2IMN20 - Real-Time Systems

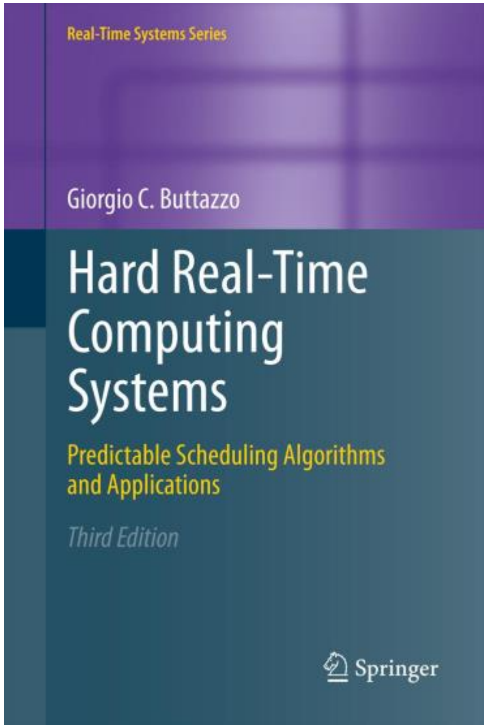
Response-time analysis for fixed-priority preemptive scheduling

Geoffrey Nelissen

2023-2024

Reference book

Chapter 4



Disclaimer:
Many slides were provided by Dr. Mitra Nasri

Some slides have been taken from Giorgio Buttazzo's course

Agenda

- RM utilization-based tests (reminder)
- ***Response-time analysis*** for periodic or sporadic tasks under preemptive fixed priority scheduling

Assumptions

- For this lecture, we assume

  A1. All jobs of τ_i execute for no more than C_i

  A2. Tasks are periodic or sporadic

  A3. Tasks are fully preemptive

  A4. Context switch, preemption, and scheduling overheads are zero

  A5. Tasks are independent:
  - no precedence relations
  - no resource constraints
  - no blocking on I/O operations

  A6. No self-suspensions

  A7. Single core

  A8. Unknown phasing/offset for the tasks

Assume that tasks are indexed according to their priority ordering, namely, $P_1 > P_2 > P_3 > \cdots > P_n$

Agenda

- RM utilization-based tests (reminder)

- ***Response-time analysis*** for periodic or sporadic tasks under preemptive fixed priority scheduling

Building a utilization-based test for RM

Task utilization

$$U_i = \frac{C_i}{T_i}$$

Task set utilization

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Finding a utilization threshold (lower bound) such that **ANY task set** with **utilization lower than that bound** is **CERTAINLY schedulable by RM**

RM utilization-based tests

Assumptions: no release jitter (i.e., $\sigma_i = 0$) and tasks have implicit deadlines (i.e, $D_i = T_i$)

Liu and Layland bound: $$\sum_{i=1}^{n} U_i \leq n\left(2^{1/n} - 1\right)$$

Hyperbolic bound: $$\prod_{i=1}^{n} (U_i + 1) \leq 2$$

QUIZ TIME

QUIZ TIME

Quiz

- Is this task set schedulable by RM according to the Liu and Layland test?

- Yes
- No

τ_i	C_i	T_i	D_i	U_i
τ_1	1.5	5	5	0.3
τ_2	4	10	10	0.4

U = 0.7

$$\sum_{i=1}^{n} U_i \leq n\left(2^{1/n} - 1\right)$$

Quiz

- If the hyperbolic bound test accepts a task set, Liu and Layland's test also accepts that task set.

- true
- false

Quiz

- Is this task set schedulable by RM according to the hyperbolic bound test?

- Yes
- No

τ_i	C_i	T_i	D_i	U_i
τ_1	1.5	5	5	0.3
τ_2	4	10	10	0.4

U = 0.7

Quiz

- The hyperbolic bound test is a necessary schedulability test.

- true
- false

Quiz

- If the Liu and Layland test accepts a task set, then the hyperbolic bound test also accepts that task set.

- true
- false

Quiz

- Both the hyperbolic bound and Liu and Layland tests are sufficient schedulability tests.

- true
- false


QUIZ TIME

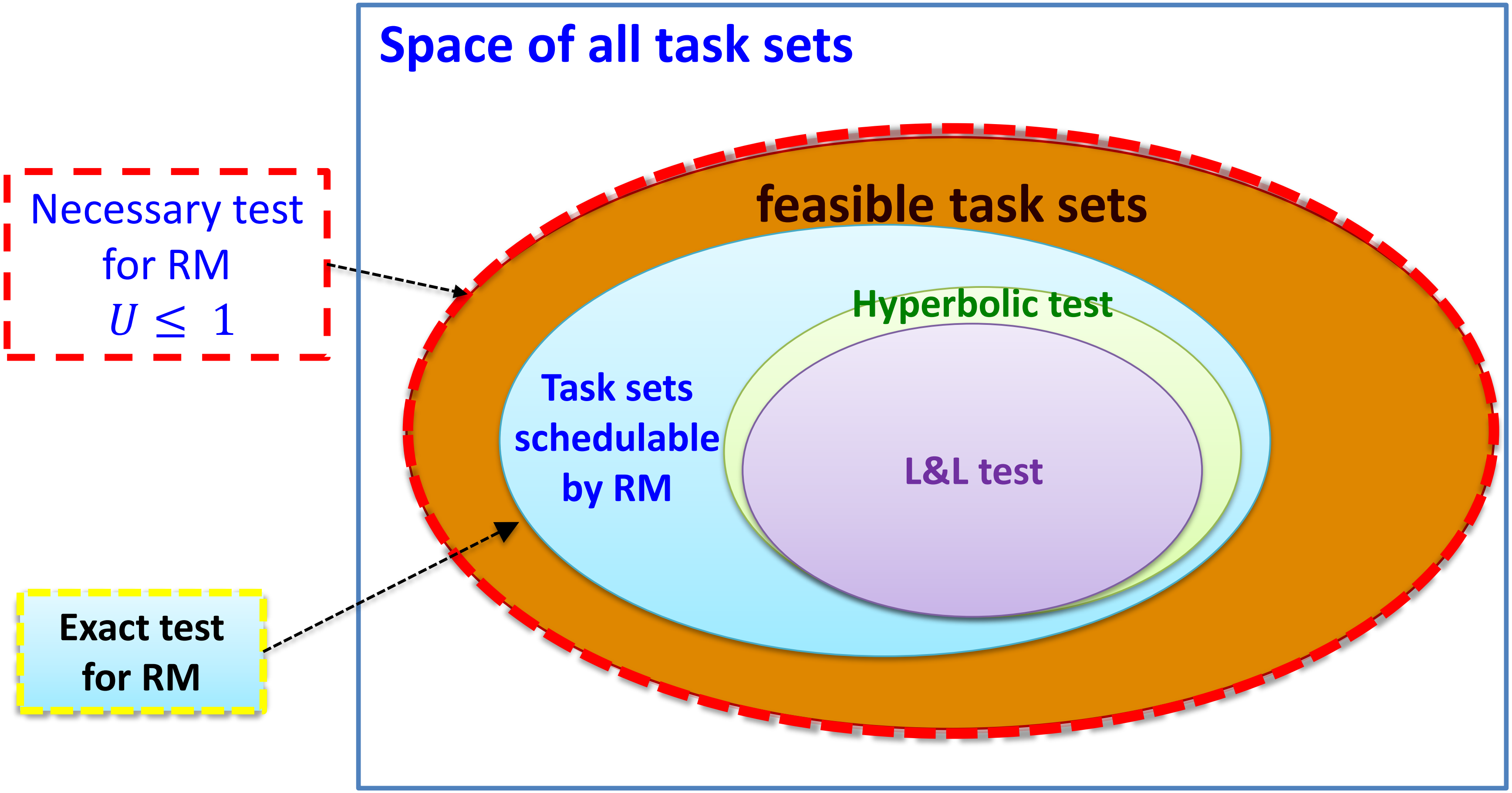
Bonus question

Quiz

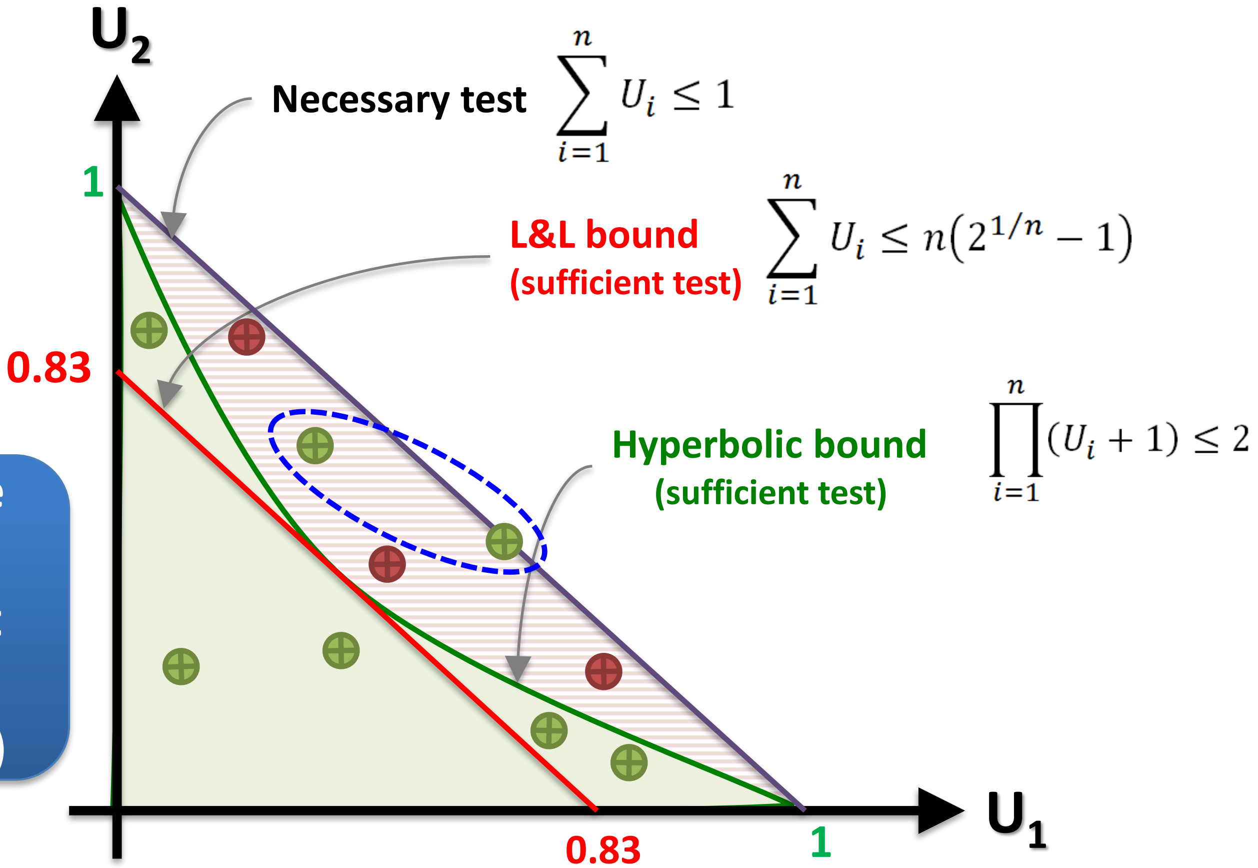
- What is the necessary schedulability/feasibility test for single core preemptive scheduling?

$$U \leq 1$$

The universe of utilization-based tests for RM



Previously on Real-Time Systems



Now we introduce an **exact** schedulability test for RM (and FP in general)

Agenda

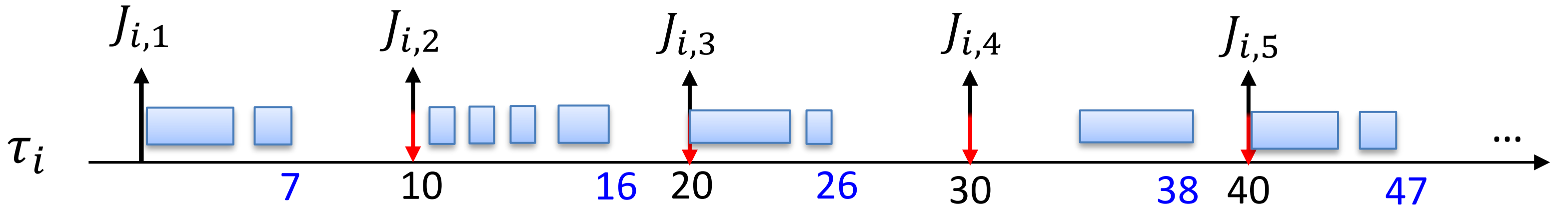
- RM utilization-based tests (reminder)

- *Response-time analysis* for periodic or sporadic tasks under FP
  - **Response-time analysis**
    - **For $\forall \tau_i, \sigma_i = 0$ and $R_i \leq T_i$**
    - **For $\forall \tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$**
    - **For $\forall \tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$ or $R_i > T_i$**
  - **Park's test**

Response-time analysis (RTA)

Unlike utilization-based tests, response-time analysis takes task's **inter-arrival time** and worst-case execution time (**WCET**) into account!

Visualization:



Worst-case response time (WCRT):

$$R_i = \sup\{R_{i,j} \mid \forall j, 1 \le j \le \infty\}$$

The schedulability test:

$$\forall \tau_i \in \tau, \qquad R_i \le D_i$$

It is not practical as it is!
We need a smarter solution

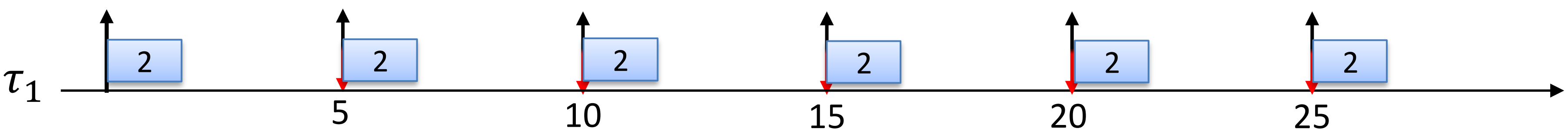
A few questions before we start

Under preemptive FP, **which tasks interfere with the execution of a task τ_i?**

Higher priority tasks

What is the **maximum number of jobs** that a **periodic** task τ_i may release in an **interval of duration *L***?

At most $\left\lceil \frac{L}{T_i} \right\rceil$ jobs if $\sigma_i = 0$



A few questions before we start

Under preemptive FP, **which tasks** **interfere** with the execution of a task τ_i?

Higher priority tasks

What is the **maximum number of jobs** that a **periodic** task τ_i may release in an **interval of duration *L***?

At most $\left\lceil \frac{L}{T_i} \right\rceil$ jobs if $\sigma_i = 0$

What is the **maximum number of jobs** that a **sporadic** task τ_i may release in an **interval of duration *L***?

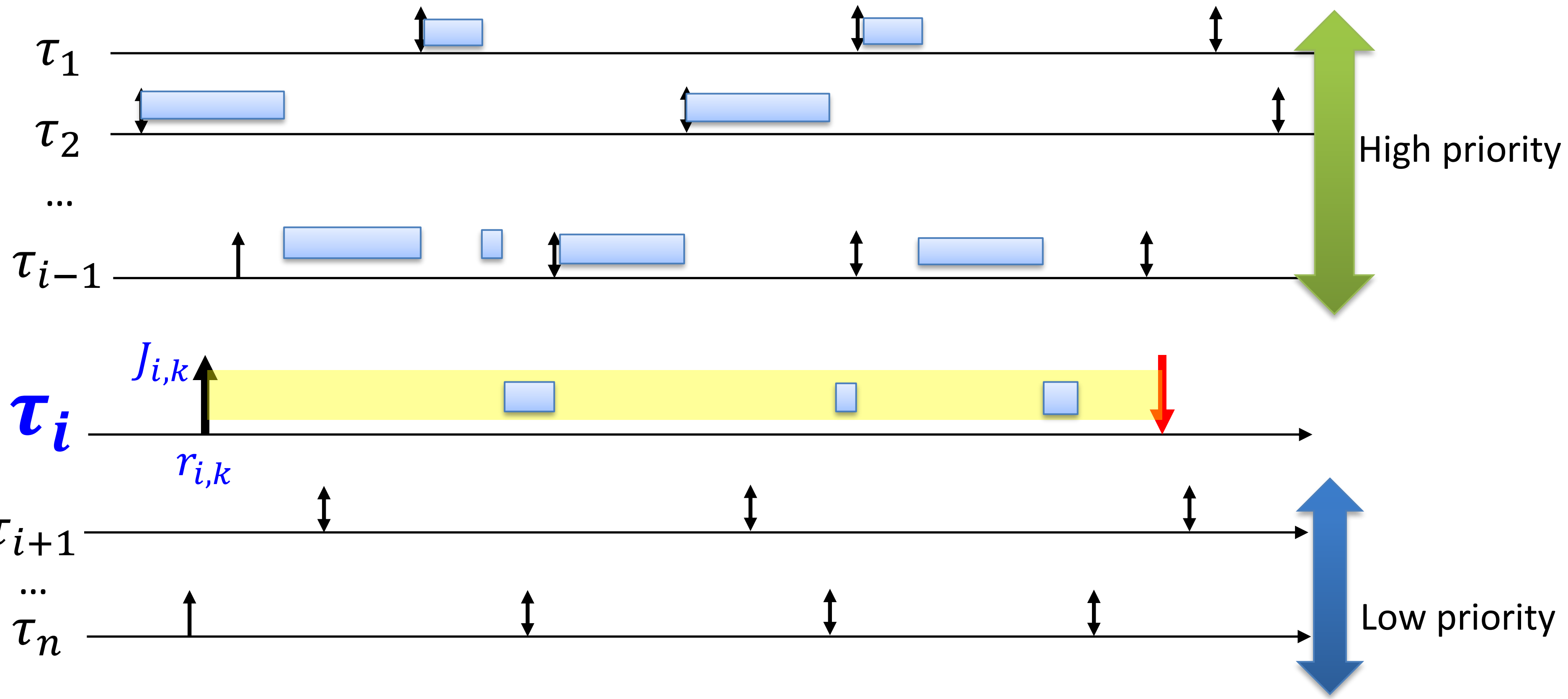
At most $\left\lceil \frac{L}{T_i} \right\rceil$ jobs if $\sigma_i = 0$

What is the **maximum** **workload** that a periodic or sporadic task τ_i may release in an interval of duration L?

At most $\left\lceil \frac{L}{T_i} \right\rceil \times C_i$ if $\sigma_i = 0$

On a search for the worst-case release scenario

What is the worst-case release scenario for a job of task τ_i under FP scheduling?

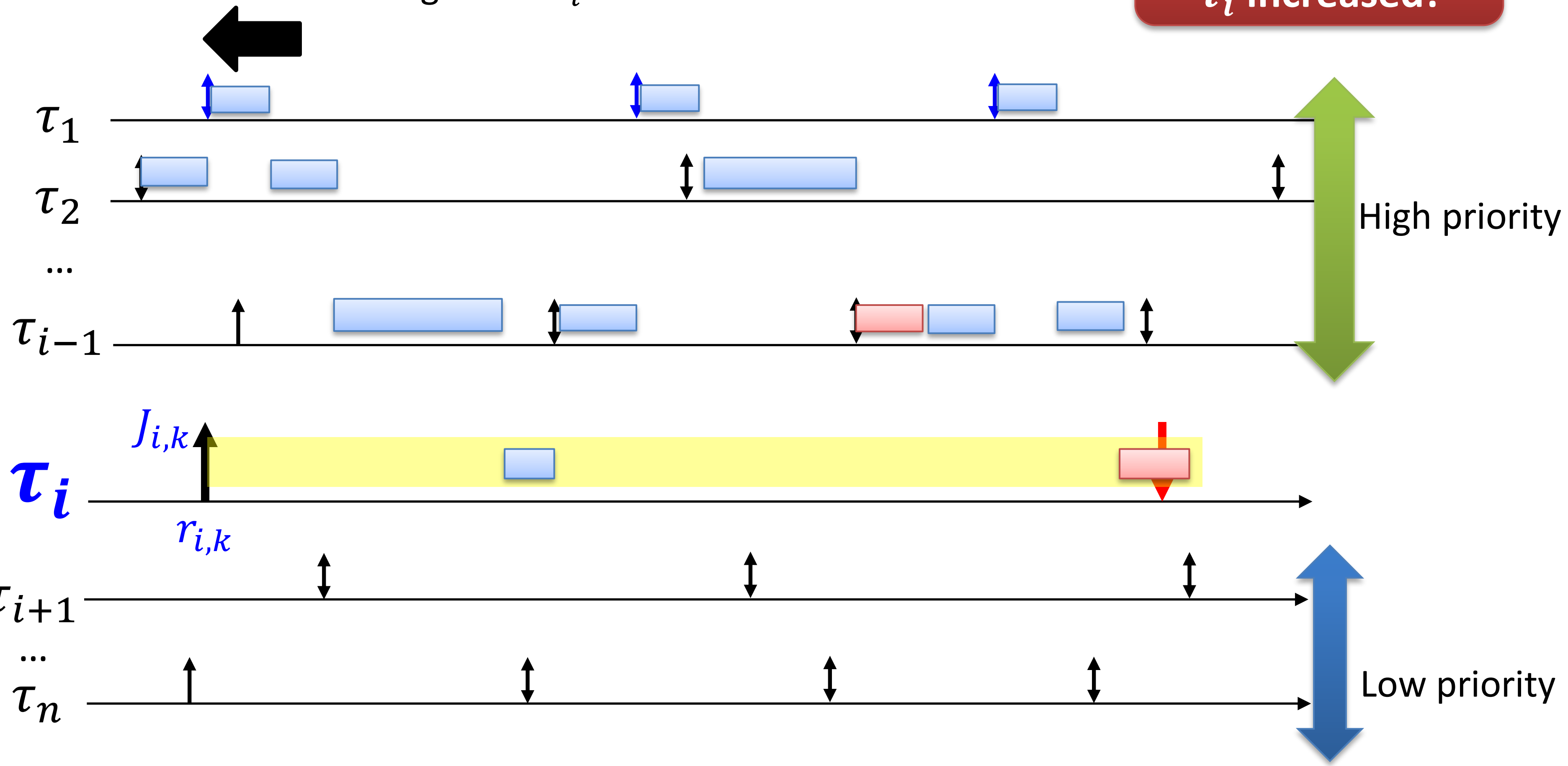


Assume that tasks are sorted and indexed by priority: **P1 > P2> … >Pn**

On a search for the worst-case release scenario

The release time of τ_1 is moved to an earlier time to align with τ_i's release

Response time of τ_i increased!

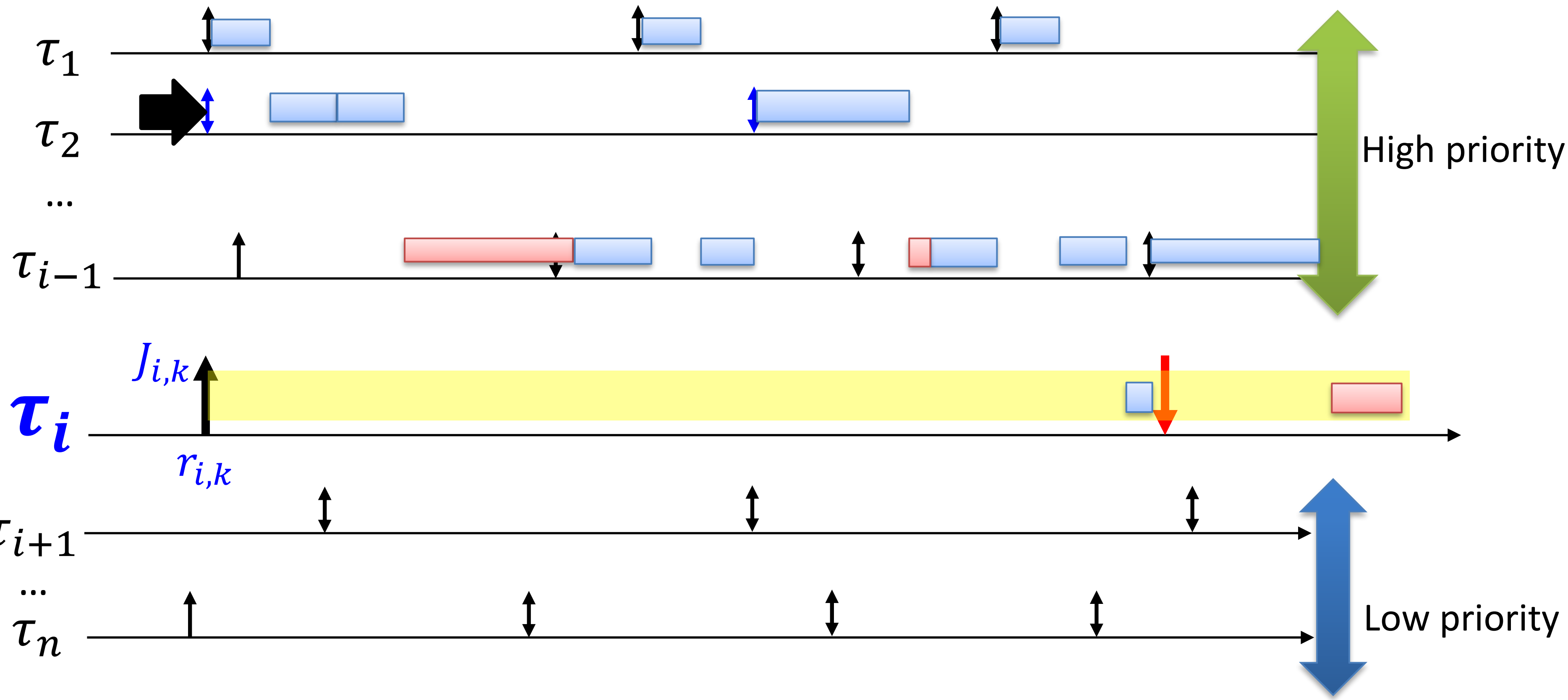


Assume that tasks are sorted and indexed by priority: **P1 < P2< … <Pn**

On a search for the worst-case release scenario

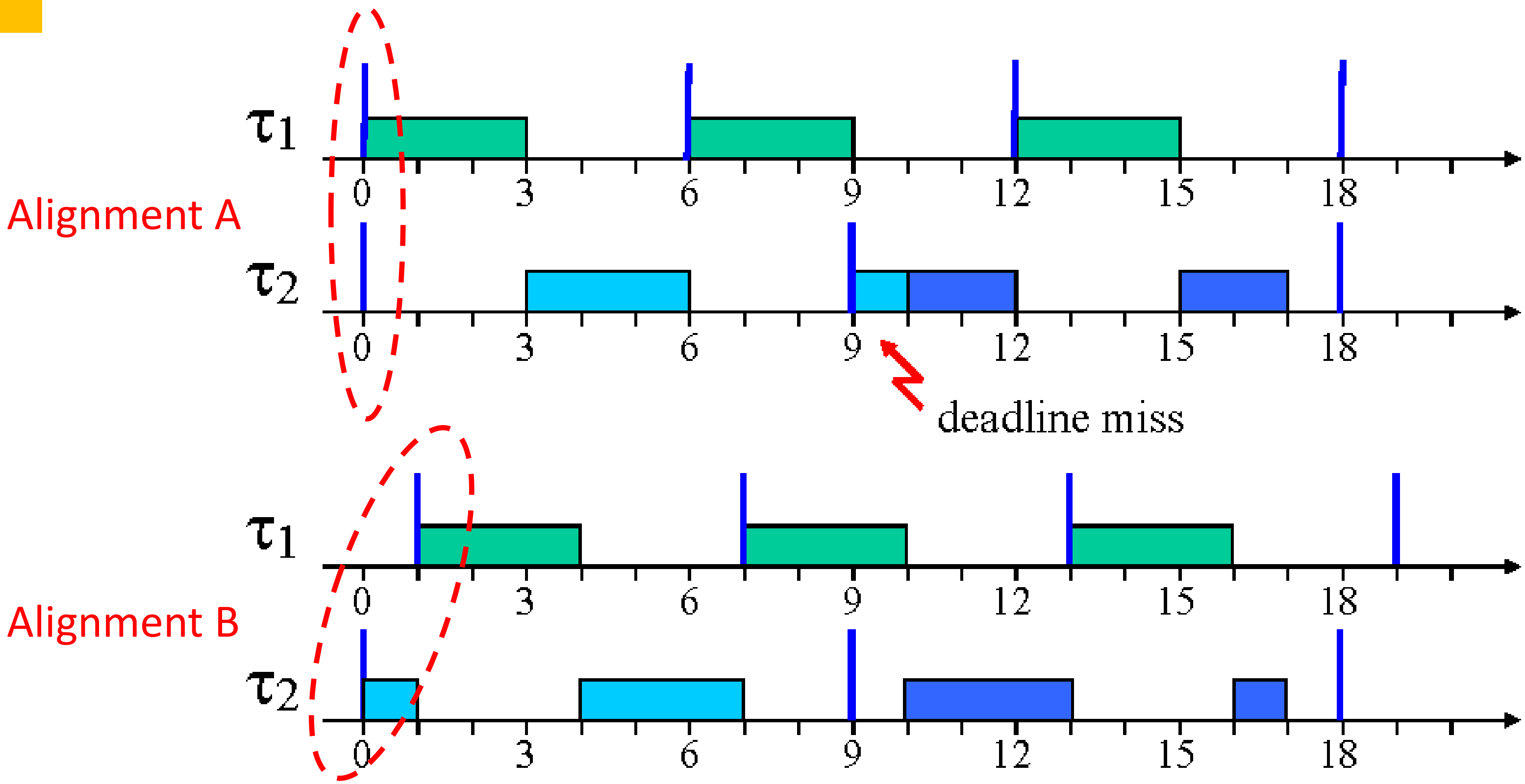
The release time of τ_2 is moved to a later time to align with τ_i's release

Response time of τ_i increased!



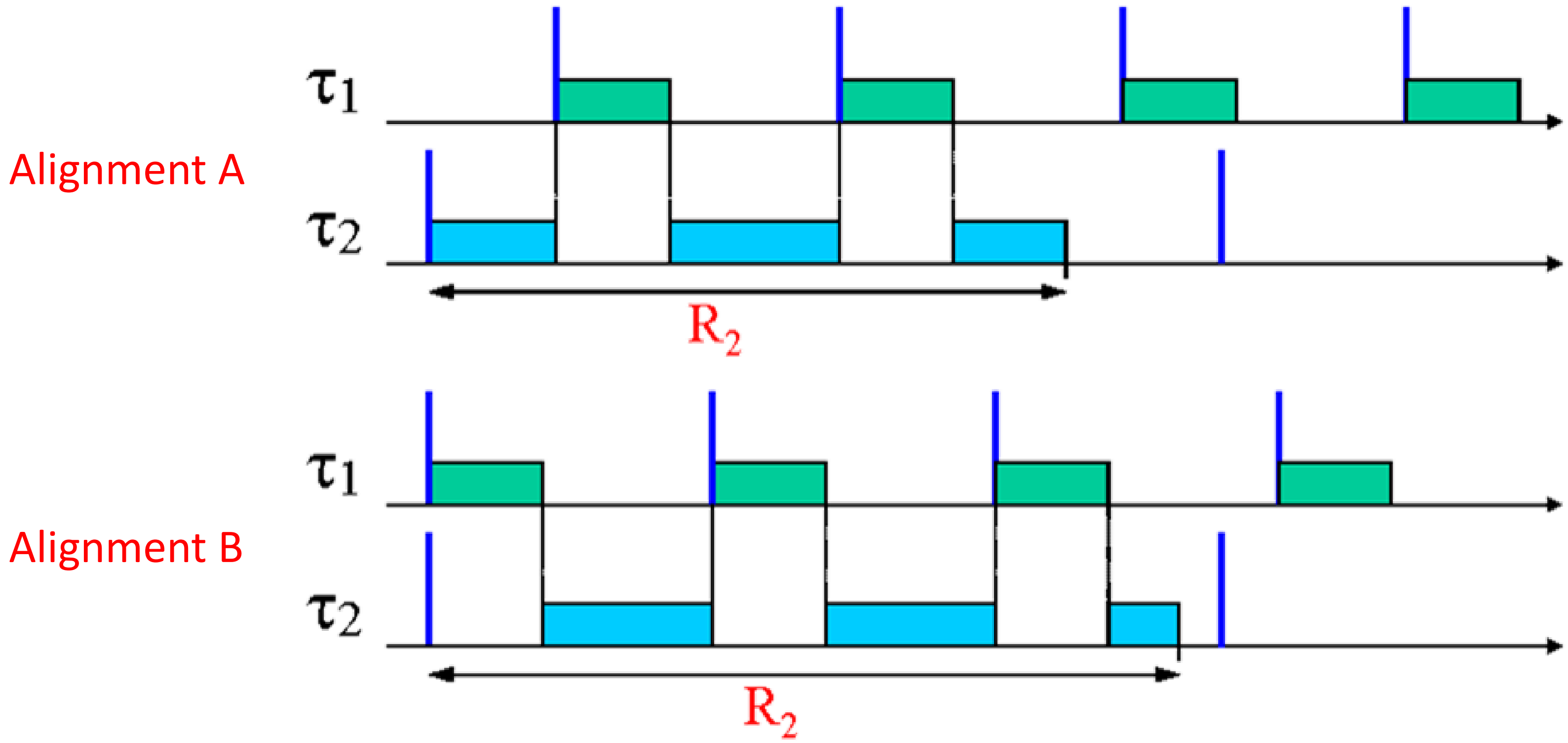
Assume that tasks are sorted and indexed by priority: **P1 < P2< … <Pn**

For fixed-priority scheduling, the WCRT depends on the alignment of release times



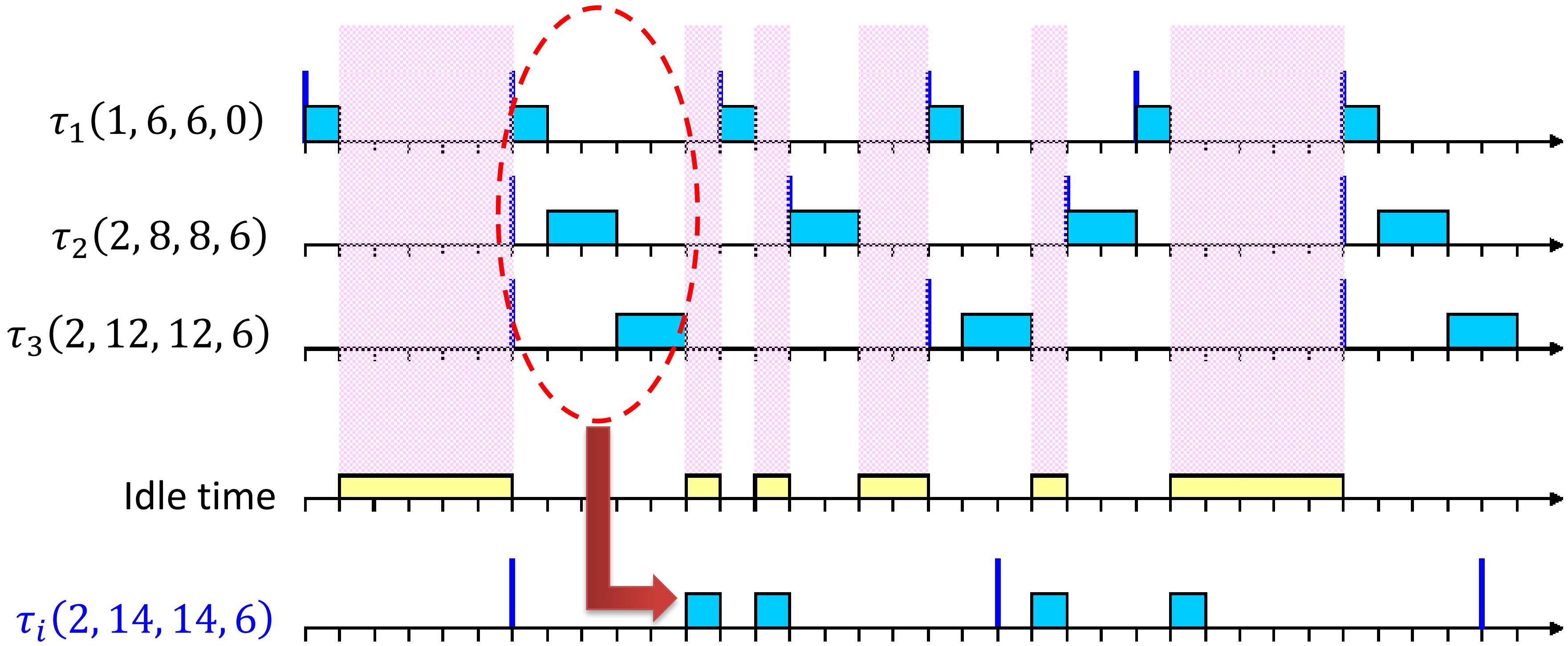
Critical instant

Under FP, for any task τ_i, **the longest response time** occurs when its release coincide with the release of **all higher-priority tasks.**



Critical instant

For **independent preemptive** tasks under fixed priorities, a critical instant of τ_i occurs when it releases a job together with all higher-priority tasks and subsequent jobs are released as fast as possible and execute for their WCET.

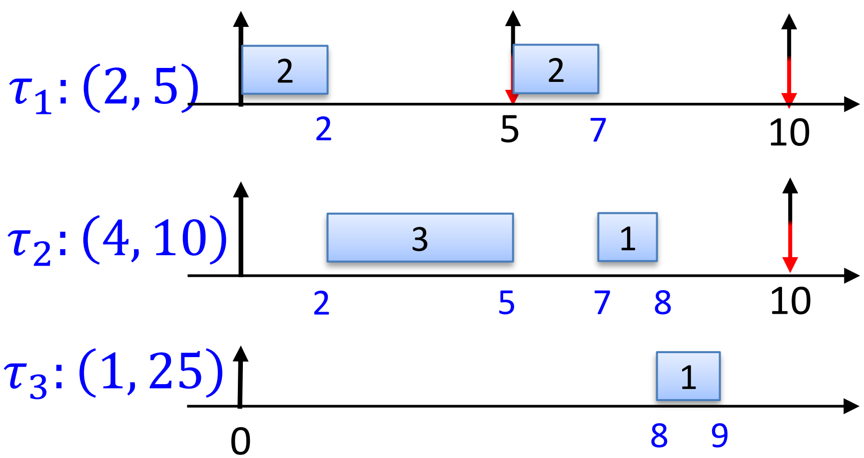


$P_1 > P_2 > P_3 > \cdots > P_i$

Notations: $\tau_i(C_i, T_i, D_i, \phi_i)$

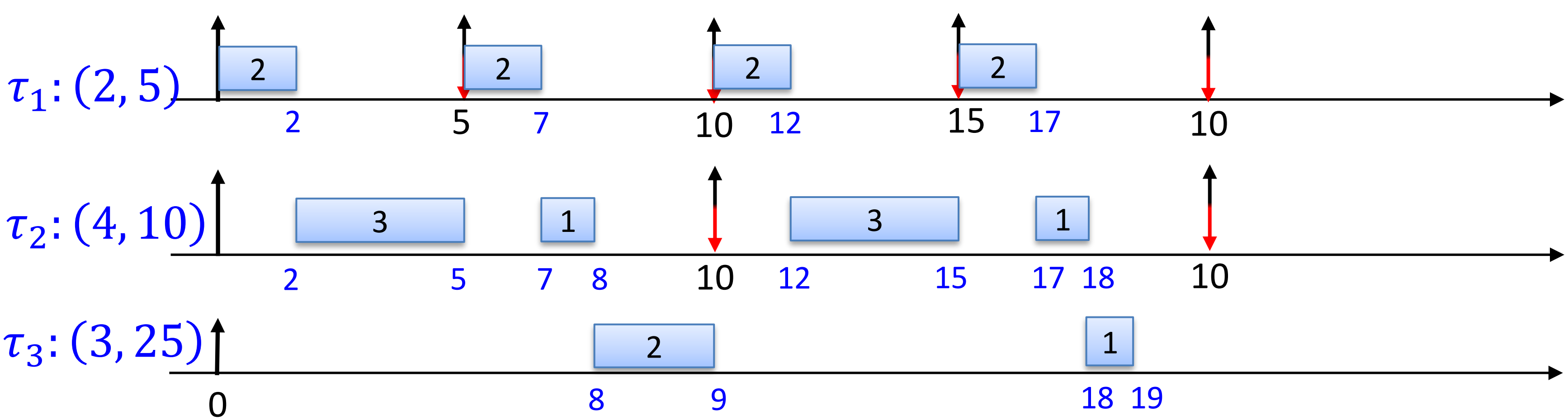
Building an equation for the WCRT

Step 1: If we know that $R_i = X$, then what is the maximum number of jobs of a high-priority task τ_k ($k < i$) that can interfere with τ_i?



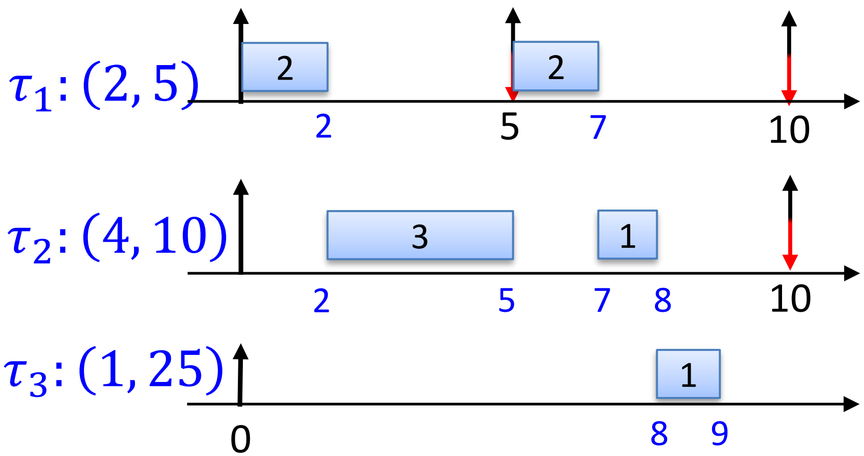
The maximum number of jobs of task τ_k released in an interval of length X is $\left\lceil \frac{X}{T_k} \right\rceil$

Ceiling operator: $\lceil x \rceil = \min\{m \in \mathbb{Z} \mid m \geq x\}$

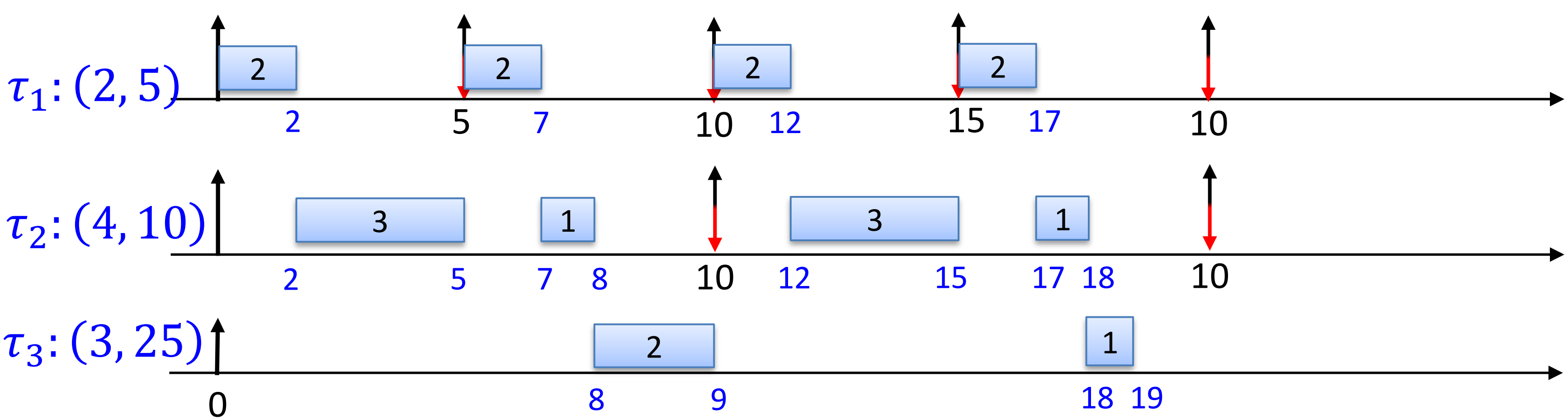


Building an equation for the WCRT

Step 2: What is the maximum workload that can be generated by a higher-priority task τ_k in an interval with length X?

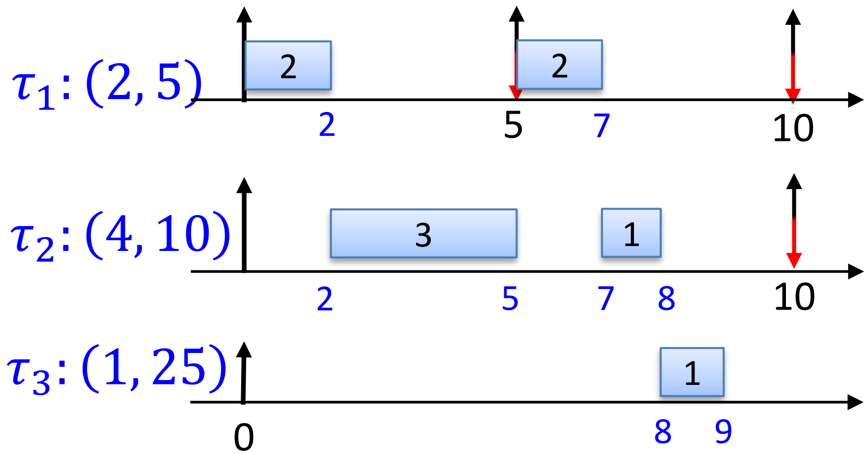


The maximum workload of task τ_k in X is $\left\lceil \frac{X}{T_k} \right\rceil \cdot C_k$



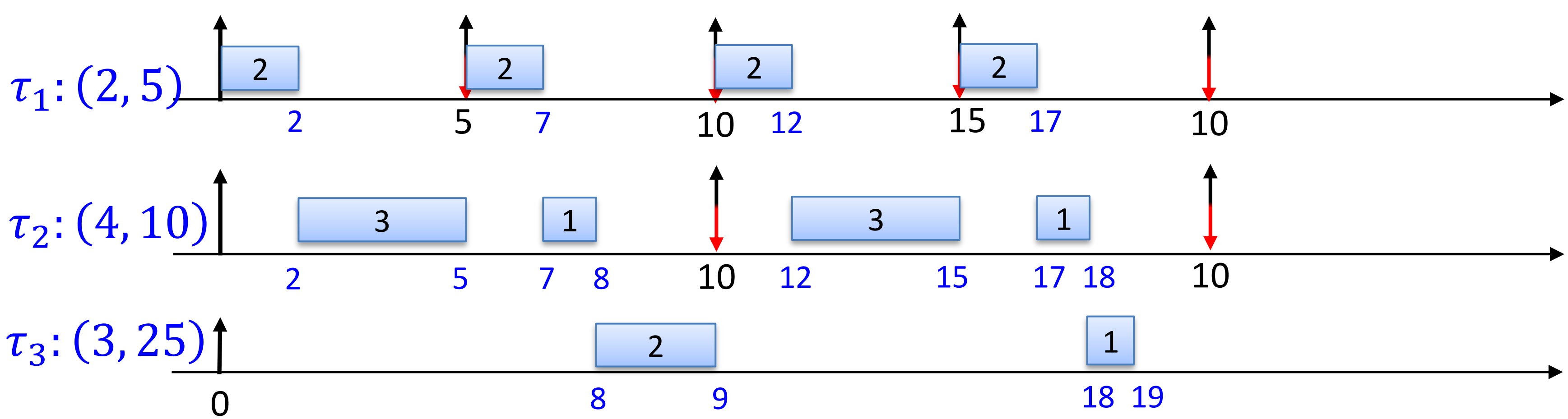
Building an equation for the WCRT

Step 3: **What is the maximum workload generated by ALL higher-priority tasks than τ_i in an interval of length X?**



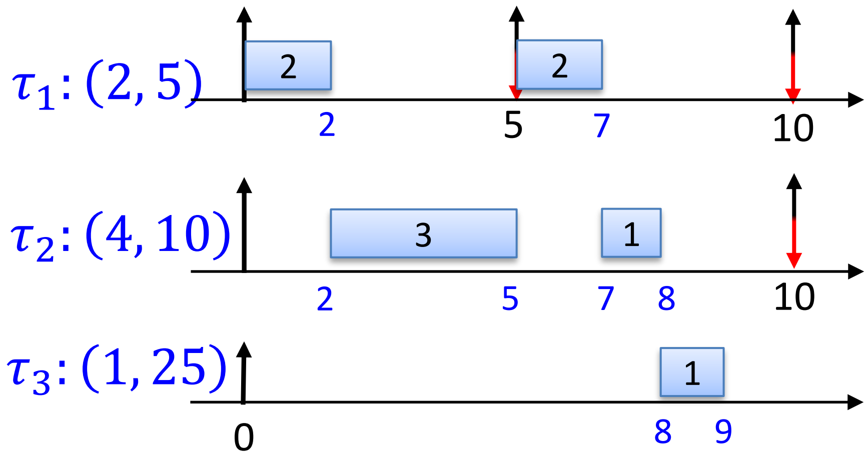
The maximum workload of higher-priority tasks than τ_i is

$$\sum_{k=1}^{i-1} \left\lceil \frac{X}{T_k} \right\rceil \cdot C_k$$



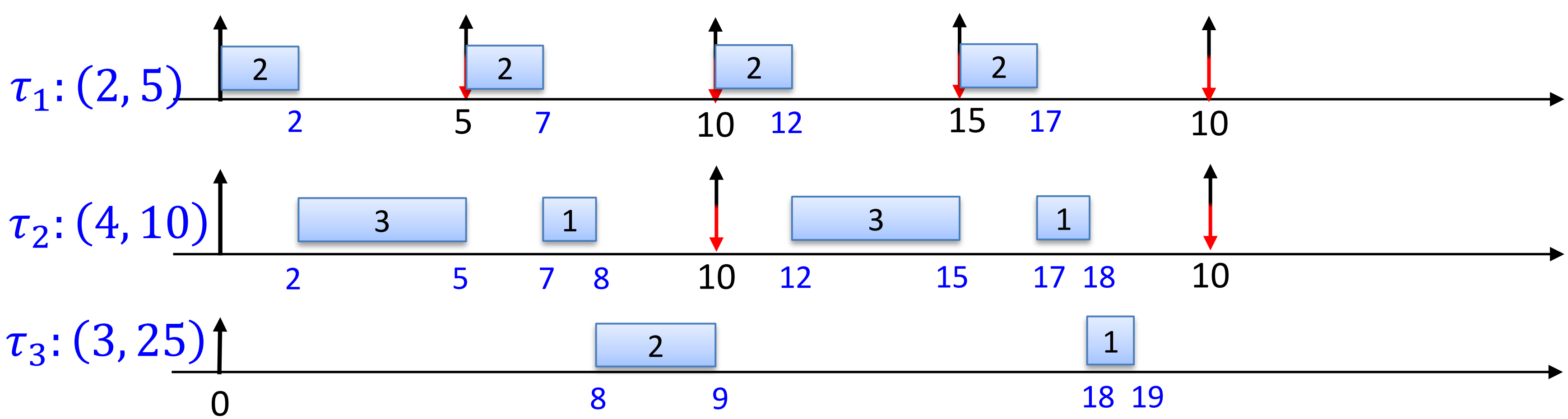
Building an equation for the WCRT

Step 4: What is the maximum workload that must be executed by higher-priority tasks and τ_i itself in an interval of length X?



The maximum workload of higher-priority tasks and τ_i is

$$C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X}{T_k} \right\rceil \cdot C_k$$

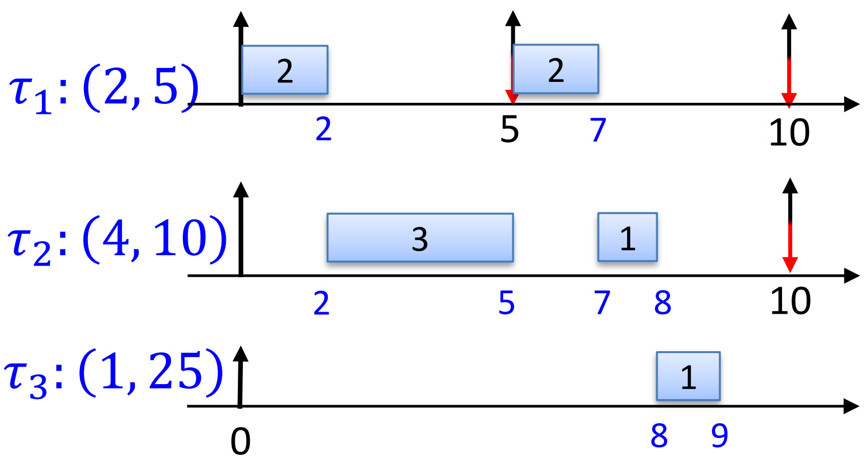


Building an equation for the WCRT

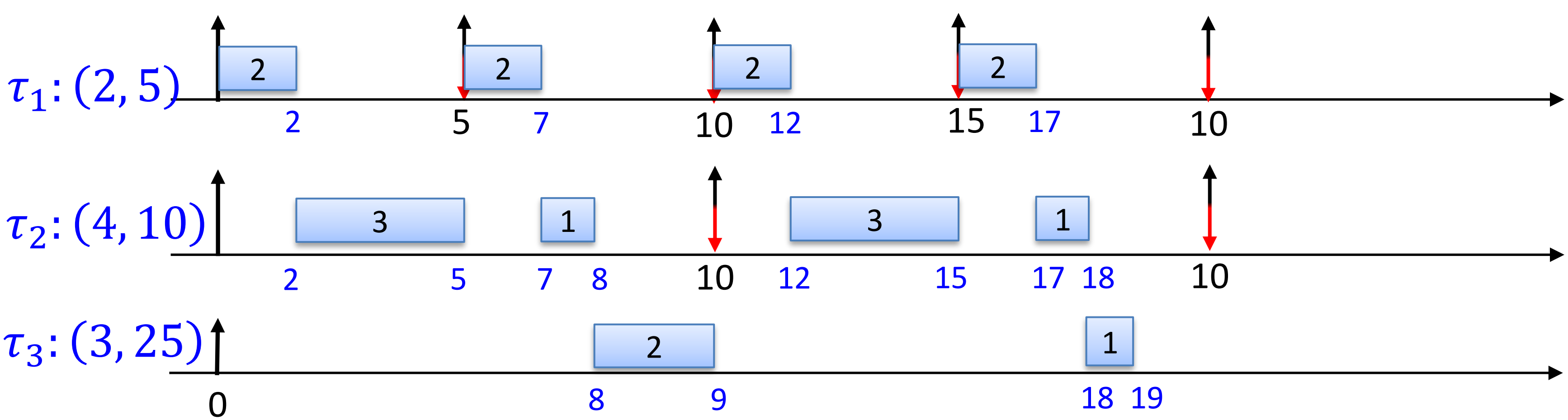
Step 5: How do we find $X = R_i$?

What we execute in X must be equal to the length of X ➔

$$X = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X}{T_k} \right\rceil \cdot C_k$$



Answer: we cannot find it directly! We need to use a fixed-point iteration method!



On the search for X

1. The response time is at least as large as the WCET, so $X \geq C_i$.

$$\boxed{R_i^{(0)} = C_i}$$

2. Figure out how much workload must be executed in X

$$\boxed{R_i^{(n)} \leftarrow C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(n-1)}}{T_k} \right\rceil \cdot C_k}$$

$$C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X}{T_k} \right\rceil \cdot C_k$$

3. Is the workload MORE than the length of X?
   1. Yes: the WCRT of τ_i must have been larger than X
   2. No: Great! So X is a safe upper bound on the WCRT of τ_i

If $R_i^{(n)} > R_i^{(n-1)}$ then continue and find $R_i^{(n+1)}$

Otherwise stop: $R_i^{(n)}$ is the WCRT

Response-time analysis (RTA)

(Audsley '90)

$$R_i = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i}{T_k} \right\rceil \cdot C_k$$

The solution is based on fixed-point iterations:

$$R_i^{(n)} = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(n-1)}}{T_k} \right\rceil \cdot C_k$$

Starting point: $R_i^{(0)} = C_i$

Iterate until: $R_i^{(n)} \le R_i^{(n-1)}$

Usage: Use $R_i^{(0)}$ to calculate $R_i^{(1)}$, then use $R_i^{(1)}$ to obtain $R_i^{(2)}$, …., continue until $R_i^{(n)} = R_i^{(n-1)}$

Understanding the terms

Make sure that the execution of τ_i finishes

For all higher-priority tasks

Multiply that by the WCET of τ_k

$$\begin{cases} R_i^{(n)} = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(n-1)}}{T_k} \right\rceil \cdot C_k \\ R_i^{(0)} = C_i \\ \text{Continue if } R_i^{(n)} > R_i^{(n-1)} \end{cases}$$

Count the maximum number of jobs released by task τ_k in the interval $[0, R_i^{(n-1)})$

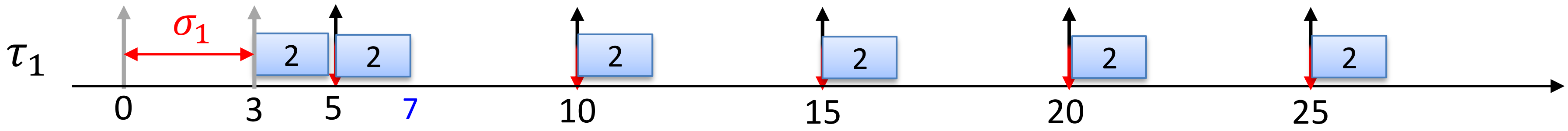
Agenda

- RM utilization-based tests (reminder)
- Response-time analysis for periodic or sporadic tasks under FP
  - **Response-time analysis**
    - For $\forall\tau_i, \sigma_i = 0$ and $R_i \leq T_i$
    - For $\forall\tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$
    - For $\forall\tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$ or $R_i > T_i$
  - **Park test ($\forall\tau_i, \sigma_i = 0$ and $R_i \leq T_i$)**

Maximizing interference

What is the **maximum number of jobs** that a **periodic or sporadic** task τ_i with **release jitter** may release in an **interval of duration *L***?

At most $\left\lceil \frac{L+\sigma_i}{T_i} \right\rceil$ jobs



Maximizing interference

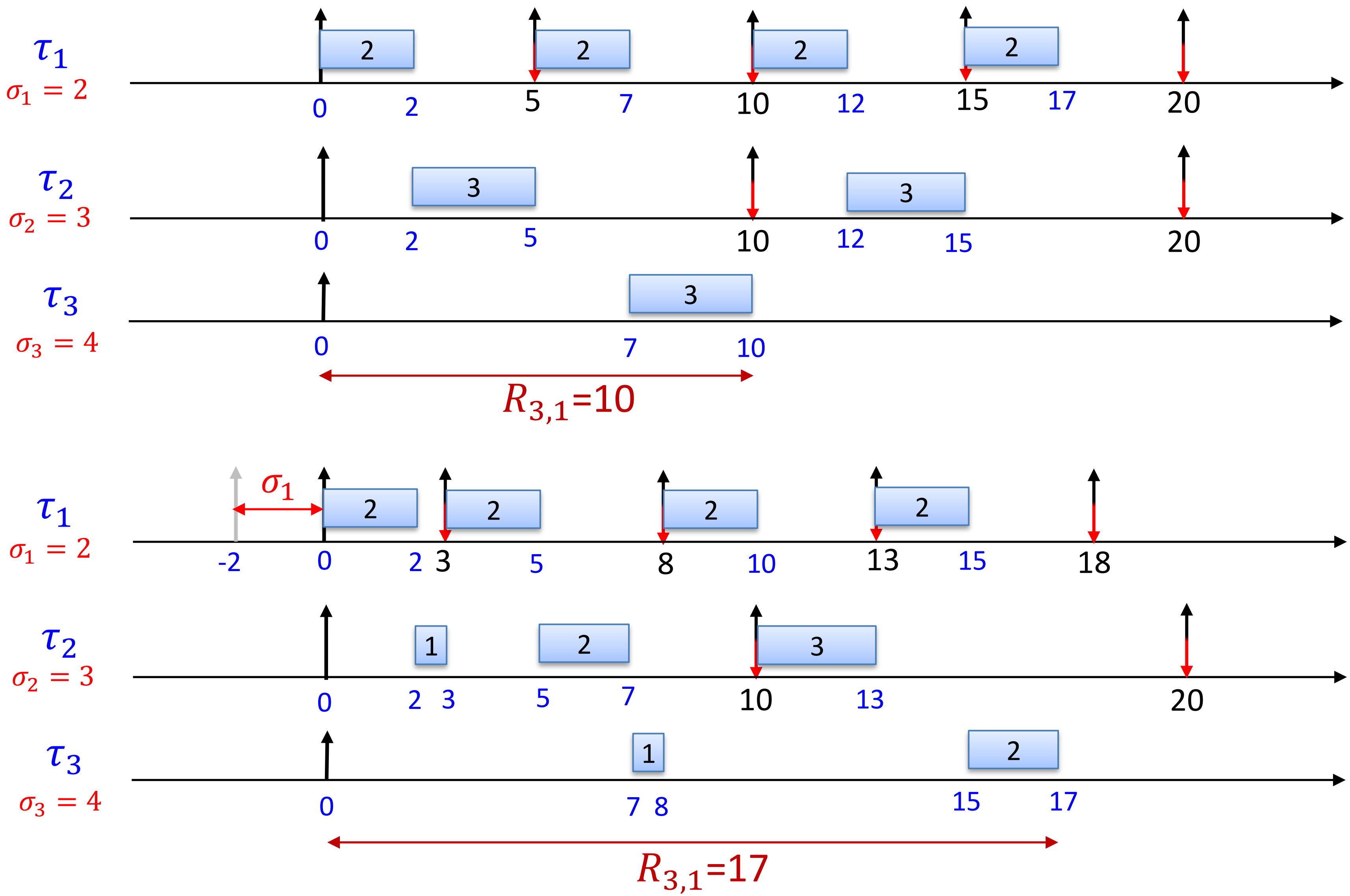
What is the **maximum number of jobs** that a **periodic or sporadic** task τ_i with **release jitter** may release in an **interval of duration *L***?

At most $\left\lceil \frac{L+\sigma_i}{T_i} \right\rceil$ jobs

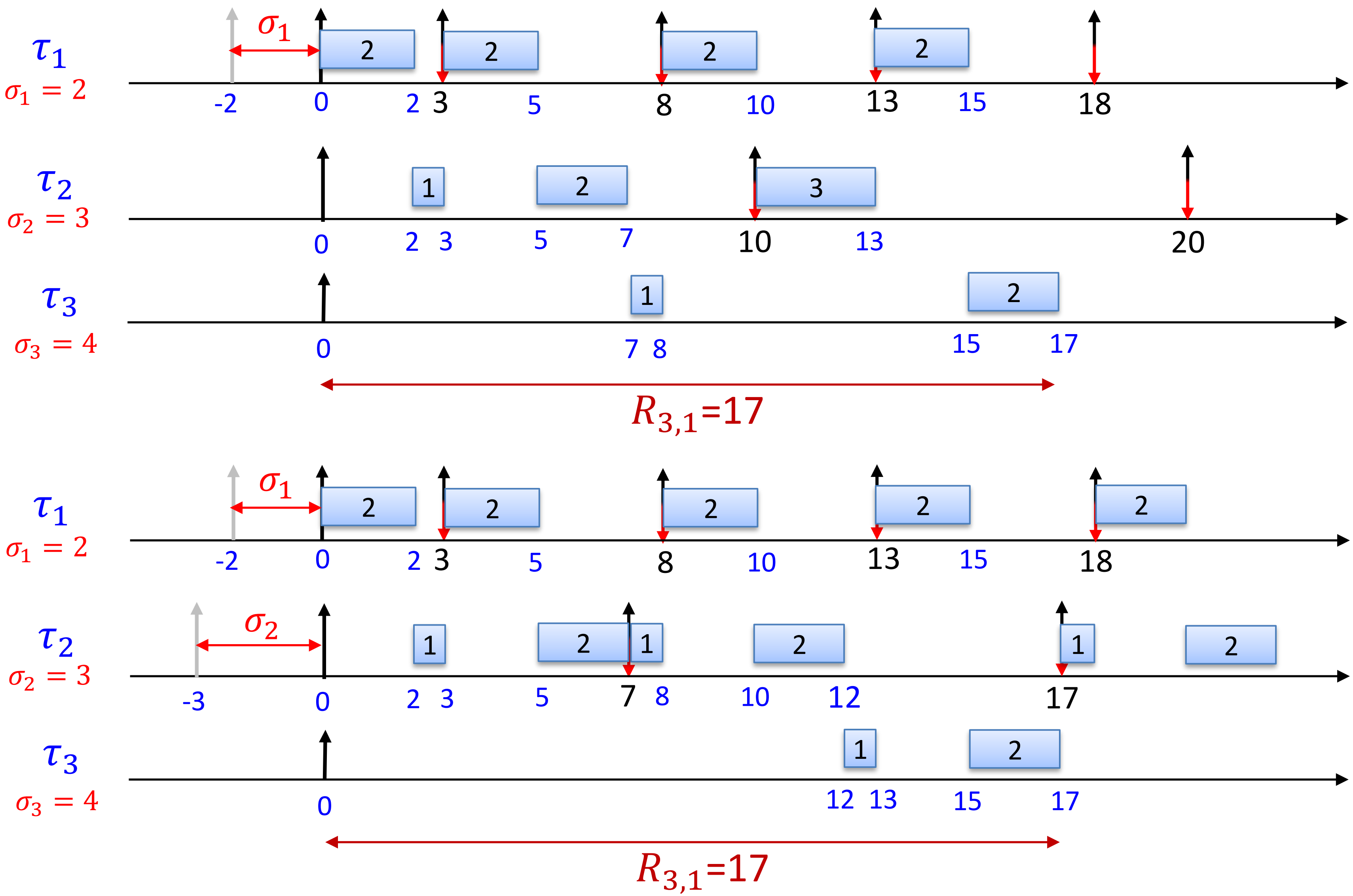
What is the **maximum** **workload** that a periodic or sporadic task τ_i with **release jitter** may release in an interval of duration *L*?

At most $\left\lceil \frac{L+\sigma_i}{T_i} \right\rceil \times C_i$

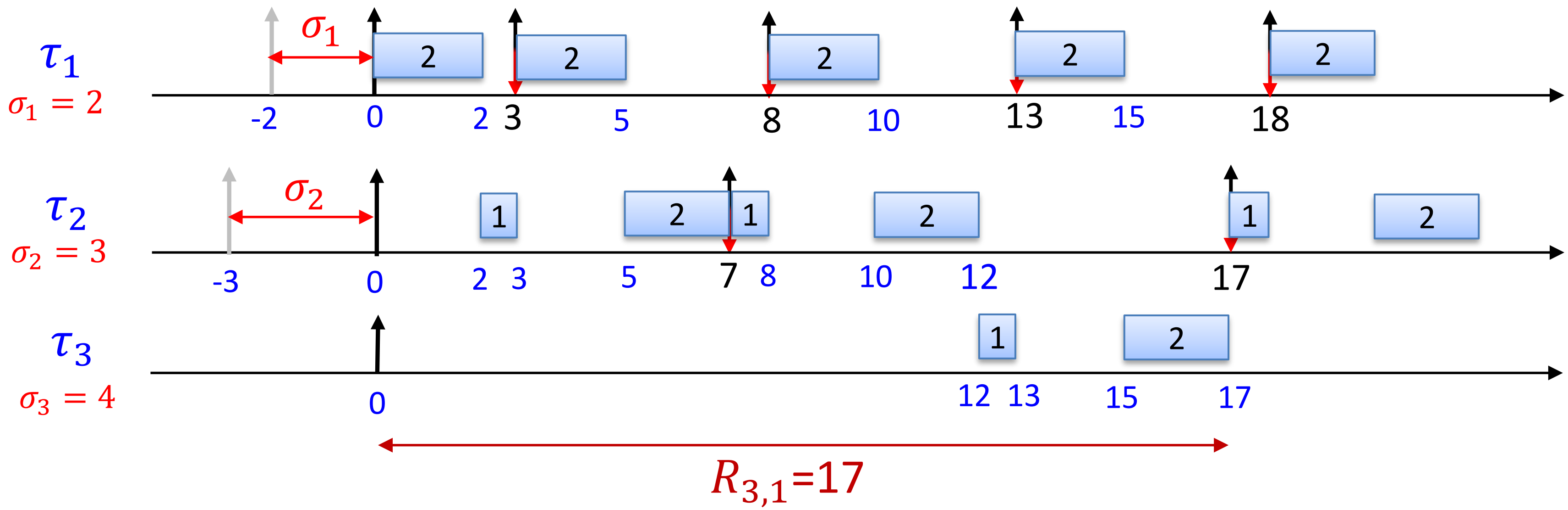
Visualizing the critical instant

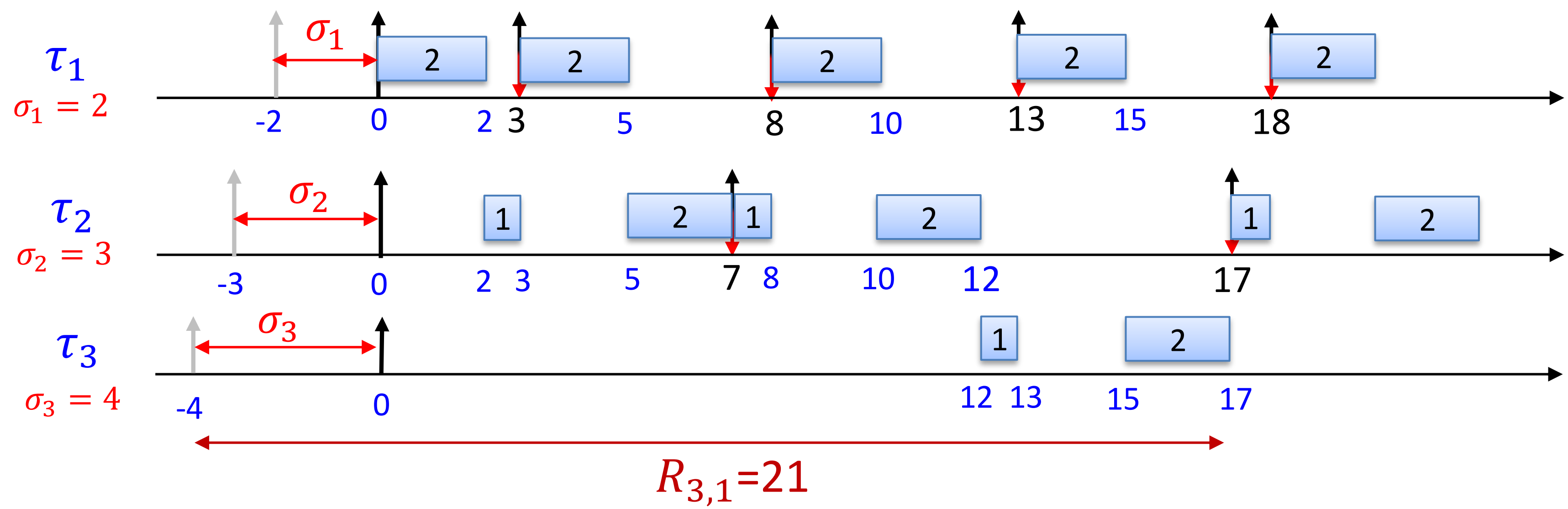


Visualizing the critical instant



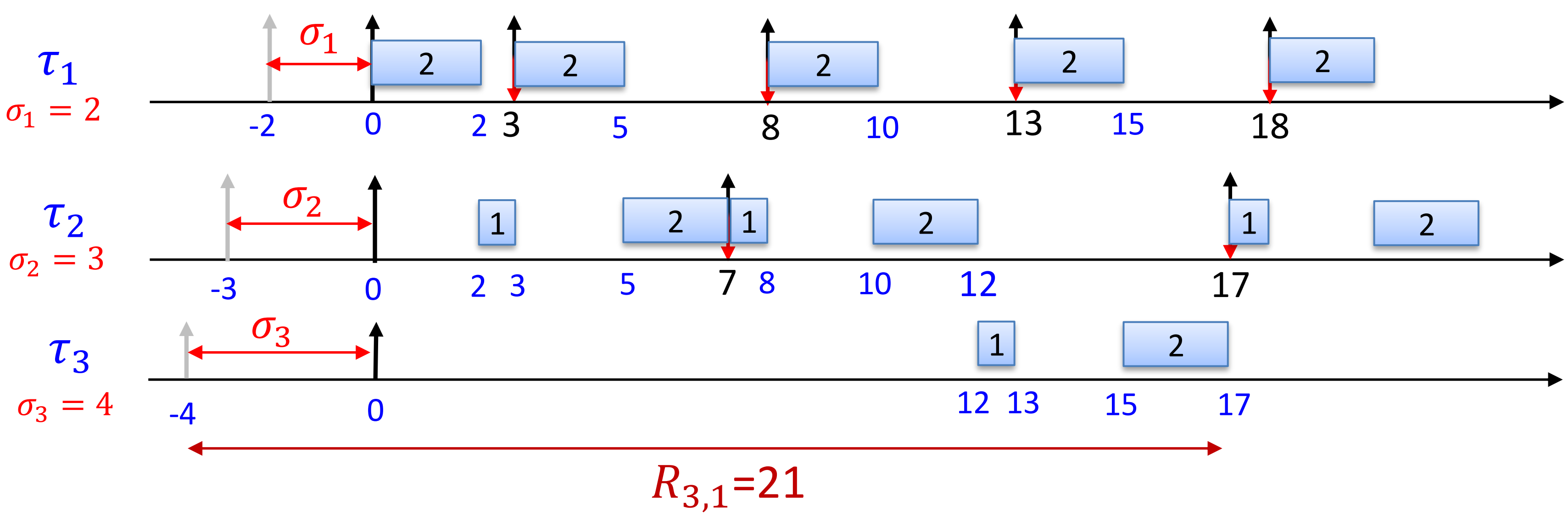
Visualizing the critical instant





Visualizing the critical instant

For **independent preemptive** tasks under fixed priorities, the critical instant of τ_i occurs when it releases a job together with all higher-priority tasks. The first job of each task is released with maximum jitter, subsequent jobs are released as fast as possible and execute for their WCET.



Response-time analysis

$$R_i = \sigma_i + X_i$$

where

$$X_i = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X_i + \sigma_k}{T_k} \right\rceil \cdot C_k$$

The solution is based on fixed-point iterations:

$$X_i^{(n)} = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X_i^{(n-1)} + \sigma_k}{T_k} \right\rceil \cdot C_k$$

Starting point:

$$X_i^{(0)} = C_i$$

Iterate until:

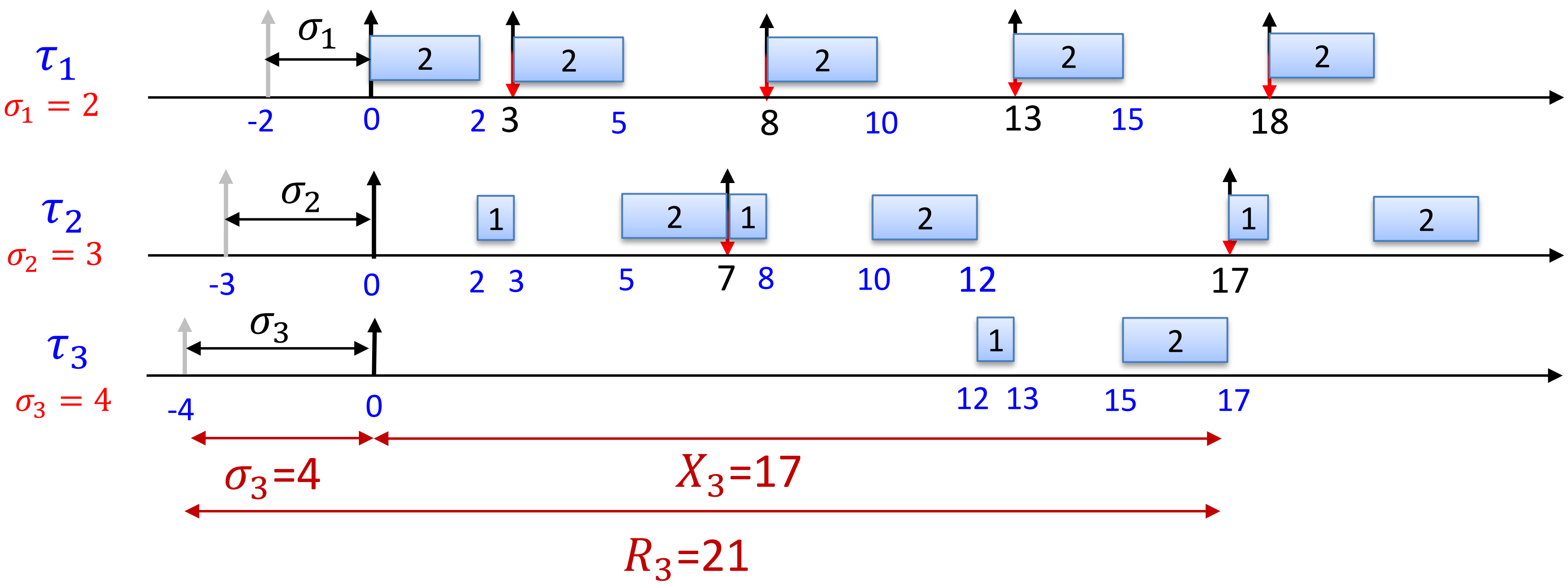
$$X_i^{(n)} \leq X_i^{(n-1)}$$

Why do we add σ_i to X_i?

The **interference** of higher priority tasks (captured in X_i) happens only **after τ_i is released**.

The **response time** of τ_i is the **difference between the finish time and arrival** time of τ_i, which happens up to σ_i time units before the release of τ_i.

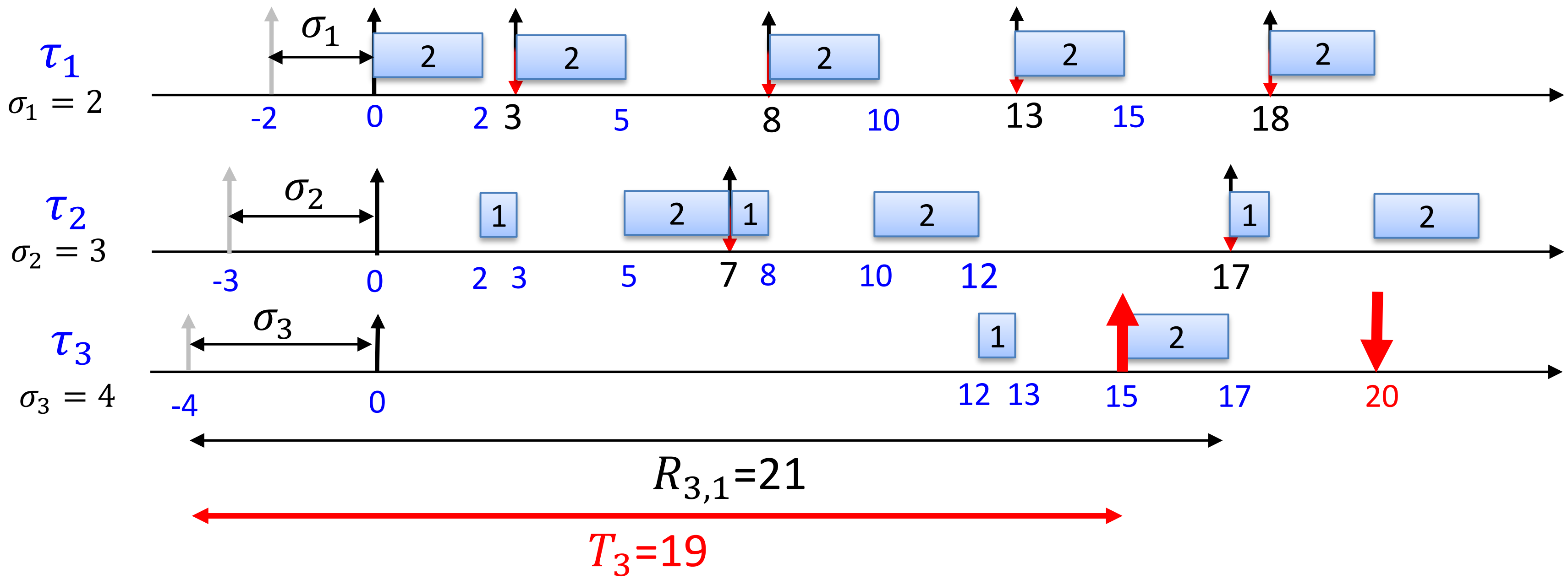
Visualizing the critical instant



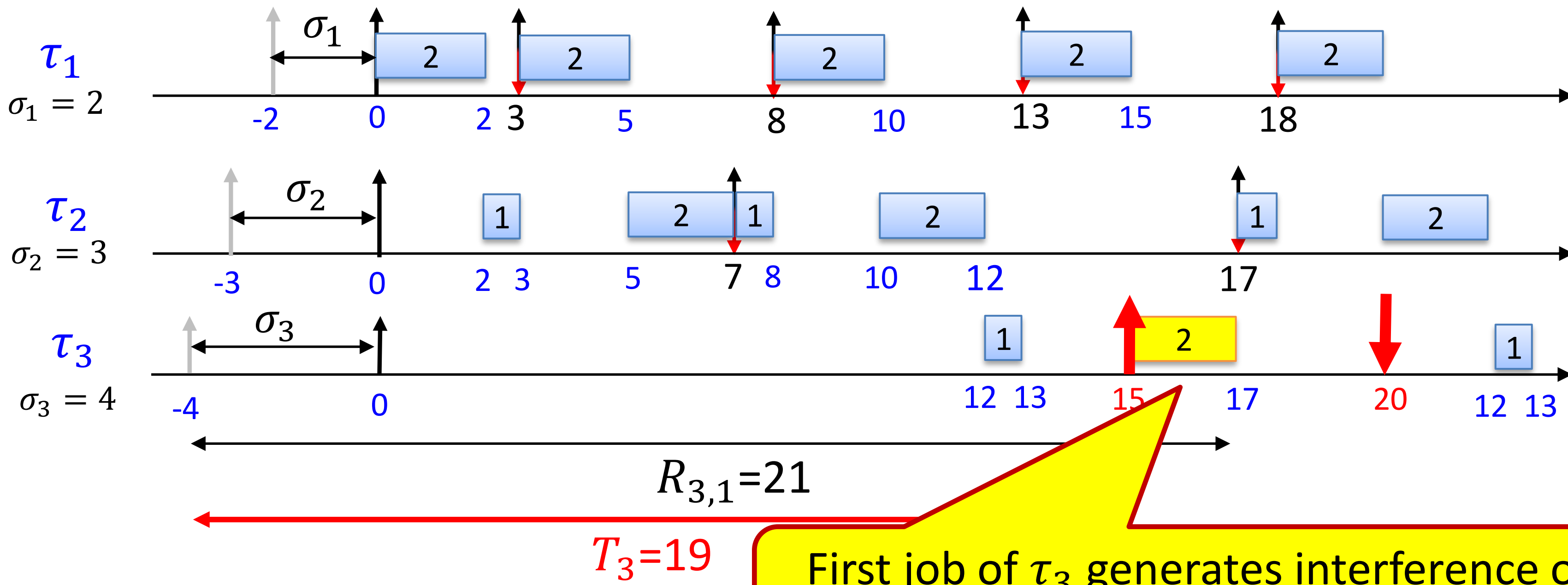
Agenda

- RM utilization-based tests (reminder)
- Response-time analysis for periodic or sporadic tasks under FP
  - **Response-time analysis**
    - For $\forall \tau_i, \sigma_i = 0$ and $R_i \leq T_i$
    - For $\forall \tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$
    - For $\forall \tau_i, \sigma_i \geq 0$ and $R_i \leq T_i$ or $R_i > T_i$
  - **Park test ($\forall \tau_i, \sigma_i = 0$ and $R_i \leq T_i$)**

Effect of $R_i > T_i$



Effect of $R_i > T_i$



First job of τ_3 generates interference on second job of τ_3

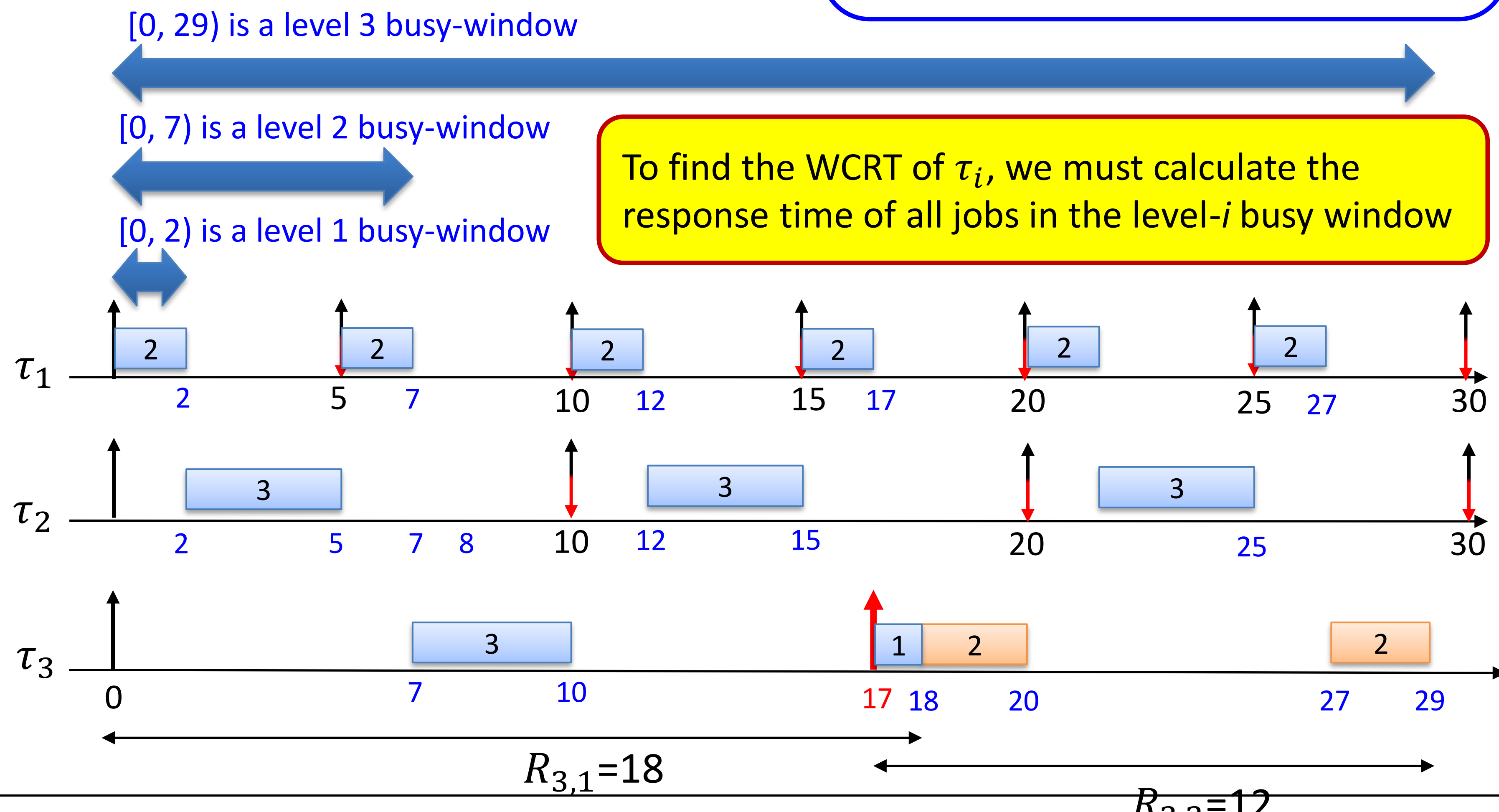
➔ The worst-case response time **may not be experienced by the first job** of τ_i **anymore**!

Solution: analyze **all jobs** released in a ***level-i busy window*** starting at a critical instant

Level *i* busy-window

A **level i busy-window** is a window of time during which the processor is busy executing tasks with a priority equal to or higher than that of τ_i.

To find the WCRT of τ_i, we must calculate the response time of all jobs in the level-*i* busy window



Level *i* busy-window

A **level i busy-window** is a window of time during which the processor is busy executing tasks with a priority equal to or higher than that of τ_i.

The **maximum length of level-i busy window** is given by the first positive solution to:

$$L_i = \sum_{k=1}^{i} \left\lceil \frac{L_i + \sigma_k}{T_k} \right\rceil \cdot C_k$$

The solution is based on fixed-point iterations:

$$L_i^{(n)} = \sum_{k=1}^{i} \left\lceil \frac{L_i^{(n-1)} + \sigma_k}{T_k} \right\rceil \cdot C_k$$

Starting point:

$$L_i^{(0)} = \sum_{k=1}^{i} C_k$$

Iterate until:

$$L_i^{(n)} \leq L_i^{(n-1)}$$

IRiS

Level *i* busy-window

A **level i busy-window** is a window of time during which the processor is busy executing tasks with a priority equal to or higher than that of τ_i.

The **maximum length of level-i busy window** is given by the first positive solution to:

$$L_i = \sum_{k=1}^{i} \left\lceil \frac{L_i + \sigma_k}{T_k} \right\rceil \cdot C_k$$

What is the **maximum number of jobs released by τ_i** in the longest **level-i busy window?**

$$N_i = \left\lceil \frac{L_i + \sigma_i}{T_i} \right\rceil$$

RTA when $R_i > T_i$

Goal:
Calculate the response time of every job of τ_i in the level-i busy window

1. Calculate the **arrival time** of the j^{th} job of τ_i in the level-i busy window (assuming the first job arrive at time 0):

$$a_{i,j} = (j-1) \times T_i$$

2. Calculate the worst-case **finish time** of the j^{th} job of τ_i in the level-i busy window:

$$f_{i,j} = X_{i,j} + \sigma_i$$

where

$$X_{i,j} = (j \times C_i) + \sum_{k=1}^{i-1} \left\lceil \frac{X_{i,j} + \sigma_k}{T_k} \right\rceil \cdot C_k$$

3. Calculate the worst-case **response time** of the j^{th} job of τ_i:

$$R_{i,j} = f_{i,j} - a_{i,j}$$

4. Calculate the worst-case **response time** of τ_i:

$$R_i = \max_{\forall k, 1 \le j \le N_i} (R_{i,j})$$

Understanding the terms

j jobs of τ_i must execute until the finish time of the j^{th} jobs of τ_i

For all higher-priority tasks

$$X_{i,j} = j \times C_i + \sum_{k=1}^{i-1} \left\lceil \frac{X_{i,j} + \sigma_k}{T_k} \right\rceil \cdot C_k$$

Multiply that by the WCET of τ_k

Time **from the release** of the first job τ_i to the finish time the j^{th} jobs of τ_i

Count the maximum number of jobs released by task τ_k in the interval [0, X_i)

$$f_{i,j} = X_{i,j} + \sigma_i$$

Time **from the arrival** of the first job τ_i to the finish time the j^{th} jobs of τ_i

Agenda

- RM utilization-based tests (reminder)
- Response-time analysis for FP
  - **Response-time analysis**
  - **Park test ($\forall \boldsymbol{\tau_i}, \boldsymbol{\sigma_i} = \mathbf{0}$ and $\boldsymbol{R_i} \leq \boldsymbol{T_i}$)**

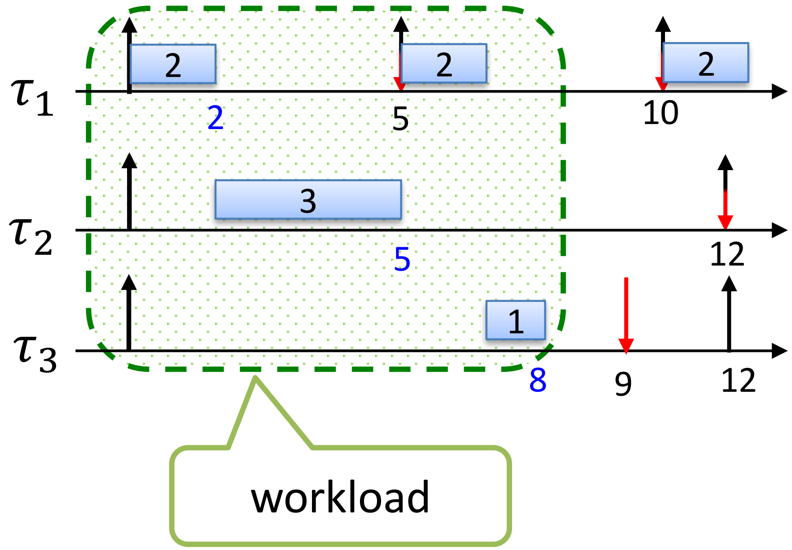
Using RTA to build fast sufficient tests when $\forall \tau_i, \sigma_i = 0$ and $R_i \le T_i$

Park's test

$$\begin{cases} R_i^{(n)} = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(n-1)}}{T_k} \right\rceil \cdot C_k \\ R_i^{(0)} = C_i \\ \text{Continue if } R_i^{(n)} > R_i^{(n-1)} \end{cases}$$

Idea:
Instead of searching for the exact WCRT, just check if the workload that must be completed before the deadline is smaller than the deadline of the task

How?



M. Park and H. Park. An efficient test method for rate monotonic schedulability. IEEE Transactions on Computers, 63(5):1309–1315, 2014

Using RTA to build fast **sufficient** tests when $\forall \tau_i, \sigma_i = 0$ and $R_i \le T_i$

Park's test

$$\begin{cases} R_i^{(n)} = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(n-1)}}{T_k} \right\rceil \cdot C_k \\ R_i^{(0)} = C_i \\ \text{Continue if } R_i^{(n)} > R_i^{(n-1)} \end{cases}$$

Idea:
Instead of searching for the exact WCRT, just check if the workload that must be completed before the deadline is smaller than the deadline of the task

$$\overbrace{C_i + \sum_{k=1}^{i-1} \left\lceil \frac{D_i}{T_k} \right\rceil \cdot C_k}^{\text{workload that must be completed before the deadline}} \le D_i$$

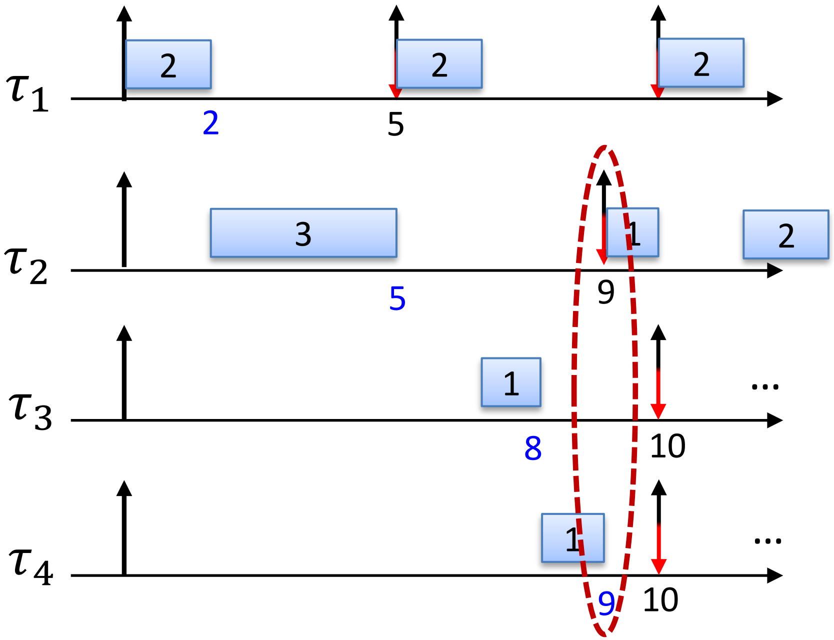
What is the computational complexity per task?

$\boldsymbol{O(n)}$

Why Park's test is just a sufficient test?

M. Park and H. Park. An efficient test method for rate monotonic schedulability. IEEE Transactions on Computers, 63(5):1309–1315, 2014

Park's test is only just a sufficient test



Is this task set schedulable by RM?

Yes! The critical instant is schedulable.
=> The WCRT of each task is smaller than its deadline

What is the WCRT of τ_3 and τ_4?

8 and 9, respectively.

What does Park's test say about τ_4?

It says τ_4 will miss its deadline

Why did it happen?

When a higher-priority task is released after the WCRT of the task under study, Park's test includes that task within the workload that must be finished by the deadline! Namely, it upper-approximate the interfering workload!

$$C_4 + \sum_{k=1}^{4-1} \left\lceil \frac{D_4}{T_k} \right\rceil \cdot C_k \leq D_4 \quad \Rightarrow 1 + \left(\left\lceil \frac{10}{5} \right\rceil \cdot 2 \right) + \left(\left\lceil \frac{10}{9} \right\rceil \cdot 3 \right) + \left(\left\lceil \frac{10}{10} \right\rceil \cdot 1 \right)$$

$$= 1 + 4 + 6 + 1 = 12 \not\leq 10$$

Summary

Schedulability analyses for preemptive FP scheduling:

- Utilization-based tests
  - LL-bound: $\boldsymbol{U \leq U_{lb}}$
  - Hyperbolic bound: $\boldsymbol{f(U_1, \dots, U_n) \leq 2}$
- Response time analysis $\forall \boldsymbol{i}, \boldsymbol{R_i \leq D_i}$
  - Simplified version: Park's test (over-approximates the interfering workload)
- Simplified test for harmonic task tests: $U \leq 1$