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 $\prod_i(1 + U_i)$ for a schedulable system?
- **Motivation:** There may be other functions of a task set rather then just utilization that also indicate schedulability.

Hyperbolic bound for RM

worst case conditions
for schedulability
of **2** tasks under RM

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

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

Hyperbolic bound for rate monotonic scheduling

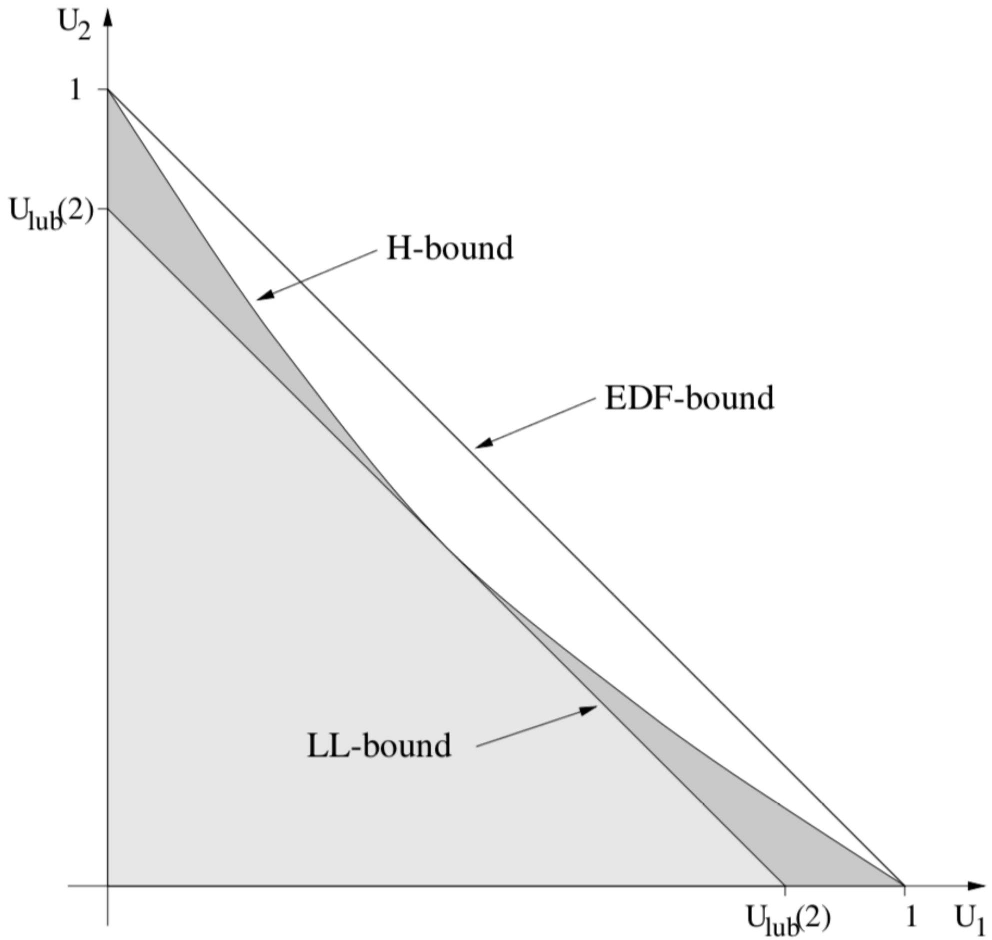
- A set of periodic tasks is schedulable if

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

- 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)

- **Question:** What happens to the hyperbolic bound if task utilizations are equal?

Feasibility regions in U-space



Hyperbolic bound is *tight*

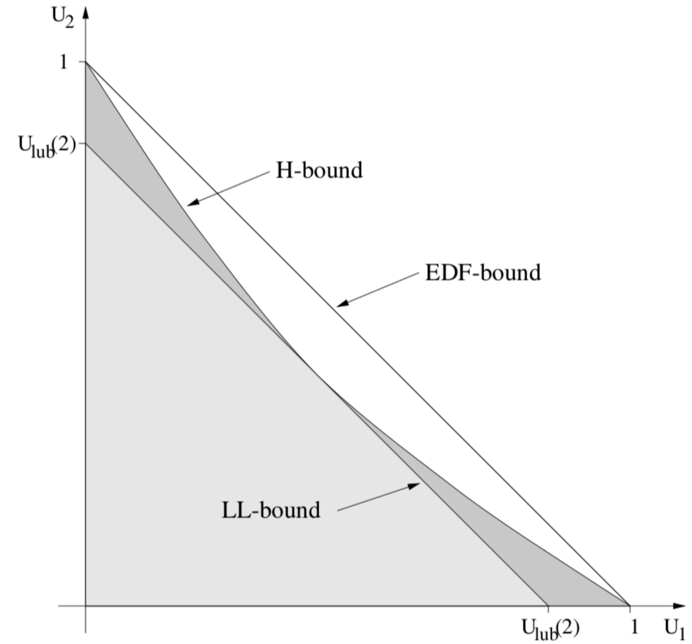
- It is the best possible bound with only knowledge of utilization factors and the number of jobs

- A utilization-based condition $C(u_1, \dots, u_n)$ for a scheduling algorithm is ***tight*** if for every utilization set $(u_1, \dots, u_n)$ with $0 \le u_i \le 1$ for which $C(u_1, \dots, u_n)$ does ***not*** hold, there exists a task set $T_1, \dots, T_n$ with utilizations $u_1, \dots, u_n$ that is ***not*** schedulable by the scheduling algorithm
  - We can construct a task set with the prescribed utilizations (which violate the schedulability condition) that is ***infeasible*** under the given algorithm

- Tightness was proved for H-bound → With the algorithm being RM and $C(u_1, \dots, u_n) \equiv \prod_{i=1}^{n}(1 + u_i) \le 2$

- **Q:** Is the LL bound tight?

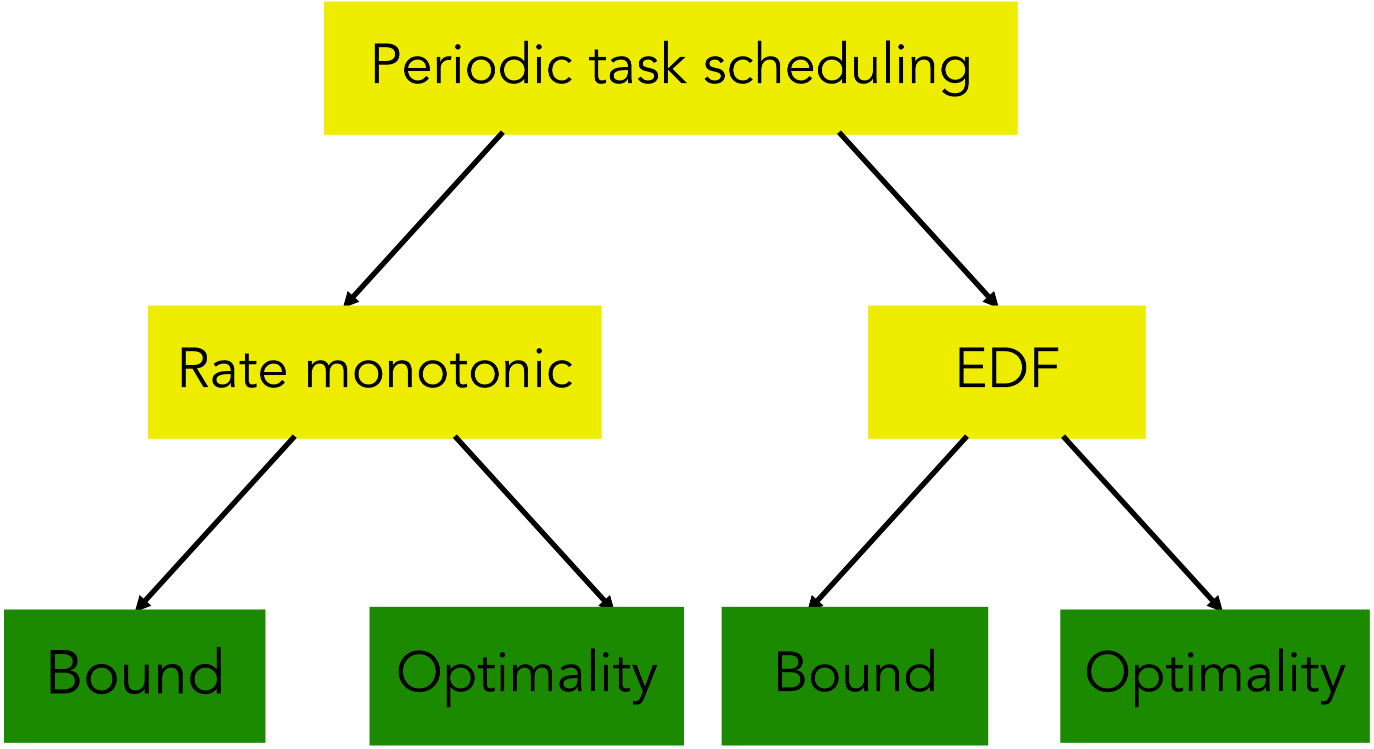
How much better is Hyperbolic bound relative LL?



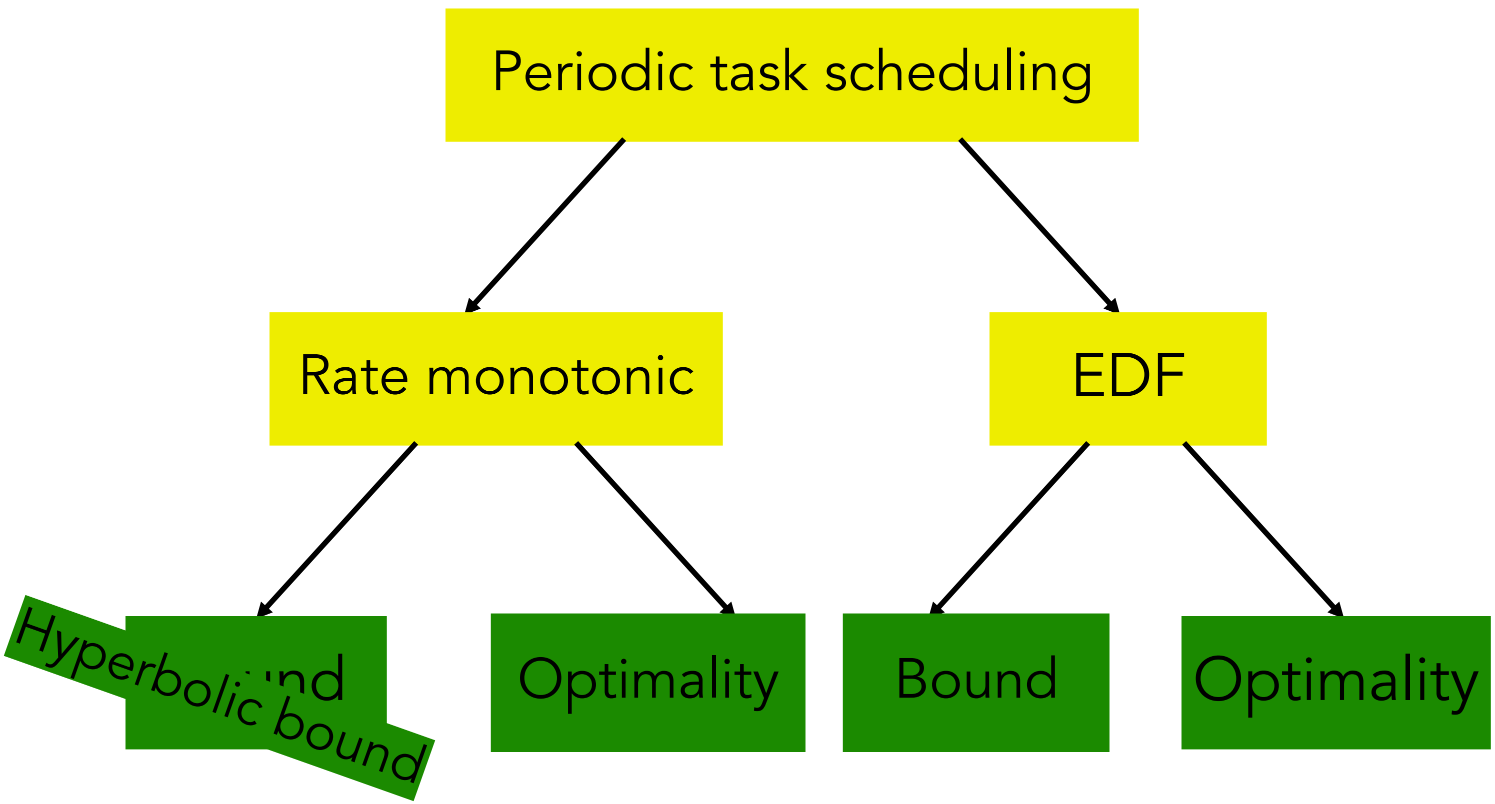
- **How do we measure the gain of H-bound over LL-bound?**
- Consider the utilization space
  - **U-space:** Subset of n-dimensional Euclidean space consisting of vectors $(u_1, \dots, u_n) \in [0,1]^n$
- Fix number of jobs n
- Volume $\text{vol}^n(A)$: n-dimensional Lebesgue measure of (measurable) set $A \subset \mathbb{R}^n$
- Take volume of H-bound region
  - Here need to find $\text{vol}^n(H)$, $H = \{u \in \mathbb{R}^n\colon\ u_i \in [0, 1], \prod_{i=1}^{n}(1 + u_i) \leq 2\}$
- Take volume of LL-bound region
  - need to find $\text{vol}^n(LL)$, $LL = \left\{u \in \mathbb{R}^n : u_i \in [0,1], \sum_{i=1}^{n} u_i \leq n\left(2^{1/n} - 1\right)\right\}$

$$\text{Asymptotic Gain: } \rho_n = \frac{\text{vol}^n(H)}{\text{vol}^n(LL)} = \sqrt{2} + O(n^{-1})$$

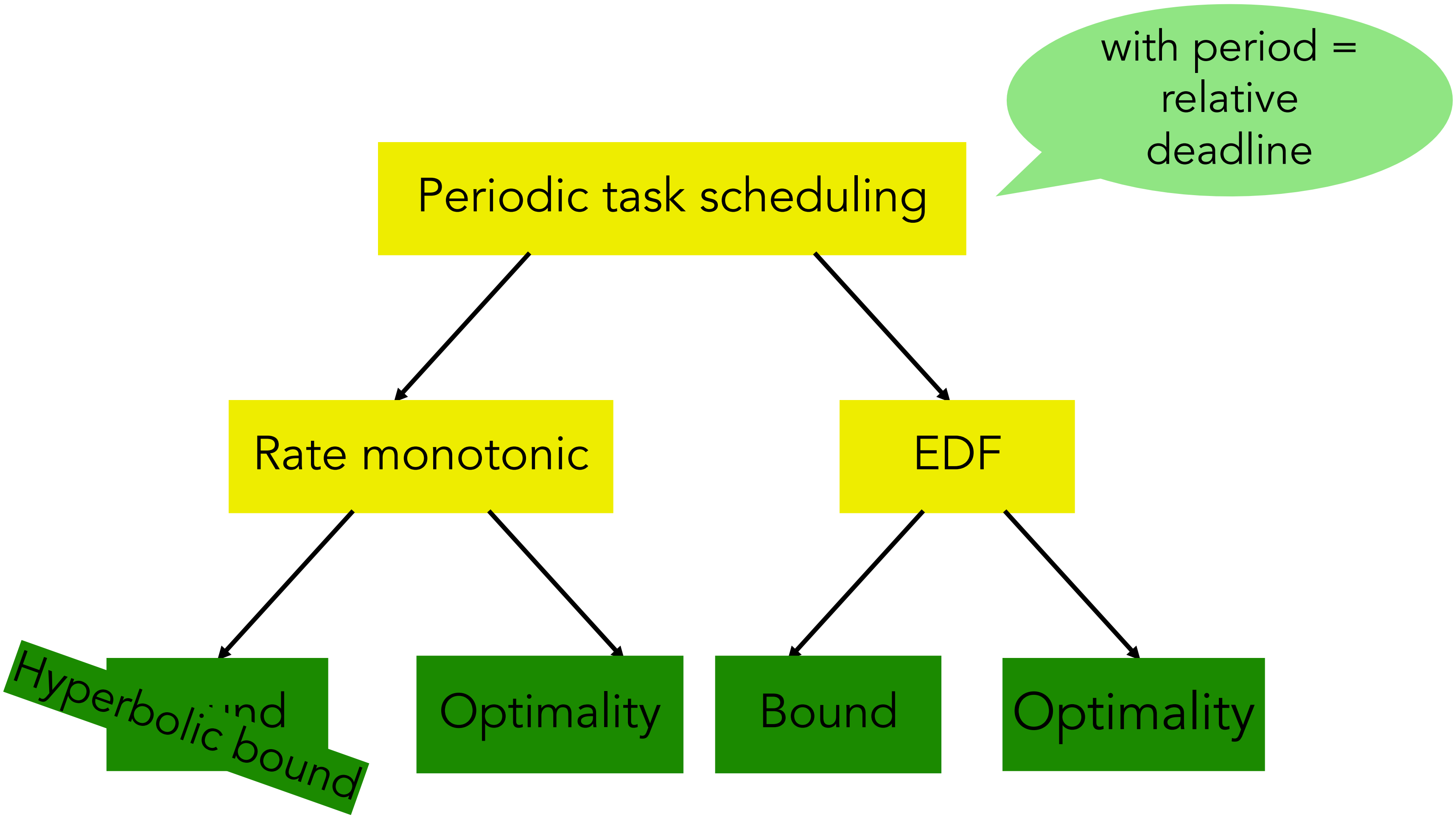
Scheduling taxonomy



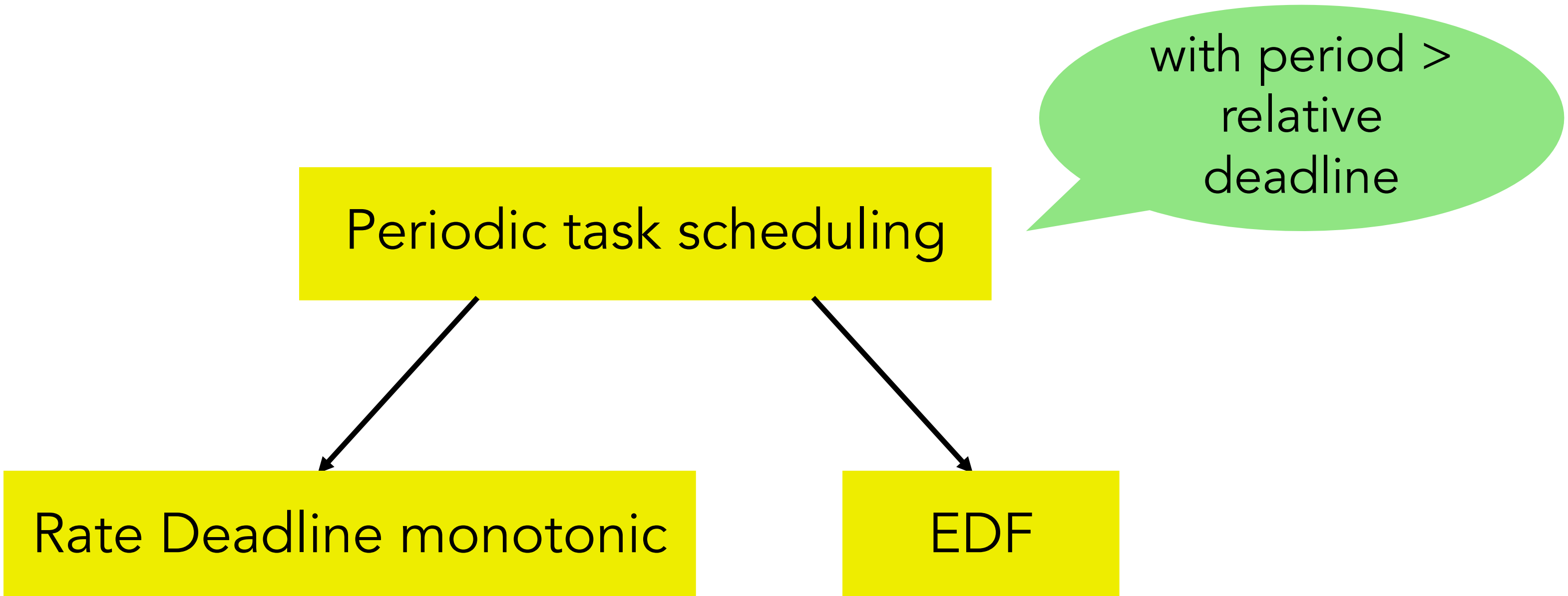
Scheduling taxonomy



Scheduling taxonomy

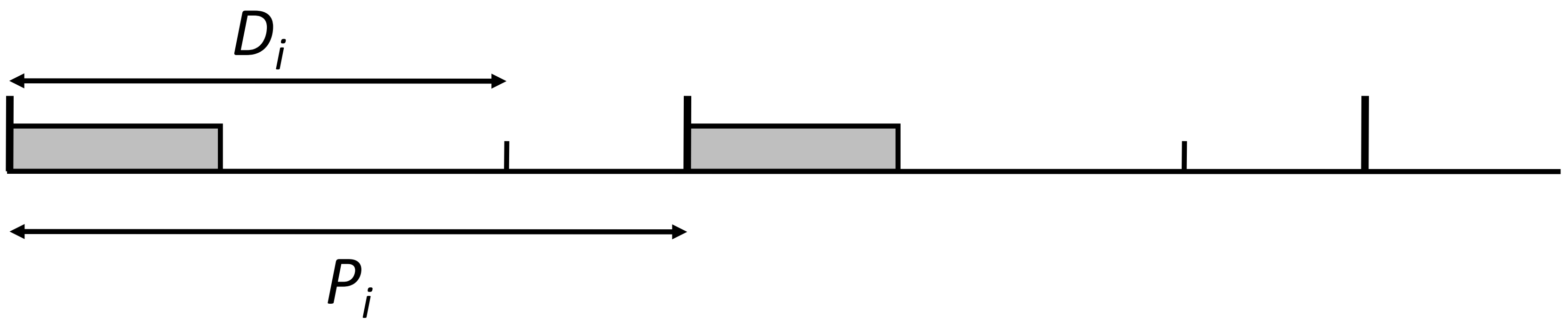


Scheduling taxonomy



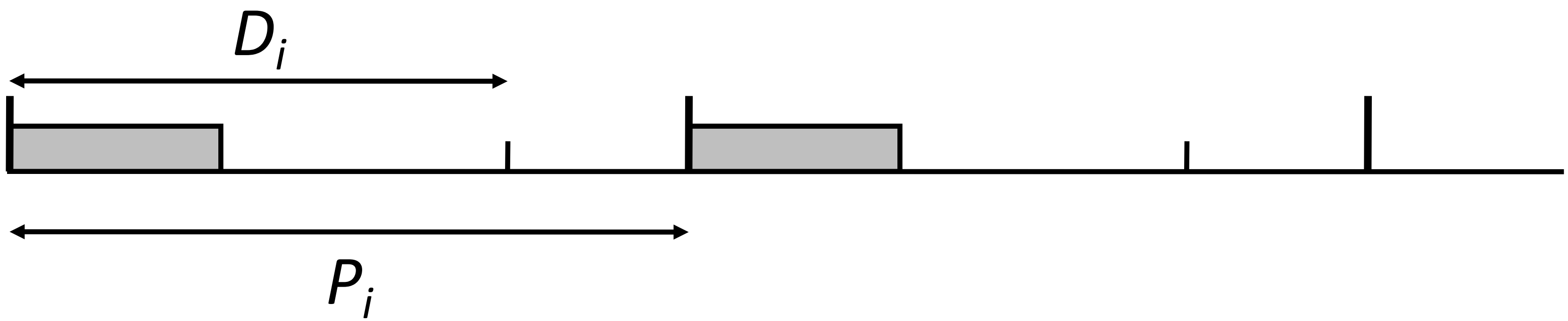
Deadline monotonic scheduling

- 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 $D_i < P_i$.

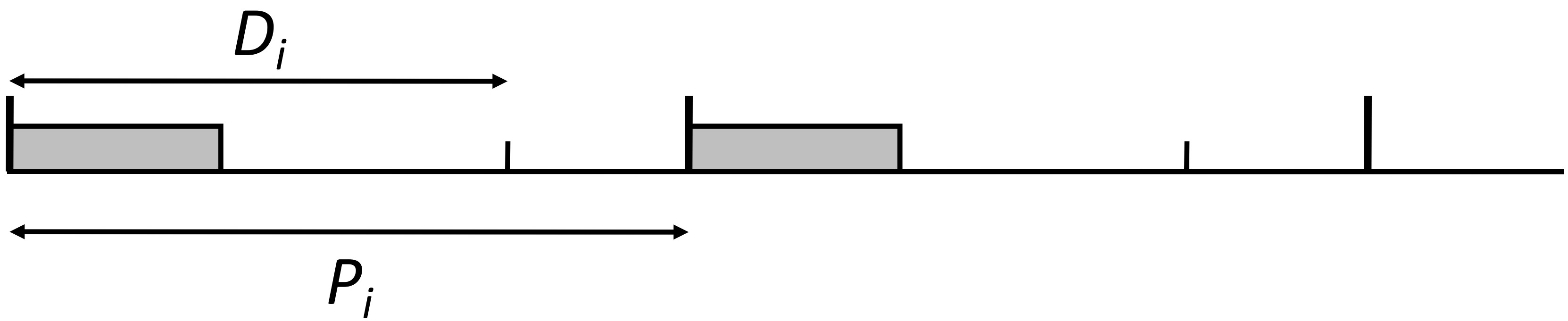


- What is the schedulability condition?

- Can not be worse than when the period of each task is reduced 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 $D_i < P_i$.

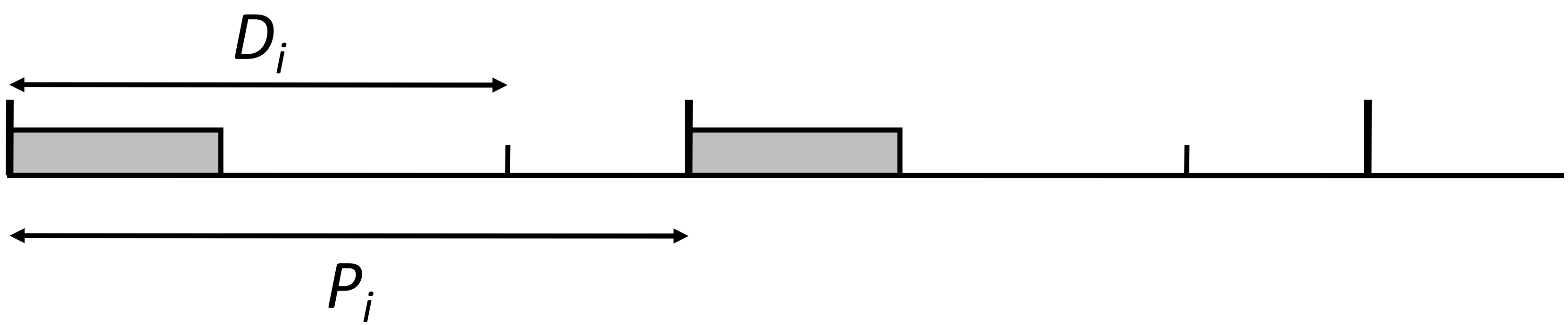


- What is the schedulability condition?
- Can not be worse than when the period of each task is reduced to D_i.

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

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 $D_i < P_i$.



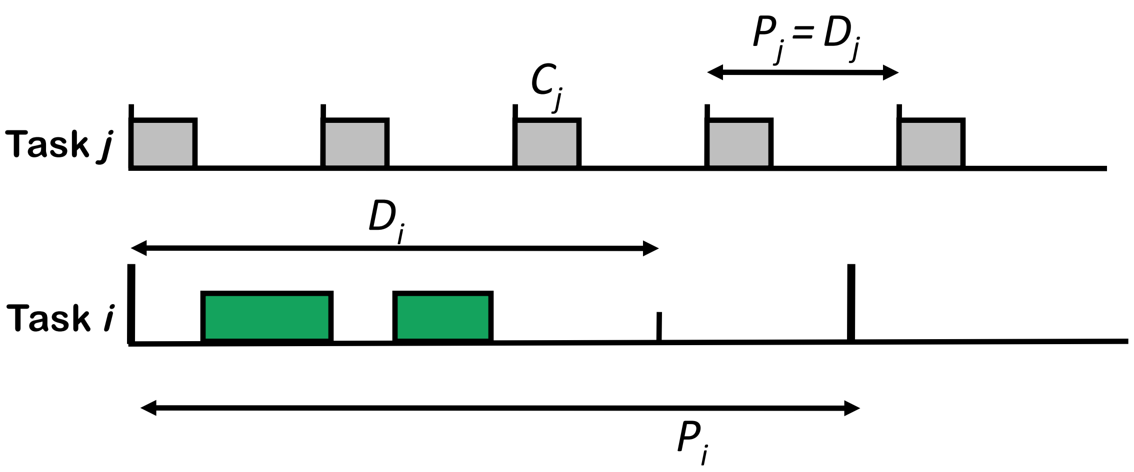
- What is the schedulability condition?
- Can not be worse than when the period of each task is reduced to D_i.

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

What is the problem?

A better condition for schedulability

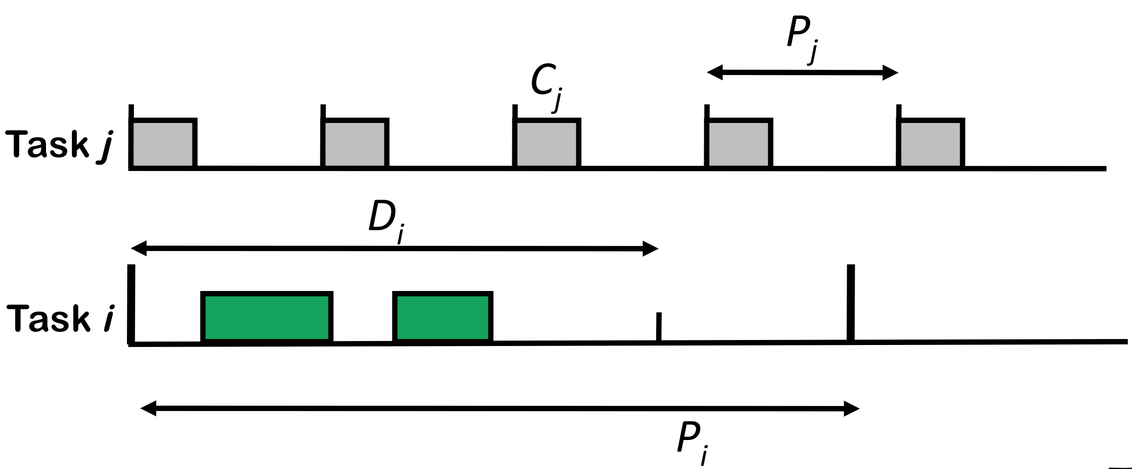
- Worst case interference from a higher priority task, *j*?



A better condition for schedulability

- Worst case <u>interference</u> 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*.



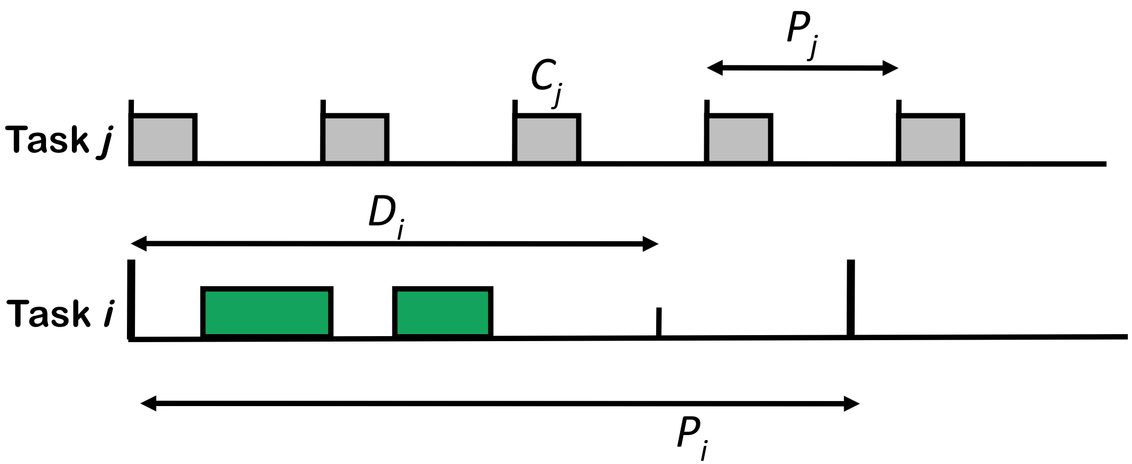
$$I_i(j) = \underbrace{\left\lceil \frac{D_i}{P_j} \right\rceil}_{\text{Number of execution requests of task } j \text{ in duration of length } D_i \text{ assuming critical instant}} C_j$$

Worst case Interference task *j* exercises on task *i*
→ upper bound on total workload *requested* by task *j* during D_i at the critical instant

Number of execution *requests of task j in duration of length D_i assuming critical instant*

A better condition for schedulability

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

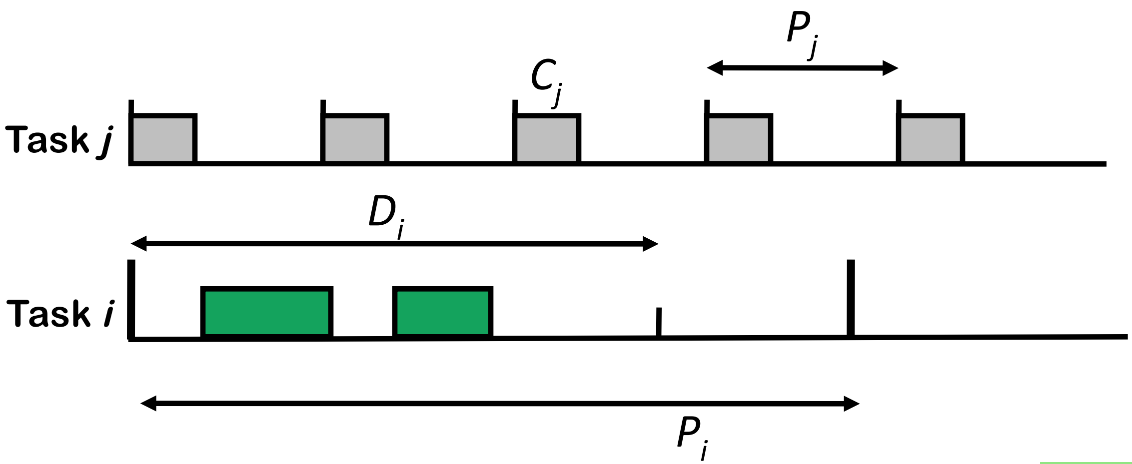


$$I_i(j) = \left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

$$\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{D_i}{P_j} \right\rceil C_j + C_i \leqslant D_i$$

A better condition for schedulability

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



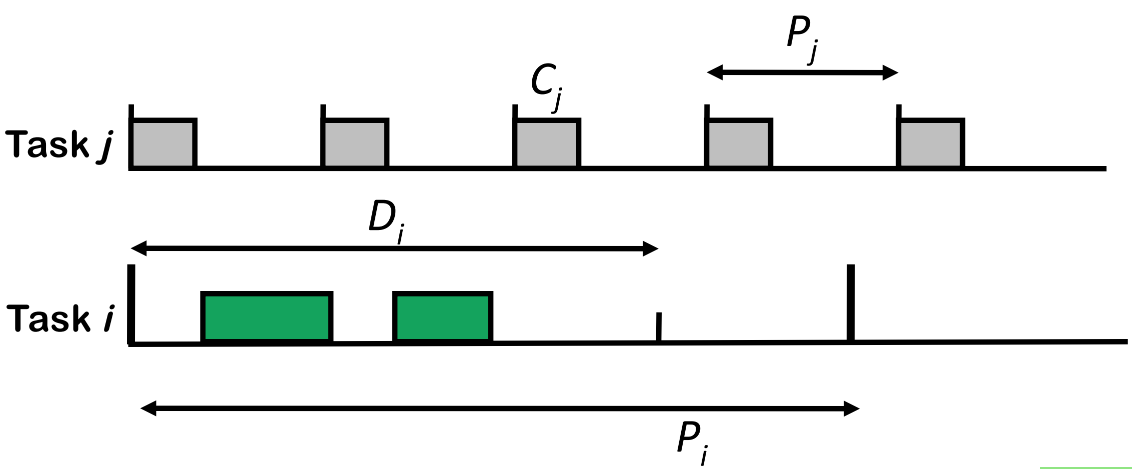
$$I_i(j) = \left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

Interference from higher priority tasks → $\boxed{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{D_i}{P_j} \right\rceil C_j} + \boxed{C_i} \leqslant D_i$

Execution time of T_i

A better condition for schedulability

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



There still is a problem!

$$I_i(j) = \left\lceil \frac{D_i}{P_j} \right\rceil C_j$$

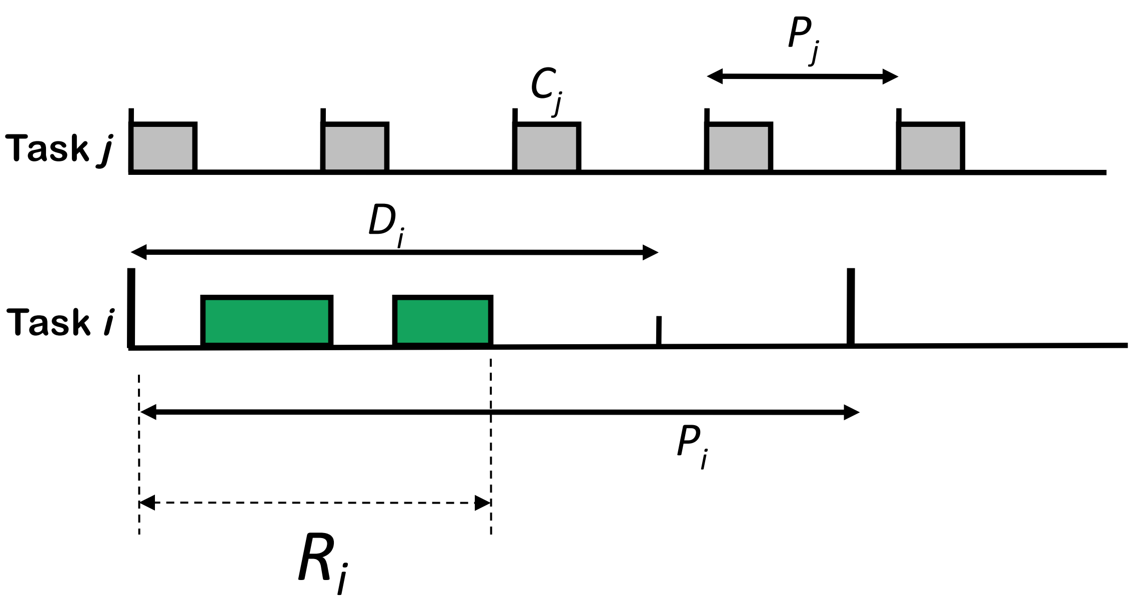
Interference from higher priority tasks →

$$\boxed{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{D_i}{P_j} \right\rceil C_j} + \boxed{C_i} \leqslant D_i$$

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



$$I_i(j) = \left\lceil \frac{\overset{R_i}{\cancel{D_i}}}{P_j} \right\rceil C_j$$

An exact condition for schedulability

(1) $\left\lceil \frac{R_i}{P_j} \right\rceil C_j$: the *exact* workload interfering with task i
in an interval of length R_i starting at the latest critical instant

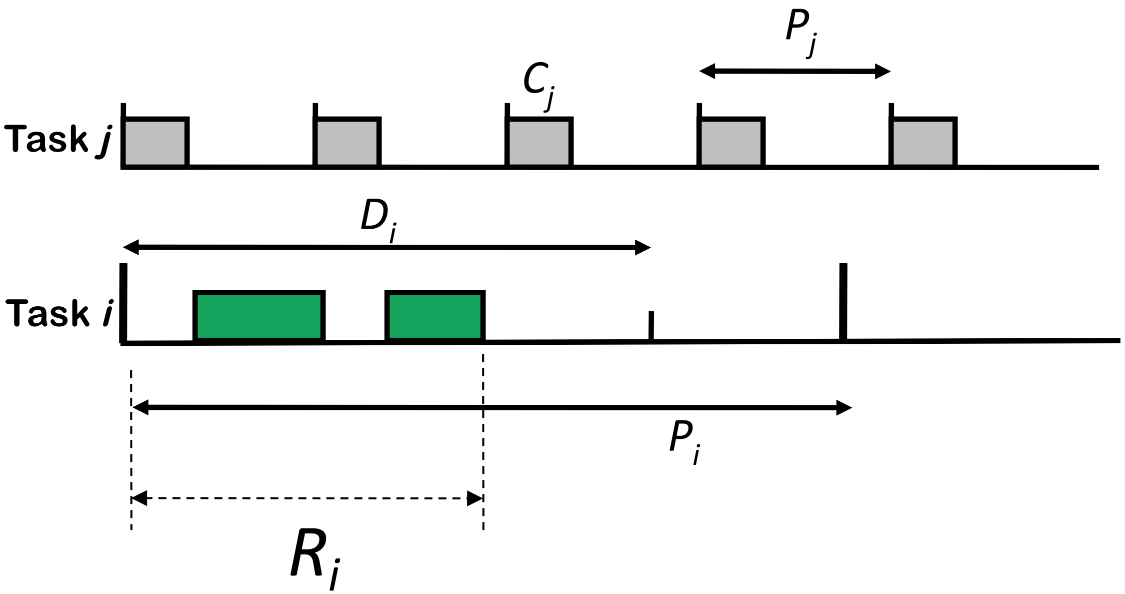
But $\left\lceil \frac{R_i}{P_j} \right\rceil C_j$ is the workload *requested* by higher priority task τ_j
in an interval of length R_i starting at the latest critical instant

And (1) is saying that this workload indeed *completes* by R_i. Why?

Because task j has higher priority than task i so all instances of task j
that arrive in interval of length R_i finish before task 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

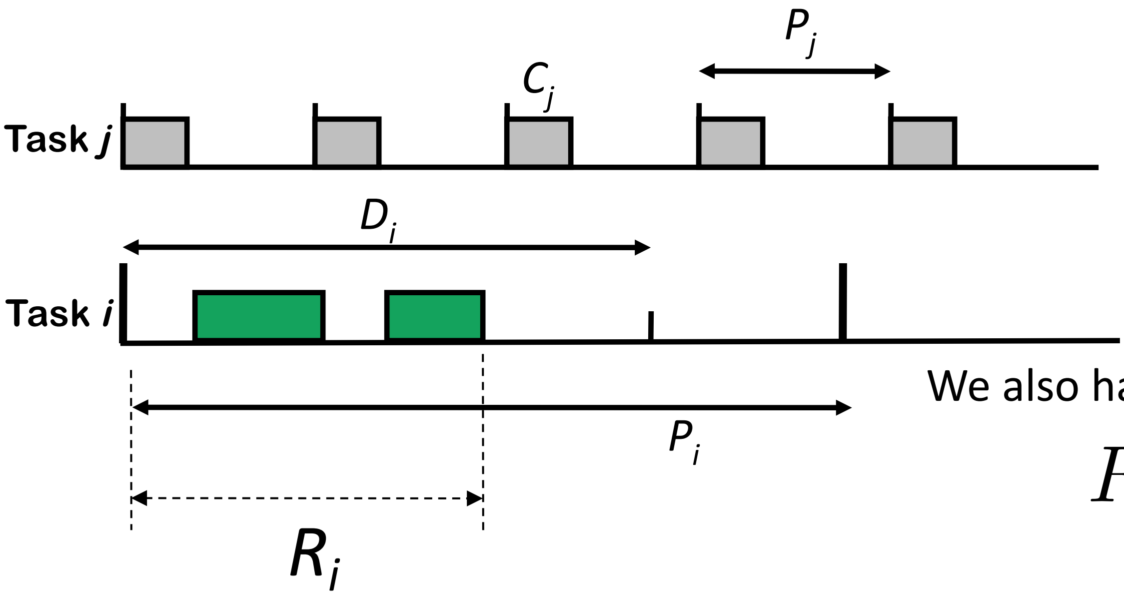


$$I_i = \sum_{j \in \mathrm{hp}(i)} I_i(j) = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

Interference on task *i* 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



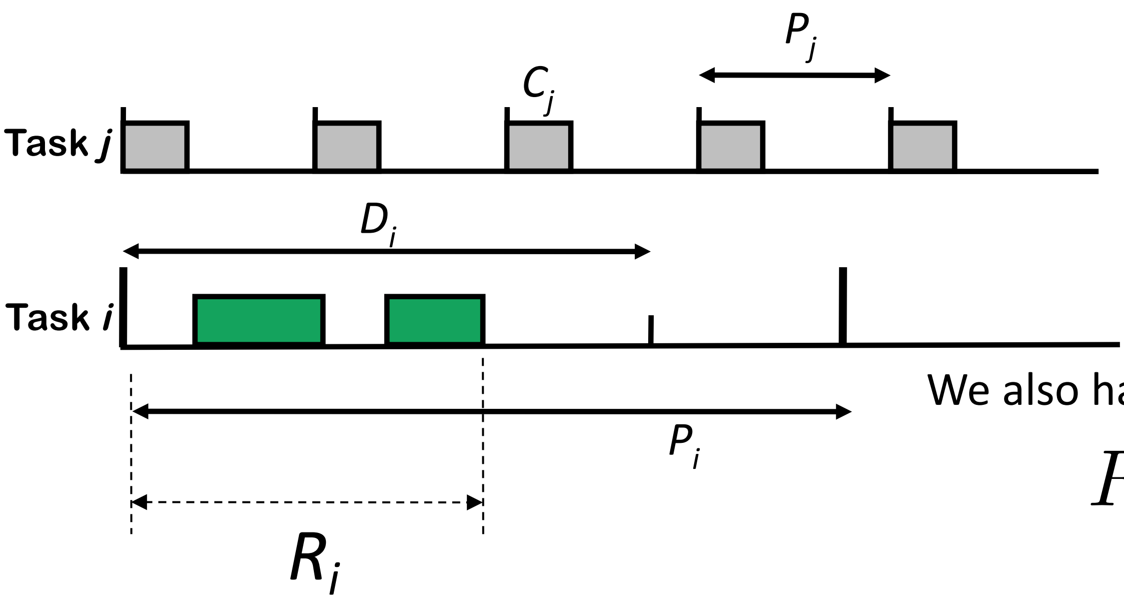
We also have the following relation:

$$R_i = I_i + C_i$$

$$I_i = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

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



We also have the following relation:

$$R_i = I_i + C_i$$

Solve iteratively for the smallest R_i to satisfy both relations

$$I_i = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

Computing Response Time

variable

$$R_i = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + C_i$$

- **What is a solution to this recurrence?**

Computing Response Time

variable

$$R_i = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + C_i$$

- **What is a solution to this recurrence?**
  - The smallest $t > 0$ such that $t = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{t}{P_j} \right\rceil C_j + C_i$ → called *fixed-point*

Computing Response Time

variable

$$R_i = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + C_i$$

- **What is a solution to this recurrence?**
  - The smallest $t > 0$ such that $t = \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{t}{P_j} \right\rceil C_j + C_i$ → called *fixed-point*
- Does a solution always exist?
- If so, how can a solution be computed in a finite number of steps? (convergence)

Finding Response Times in a Real-Time System M. Joseph P. Pandya.
***The Computer Journal*, Volume 29, Issue 5, 1 January 1986, Pages 390–395.**

Computing Response Time

$$\text{Recurrence: } R_i^{(n+1)} = \underbrace{\sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i^{(n)}}{P_j} \right\rceil C_j}_{I_i^{(n)}} + C_i$$

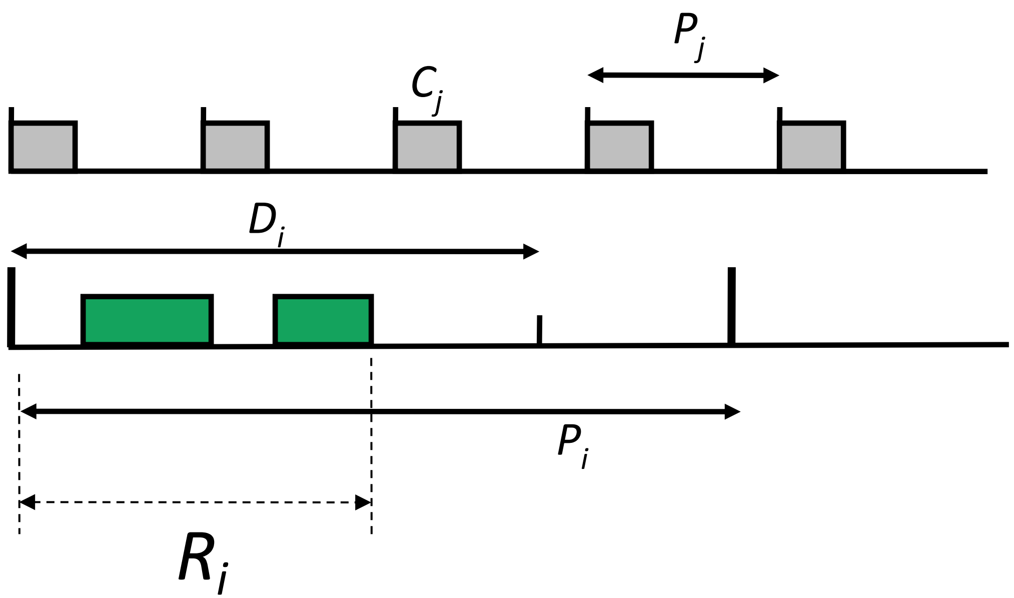
- What is a proper initial value $R_i^{(0)}$?
  - Any *lower* bound on the response time → take $R_i^{(0)} = C_i$ or $R_i^{(0)} = \sum_{j \in \text{hp}(i)} C_j$
  - Affects the rate of convergence

Exact Response Time: Solution Existence

- It was shown that recurrence converges if $\sum_{j \in \mathrm{hp}(i)} u_j \leq 1$
- Easy to see that $R_i^{(n+1)} \geq R_i^{(n)}$ $\rightarrow$
  - Induction on n + reason about $\left(R_i^{n+1} - R_i^{n}\right)$ + use fact that $x \mapsto \lceil x \rceil$ is increasing
- Stop at first n for which $R_i^{(n+1)} = R_i^{(n)}$
- Recurrence might not converge if $\sum_{j \in \mathrm{hp}(i)} u_j > 1$
  - If only want to know whether or not taskset is schedulable $\rightarrow$ Terminate as soon as $R_i^{(n)} > D_i$ or $R_i^{(n)} > P_i$

Example

$$R_i^{(n+1)} = \underbrace{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(n)}}{P_j} \right\rceil C_j}_{I_i^{(n)}} + C_i$$



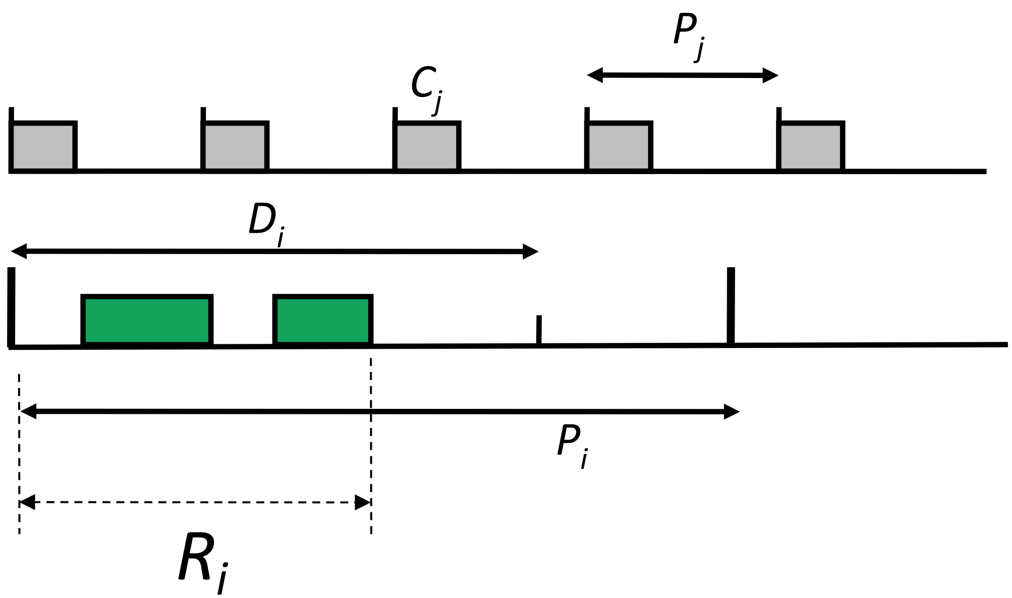
Consider a system of two tasks:

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

$$R_i^{(n+1)} = \underbrace{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(n)}}{P_j} \right\rceil C_j}_{I_i^{(n)}} + C_i$$

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

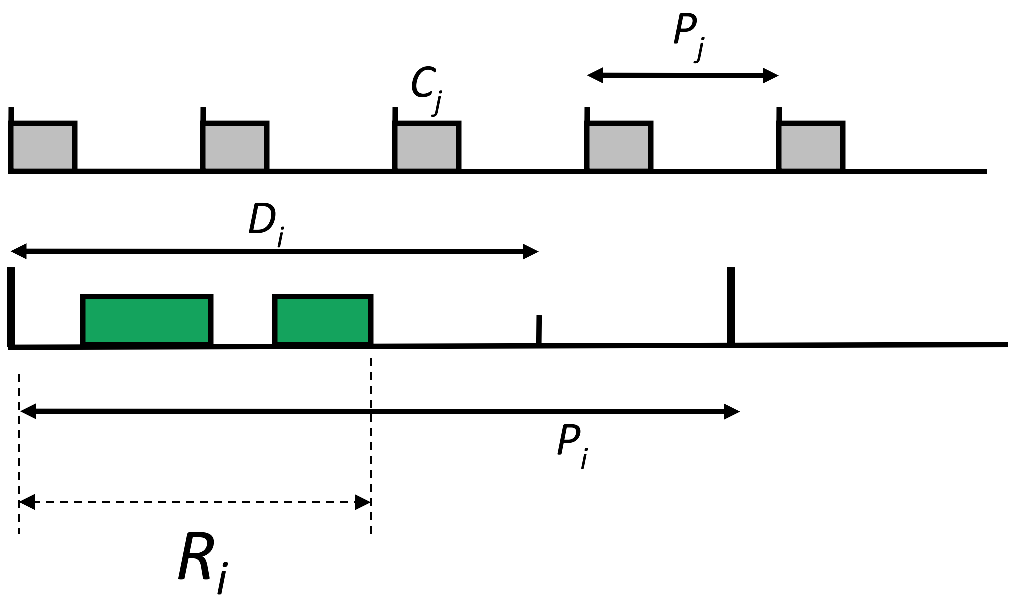


Consider a system of two tasks:

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

$$R_i^{(n+1)} = \underbrace{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(n)}}{P_j} \right\rceil C_j}_{I_i^{(n)}} + C_i$$



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

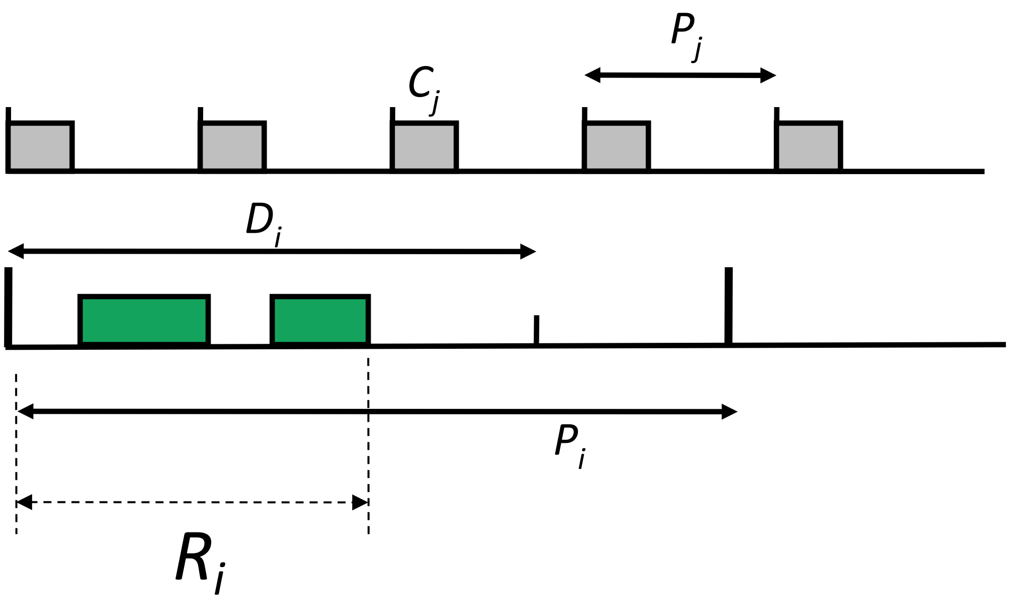
$$R_2^{(1)} = I_2^{(0)} + C_2 = 3$$
$$I^{(1)} = \lceil 3/1.7 \rceil (0.5) = 1$$

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

$$R_i^{(n+1)} = \underbrace{\sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^{(n)}}{P_j} \right\rceil C_j}_{I_i^{(n)}} + C_i$$



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

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

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

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

3 < 3.2; Task 2 is schedulable.

RTA Algorithm

DM_guarantee (Γ) {
  for (each $\tau_i \in \Gamma$) {
    $I_i = \sum_{k=1}^{i-1} C_k$;
    do {
      $R_i = I_i + C_i$;
      if $(R_i > D_i)$ return(UNSCHEDULABLE);
      $I_i = \sum_{k=1}^{i-1} \left\lceil \frac{R_i}{T_k} \right\rceil C_k$;
    } **while** $(I_i + C_i > R_i)$;
  }
  return(SCHEDULABLE);
}

Response Time Analysis: Complexity

- Is this test efficient?
- Assuming all instance parameters are integers:
  - Inner loop adds at most 1 to interference until deadline is reached
- Runs in time $O(nP_{\max})$, where $P_{\max}$ is the largest period
  - test runs in **pseudo-polynomial** time, not efficient as periods become larger
  - Not suitable for online admission control

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