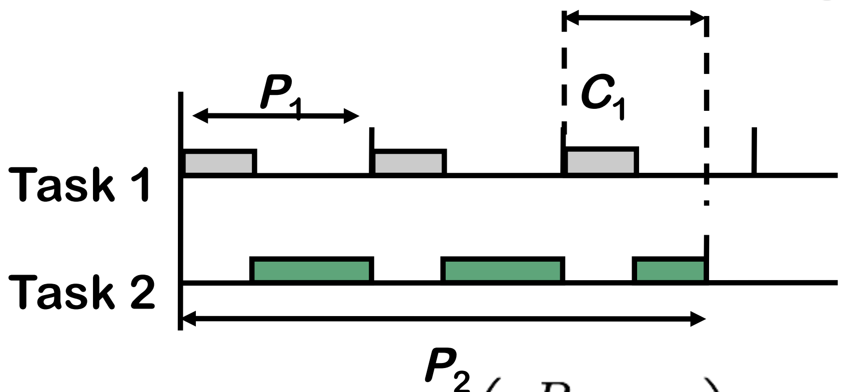
- Consider a periodic system of two tasks
- Let $U_i = C_i / P_i$ (for $i = 1,2$)
- What is the maximum value of $\Pi_i(1\text{-}U_i)$ for a schedulable system?
- Motivation: There may be other functions of a task set rather then just utilization that also indicate schedulability.

Finding the utilization bound for RM scheduling

The minimum utilization case

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

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



To minimize *U*, we must have

$$\lfloor \frac{P_2}{P_1} \rfloor = 1$$

$$C_2 = P_2 - C_1 \left(\lfloor \frac{P_2}{P_1} \rfloor + 1 \right)$$

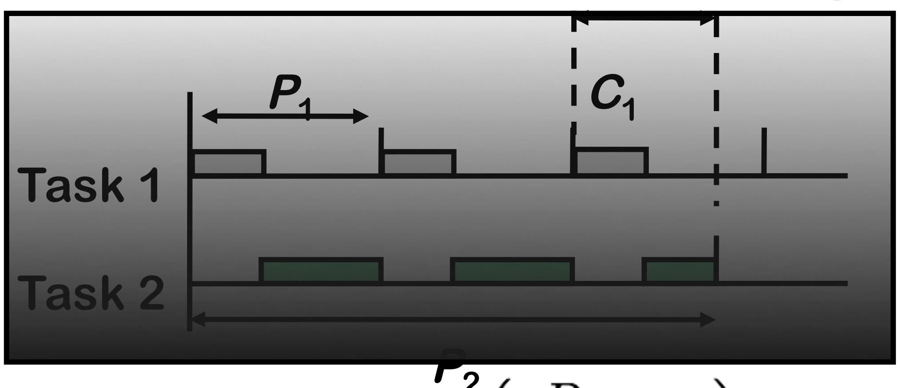
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = 1 + \frac{C_1}{P_2} \left[\frac{P_2}{P_1} - \lfloor \frac{P_2}{P_1} \rfloor - 1 \right]$$

Finding the utilization bound for RM scheduling

The minimum utilization case

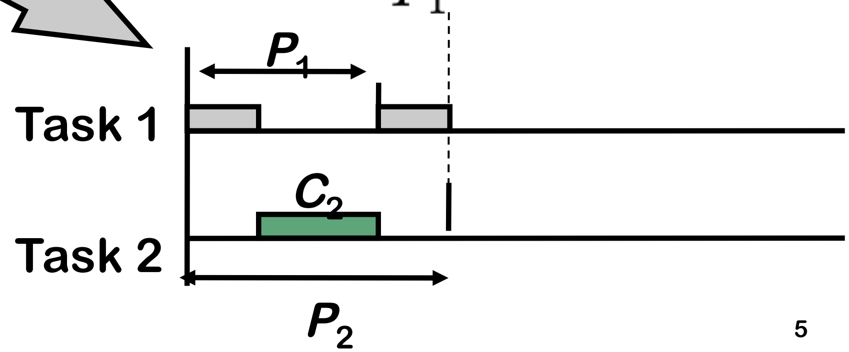
$$C_1 = P_2 - \lfloor \frac{P_2}{P_1} \rfloor P_1 = P_1 \left(\frac{P_2}{P_1} - \lfloor \frac{P_2}{P_1} \rfloor \right)$$

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



To minimize *U*, we must have

$$\lfloor \frac{P_2}{P_1} \rfloor = 1$$



$$C_2 = P_2 - C_1 \left(\lfloor \frac{P_2}{P_1} \rfloor + 1 \right)$$

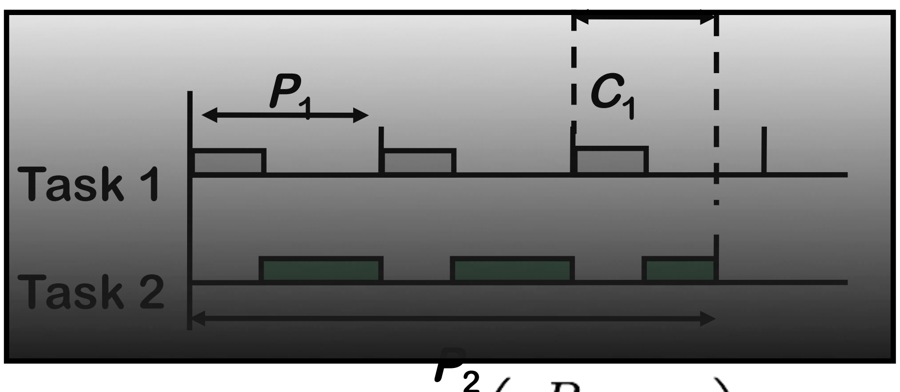
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = 1 + \frac{C_1}{P_2} \left[\frac{P_2}{P_1} - \lfloor \frac{P_2}{P_1} \rfloor - 1 \right]$$

Finding the utilization bound for RM scheduling

The minimum utilization case

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

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

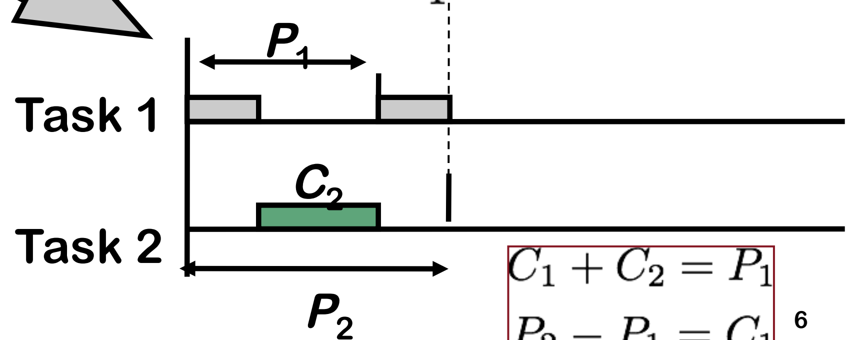


$$C_2 = P_2 - C_1 \left(\lfloor \frac{P_2}{P_1} \rfloor + 1 \right)$$

$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = 1 + \frac{C_1}{P_2} \left[\frac{P_2}{P_1} - \lfloor \frac{P_2}{P_1} \rfloor - 1 \right]$$

To minimize *U*, we must have

$$\lfloor \frac{P_2}{P_1} \rfloor = 1$$



$$\boxed{\begin{aligned} C_1 + C_2 &= P_1 \\ P_2 - P_1 &= C_1 \end{aligned}}$$

Solutions

Critically schedulable

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

Schedulable

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

Hyperbolic bound

Solutions

Critically schedulable

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

Generalizes to task sets with n tasks

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

Hyperbolic bound

Hyperbolic bound for rate monotonic scheduling

- A set of periodic tasks is schedulable if

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

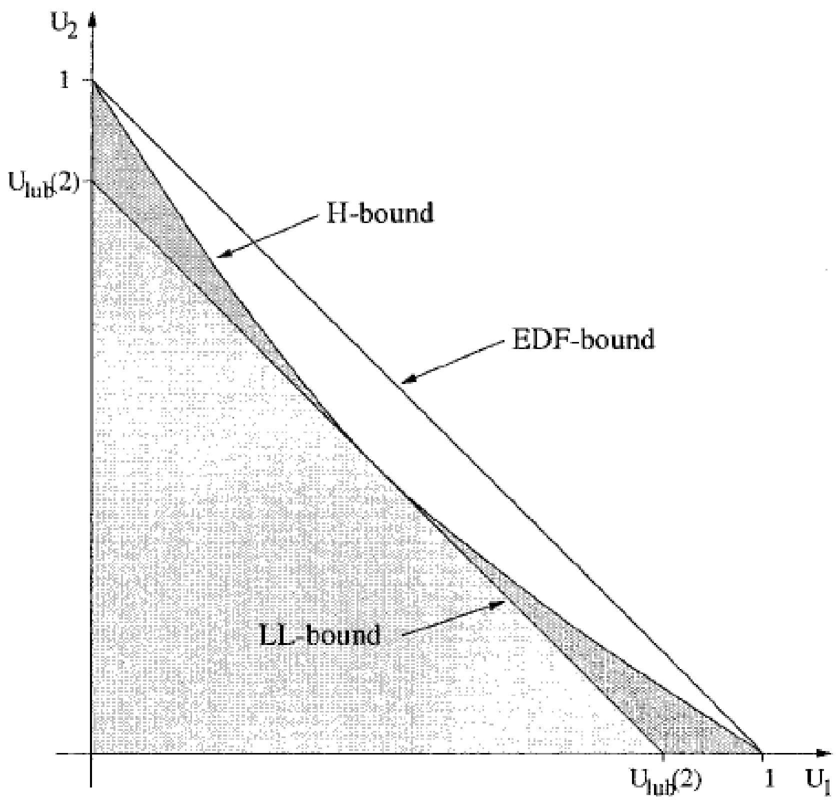


Figure 4.9 Schedulability bounds for RM and EDF in the utilization space.

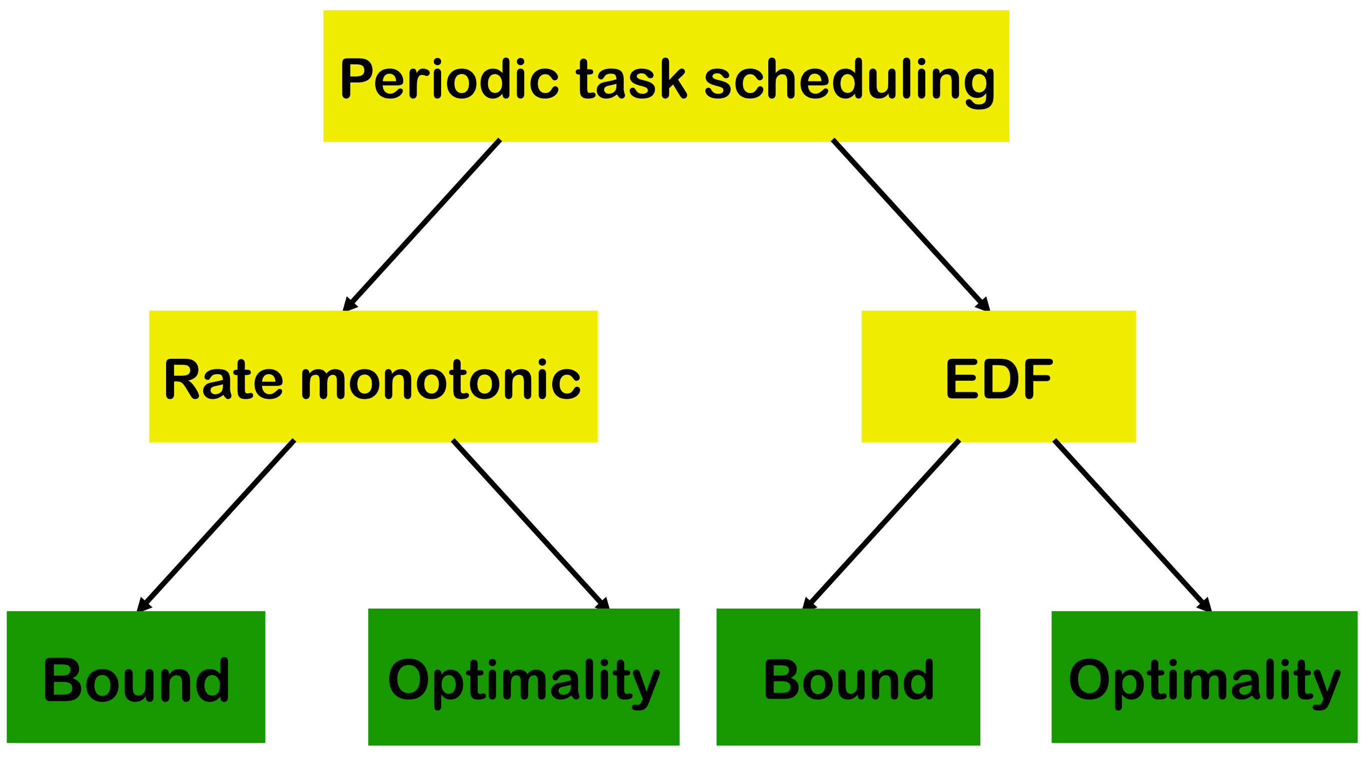
Hyperbolic bound for rate monotonic scheduling

- A set of periodic tasks is schedulable if

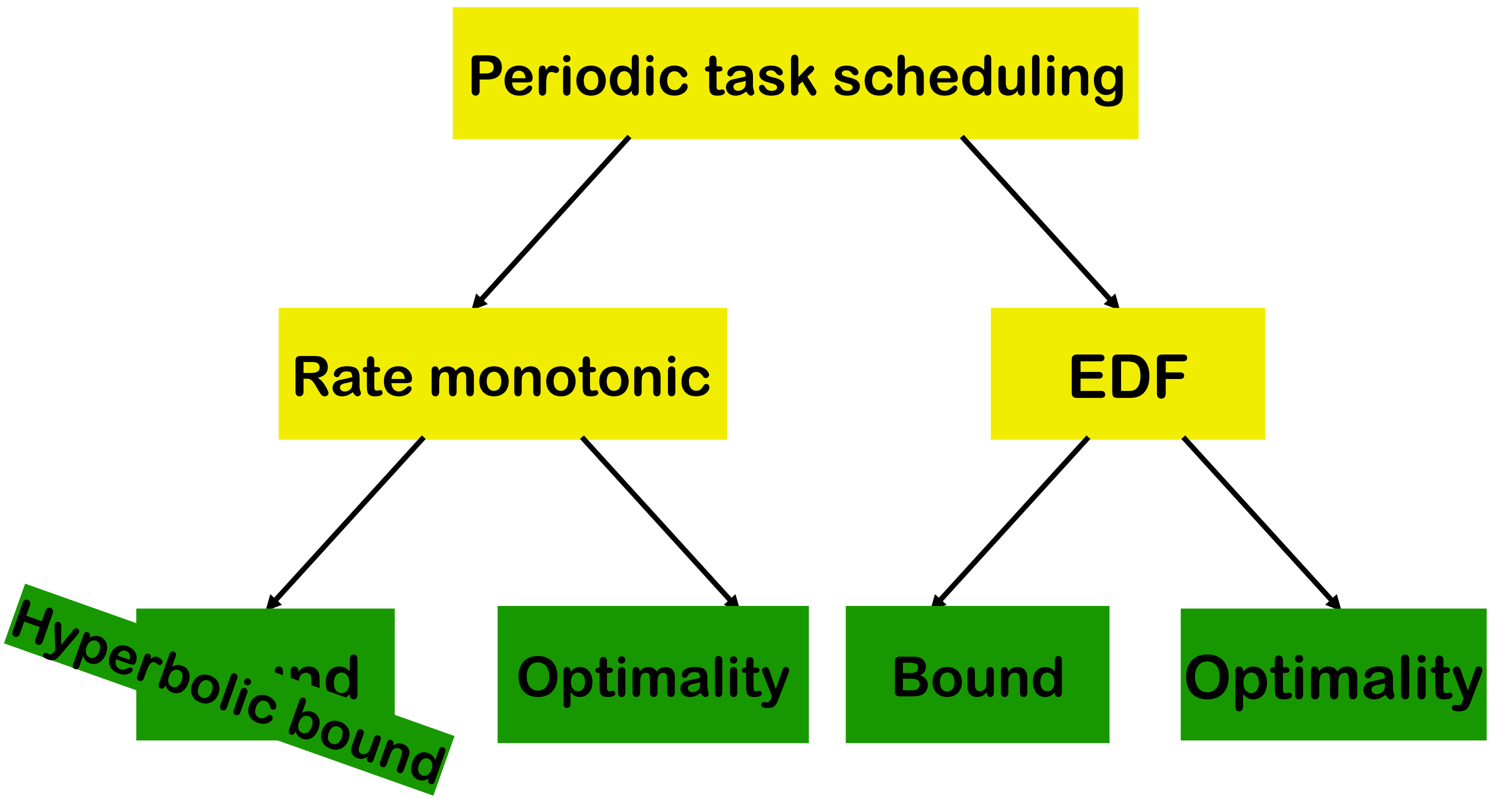
$$\prod_i (U_i + 1) \leq 2$$

- It is a better bound than the Liu and Layland bound $U \leq n(2^{1/n} - 1)$
- Example: consider a system with two tasks such that U_1=0.8 and U_2=0.1
- U = 0.9 > 0.83 (unschedulable according to the Liu and Layland bound)
- $(1+U_1)(1+U_2)$ = (1.8)(1.1) = 1.98 < 2 (schedulable according to the hyperbolic bound)

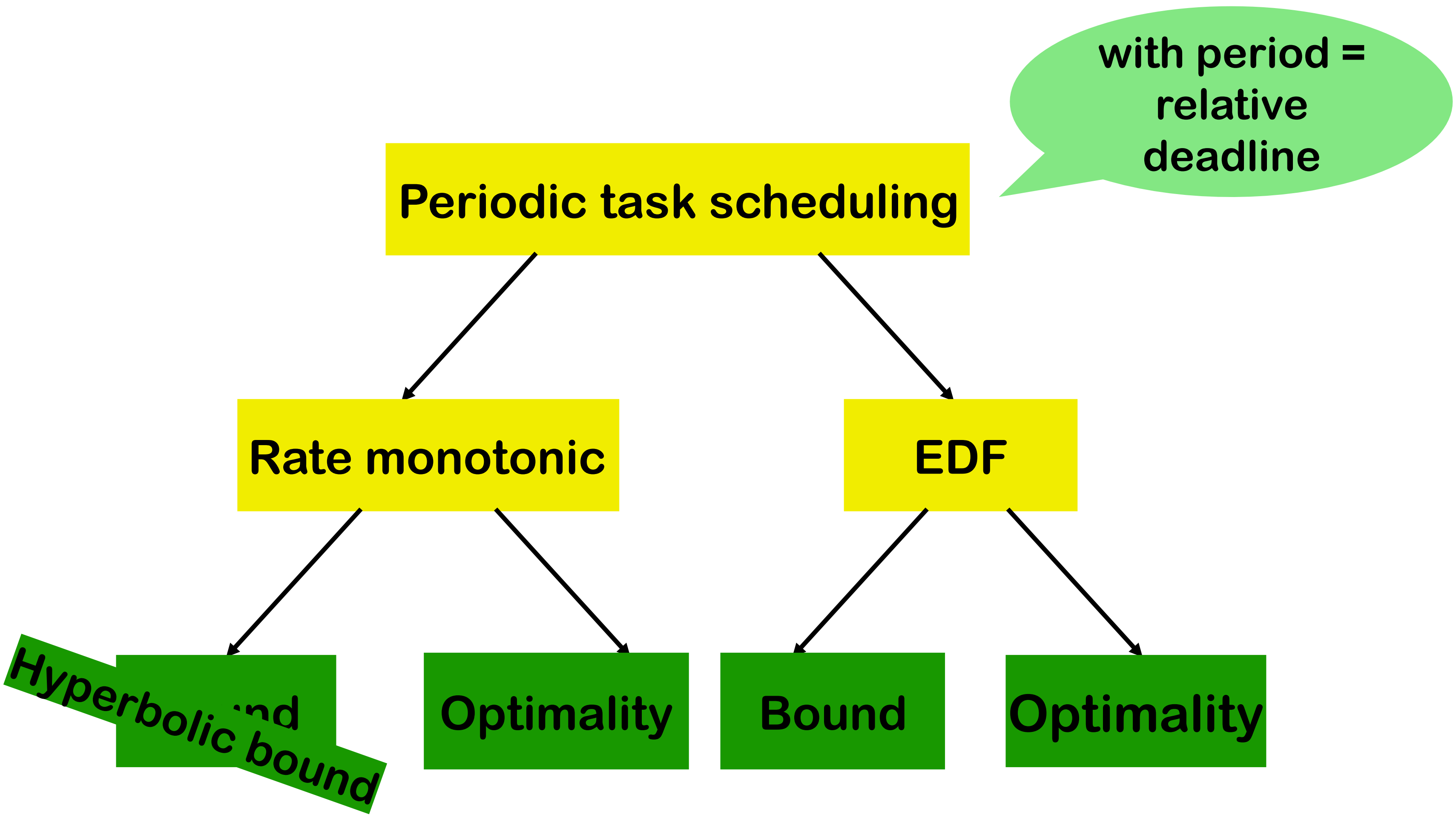
Scheduling taxonomy



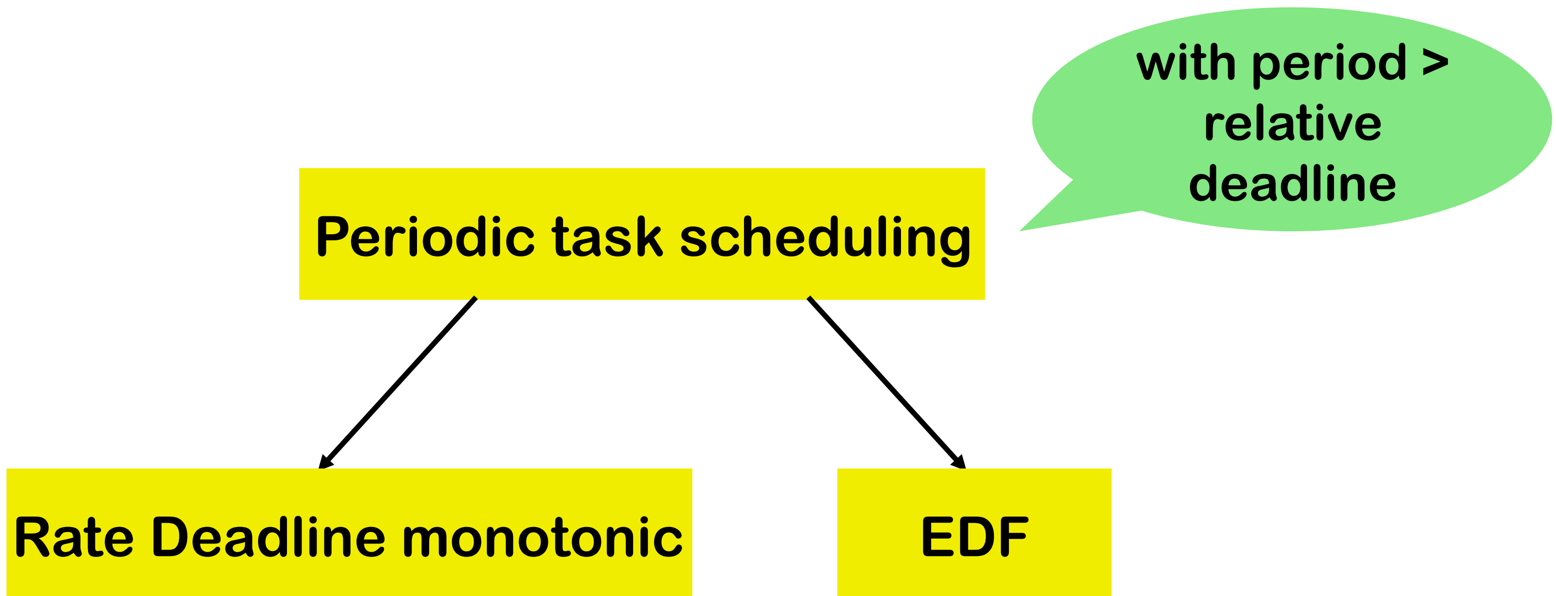
Scheduling taxonomy



Scheduling taxonomy

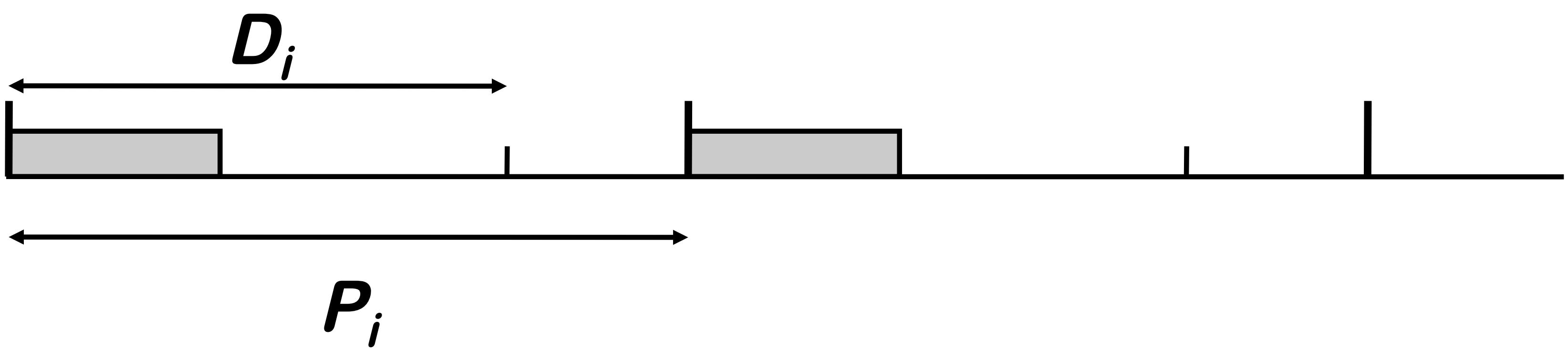


Scheduling taxonomy



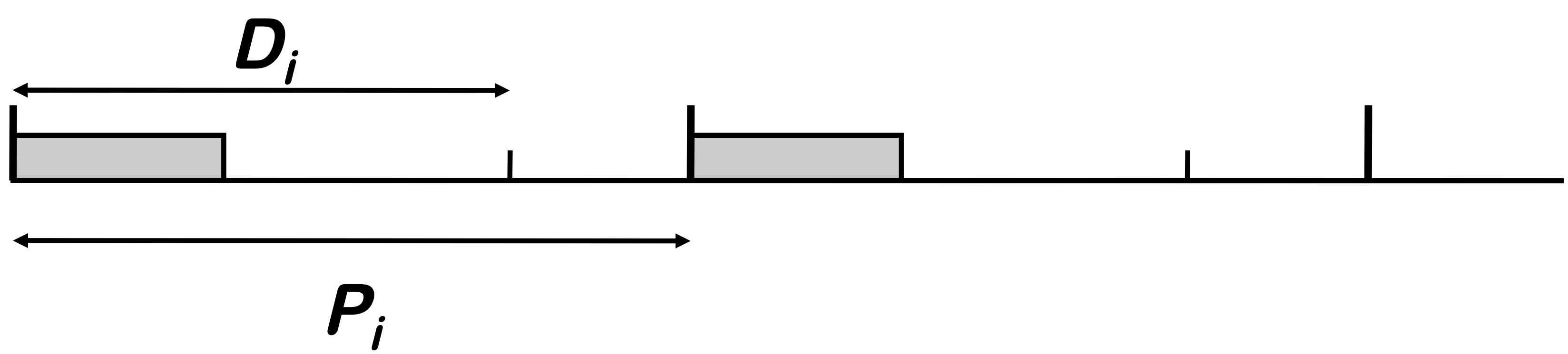
Deadline monotonic scheduling

- 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 $D_i < P_i$.

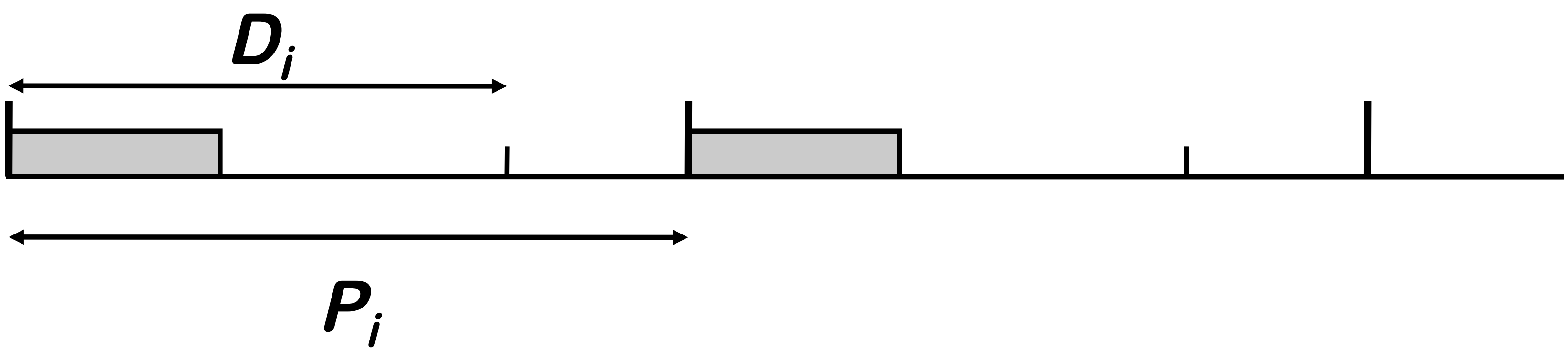


•What is the schedulability condition?

•Can not be worse than when the period of each task is reduce to D_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 $D_i < P_i$.



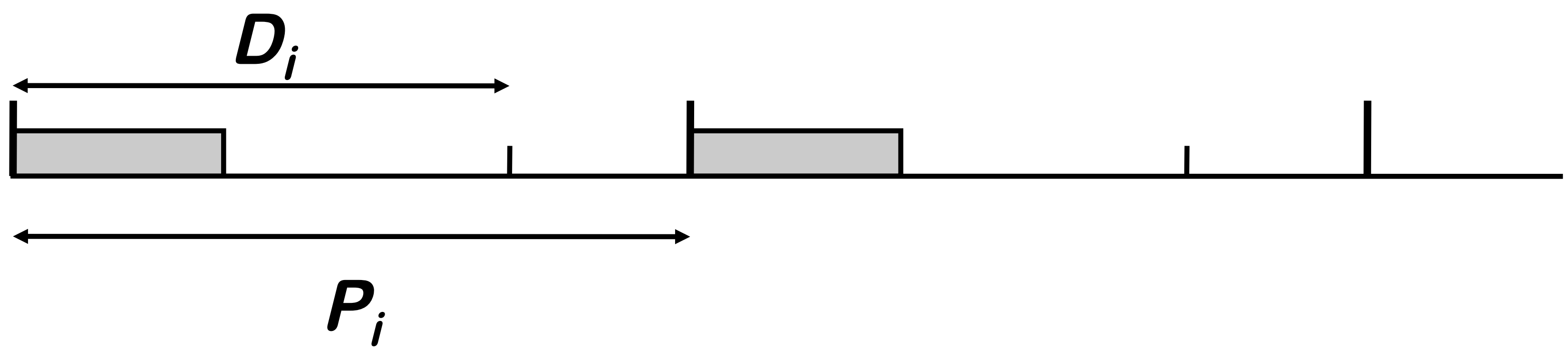
•What is the schedulability condition?

•Can not be worse than when the period of each task is reduced to D_i.

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

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?

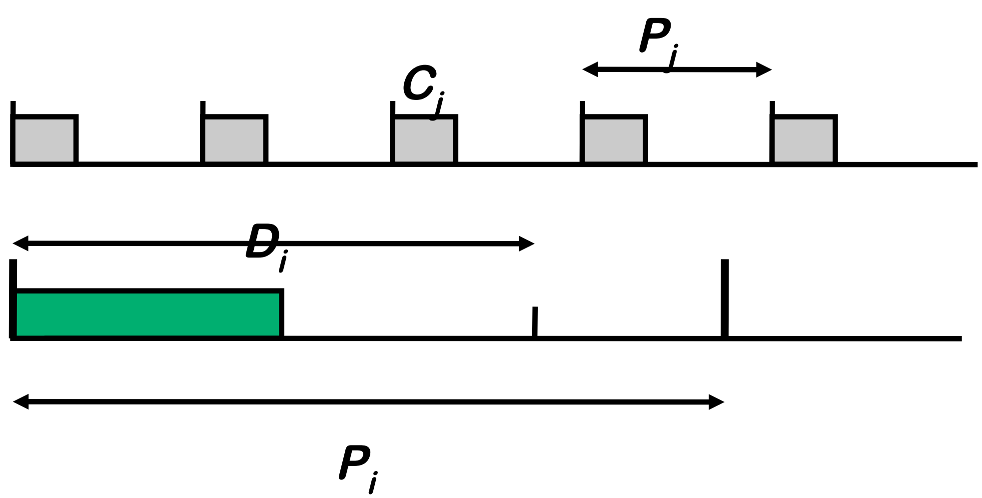
•Can not be worse than when the period of each task is reduced to D_i.

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

What is the problem?

A better condition for schedulability

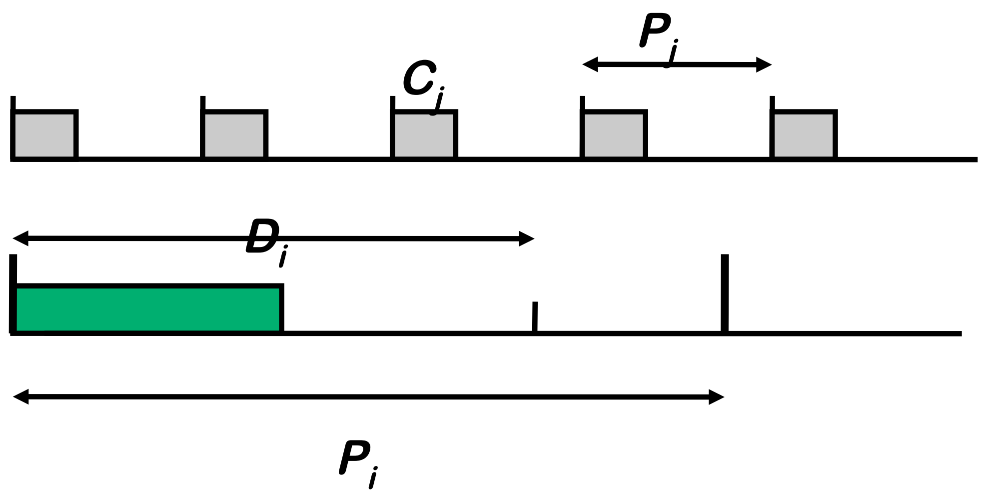
- Worst case interference from a higher priority task, *j*?



A better condition for schedulability

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

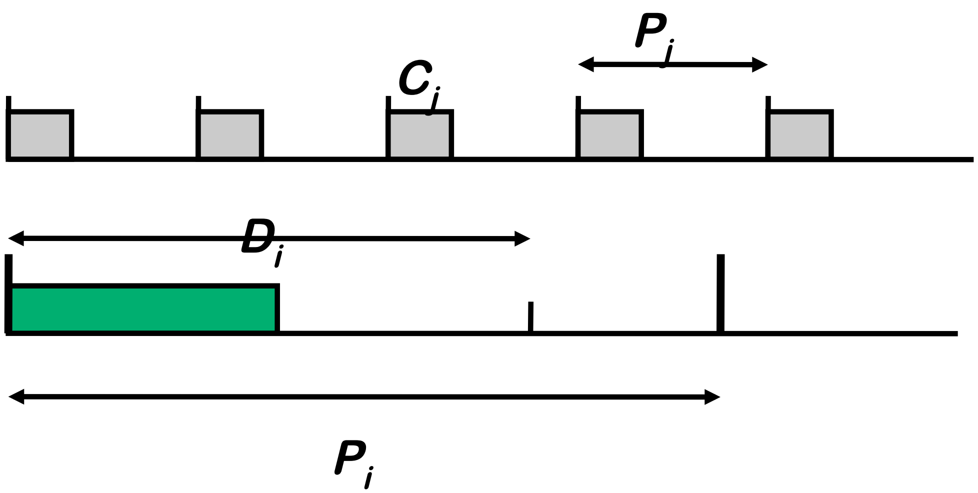
Time required by a higher priority task in an interval of length that corresponds to the relative deadline of task *i*.



$$I_j = \lceil \frac{D_i}{P_j} \rceil C_j$$

A better condition for schedulability

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

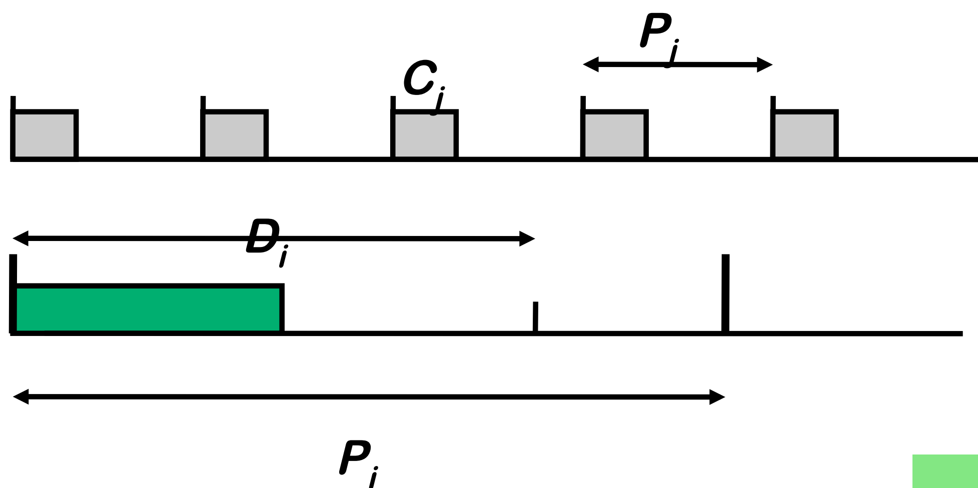


$$I_j = \lceil \frac{D_i}{P_j} \rceil C_j$$

$$\sum_j \lceil \frac{D_i}{P_j} \rceil C_j + C_i \leq D_i$$

A better condition for schedulability

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



$$I_j = \lceil \frac{D_i}{P_j} \rceil C_j$$

Interference from higher priority tasks

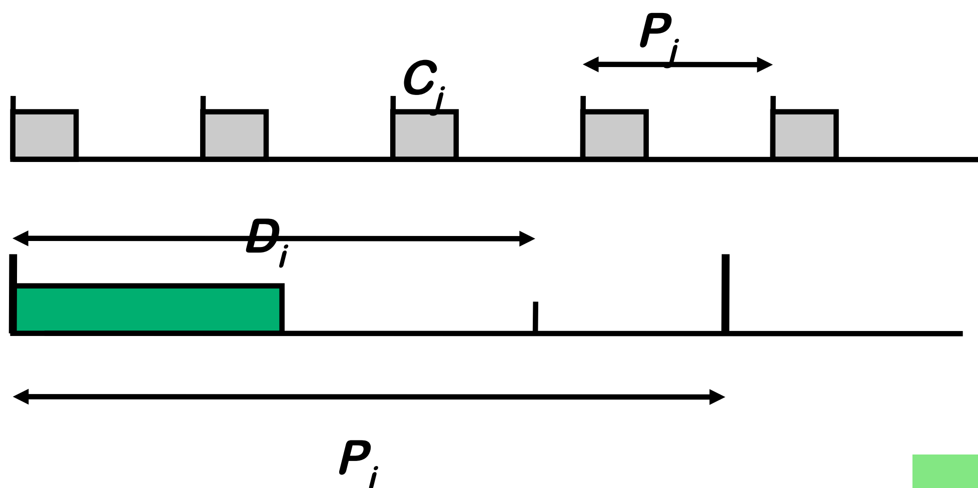
$$\boxed{\sum_j \lceil \frac{D_i}{P_j} \rceil C_j} + \boxed{C_i} \leq D_i$$

Execution time of T_i

A better condition for schedulability

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

There still is a problem!



$$I_j = \lceil \frac{D_i}{P_j} \rceil C_j$$

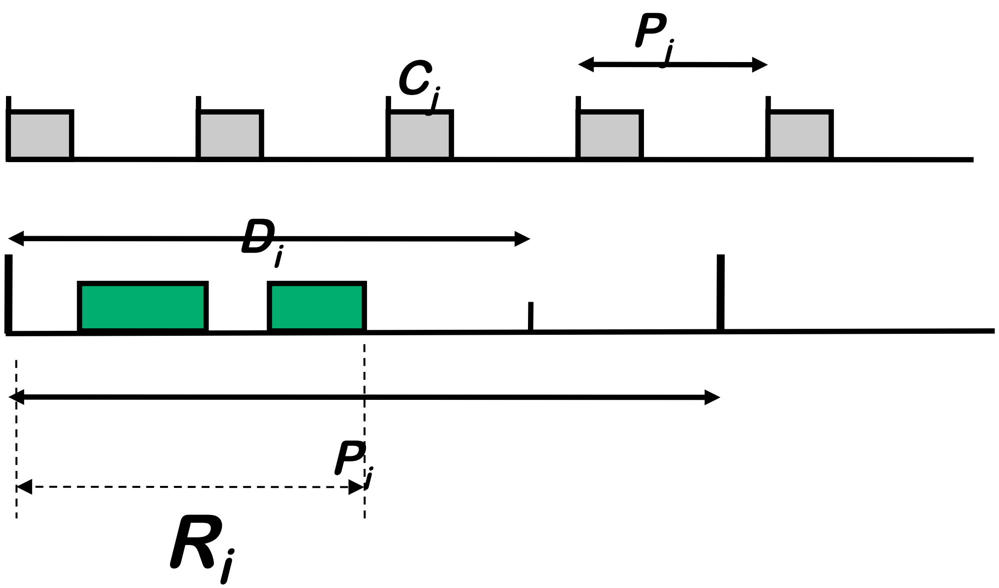
Interference from higher priority tasks

$$\underbrace{\sum_j \lceil \frac{D_i}{P_j} \rceil C_j}_{\text{Interference from higher priority tasks}} + \underbrace{C_i}_{\text{Execution time of } T_i} \leq D_i$$

Execution time of T_i

An exact condition for schedulability

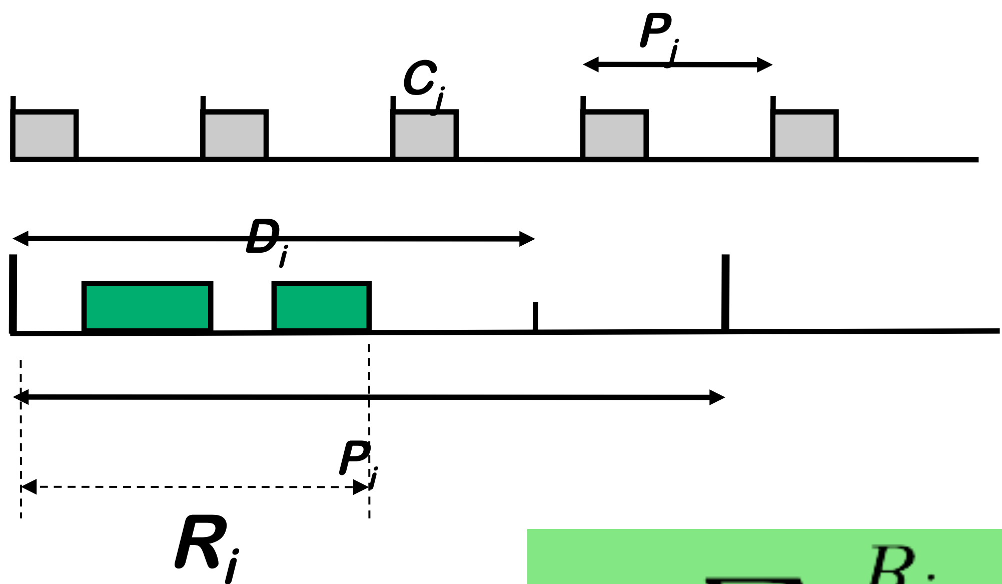
- Interference exists only till a job completes execution, i.e., up to the response time R_i
- Not necessarily up to the relative deadline D_i



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

An exact condition for schedulability

- Interference exists only till a job completes execution, i.e., up to the response time R_i
- Not necessarily up to the relative deadline D_i

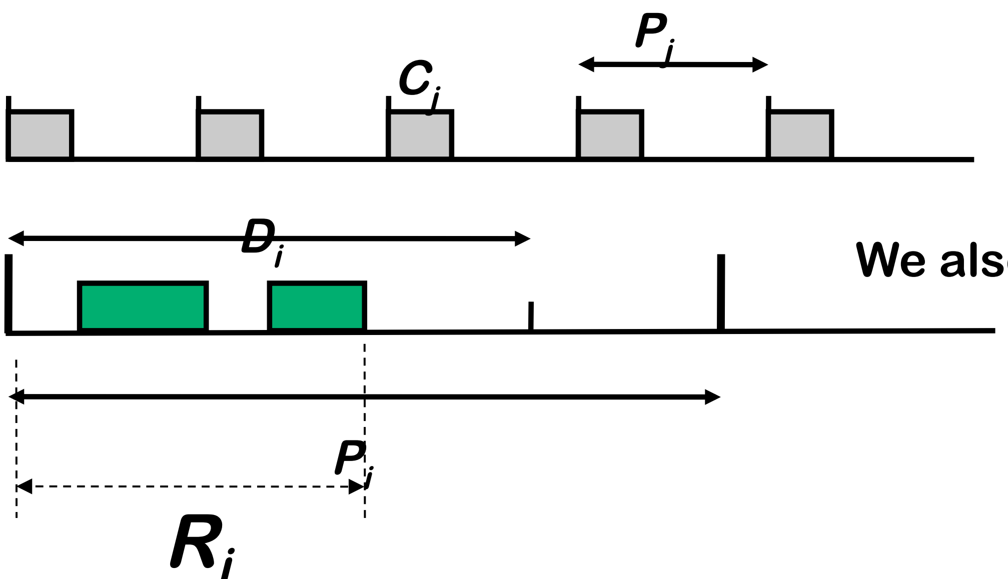


$$I = \sum_j \lceil \frac{R_i}{P_j} \rceil C_j$$

Interference from all higher priority tasks

An exact condition for schedulability

- Interference exists only till a job completes execution, i.e., up to the response time R_i
- Not necessarily up to the relative deadline D_i



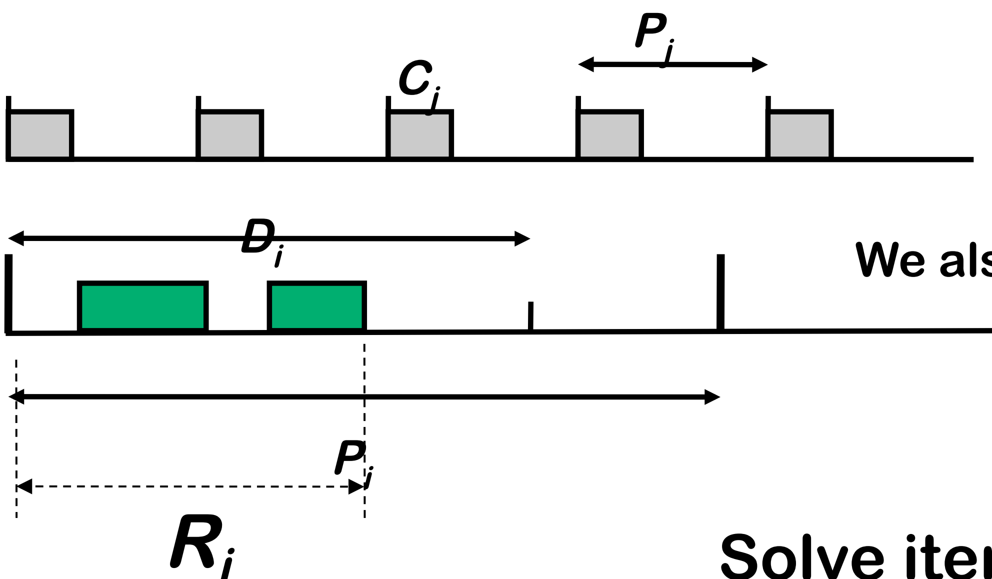
We also have the following relation:

$$R_i = I + C_i$$

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

An exact condition for schedulability

- Interference exists only till a job completes execution, i.e., up to the response time R_i
- Not necessarily up to the relative deadline D_i



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

We also have the following relation:

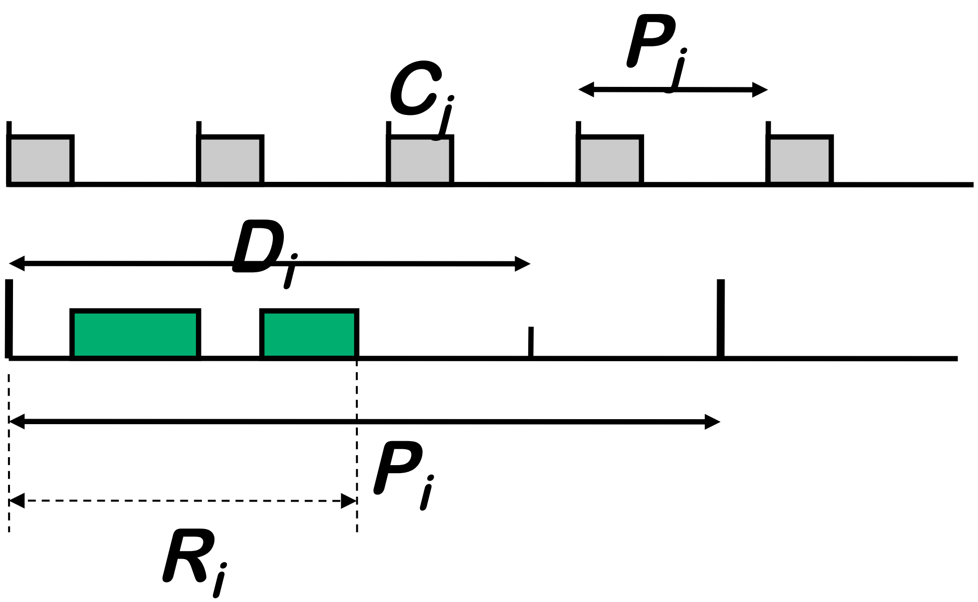
$$R_i = I + C_i$$

Solve iteratively for the smallest R_i to satisfy both relations

Example

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

$$R_i = I + C_i$$



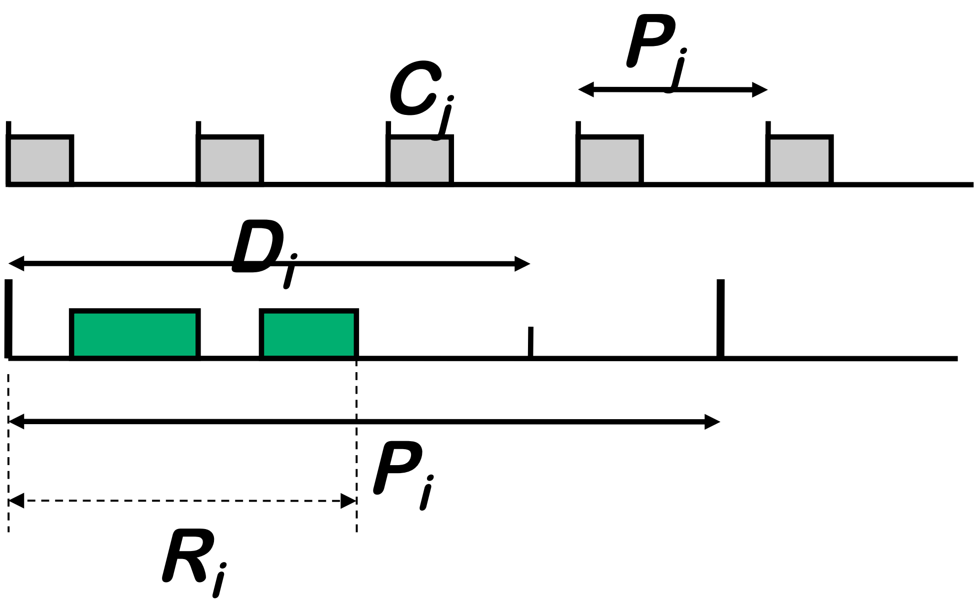
Consider a system of two tasks:

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

Example

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

$$R_i = I + C_i$$



$$R_2^{(0)} = C_2 = 2$$
$$I^{(0)} = \lceil 2/1.7 \rceil (0.5) = 1$$

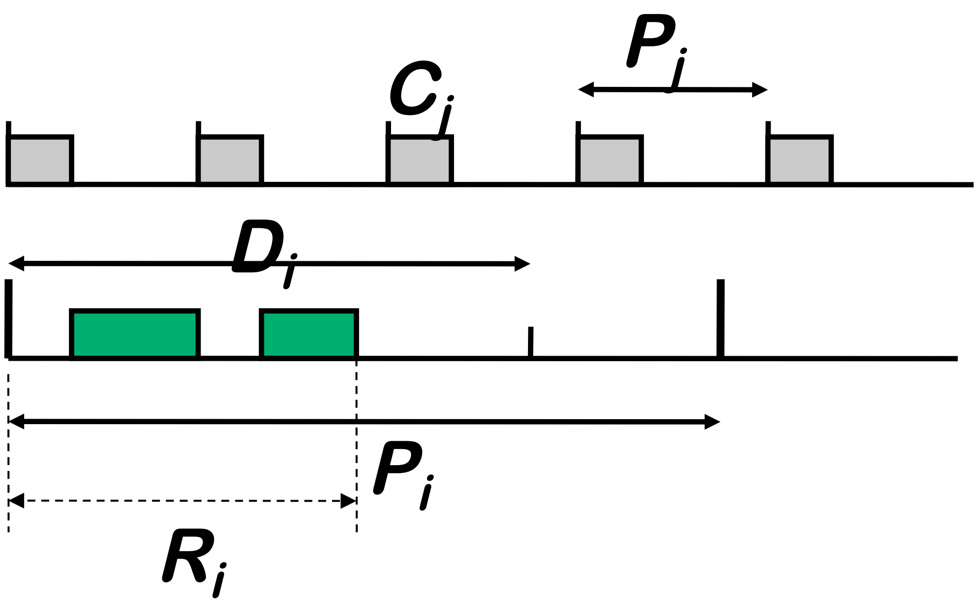
Consider a system of two tasks:

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

Example

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

$$R_i = I + C_i$$



$$R_2^{(0)} = C_2 = 2$$
$$I^{(0)} = \lceil 2/1.7 \rceil (0.5) = 1$$

$$R_2^{(1)} = I_2^{(0)} + C_2 = 3$$
$$I^{(1)} = \lceil 3/1.7 \rceil (0.5) = 1$$

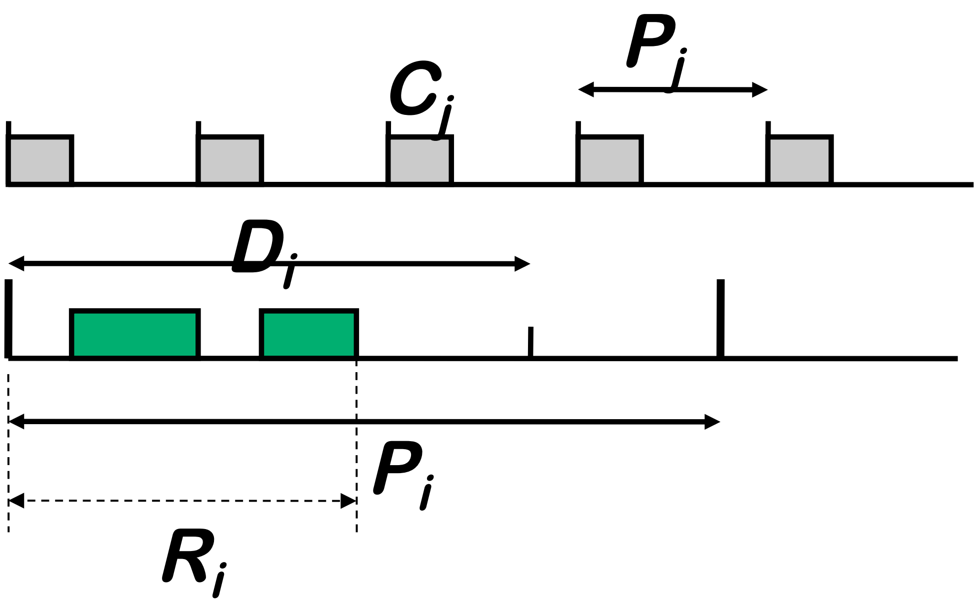
Consider a system of two tasks:

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

Example

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

$$R_i = I + C_i$$



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

$$R_2^{(0)} = C_2 = 2$$
$$I^{(0)} = \lceil 2/1.7 \rceil (0.5) = 1$$

$$R_2^{(1)} = I_2^{(0)} + C_2 = 3$$
$$I^{(1)} = \lceil 3/1.7 \rceil (0.5) = 1$$

$$R_2^{(2)} = I^{(1)} + C_2 = 3$$
$$R_2^{(2)} = R_2^{(1)}$$

3 < 3.2; Task 2 is schedulable.

Lecture summary

- There are better utilization bounds than the Liu & Layland utilization bound: the hyperbolic bound
- When the relative deadline of a task is less than its period, we can apply utilization bounds
  - But such tests are even more pessimistic than normal
- We can apply exact tests for schedulability when deadlines are less than or equal to periods
  - Such tests require more computation
  - Iterative process

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

$$R_i = I + C_i$$