TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Non-Work-Conserving Scheduling of Non-Preemptive Hard Real-Time Tasks Based on Fixed Priorities

Mitra Nasri Gerhard Fohler

Chair of Real-Time Systems
Technische Universität Kaiserslautern
Germany

Nov. 2015

Why Non-preemptive Scheduling?

It is inevitable in many systems

- Because of design or architecture
- CAN networks
- GPU

More timing predictability

- Better estimation of the worst-case execution time (WCET)
- More predictability in cache behavior

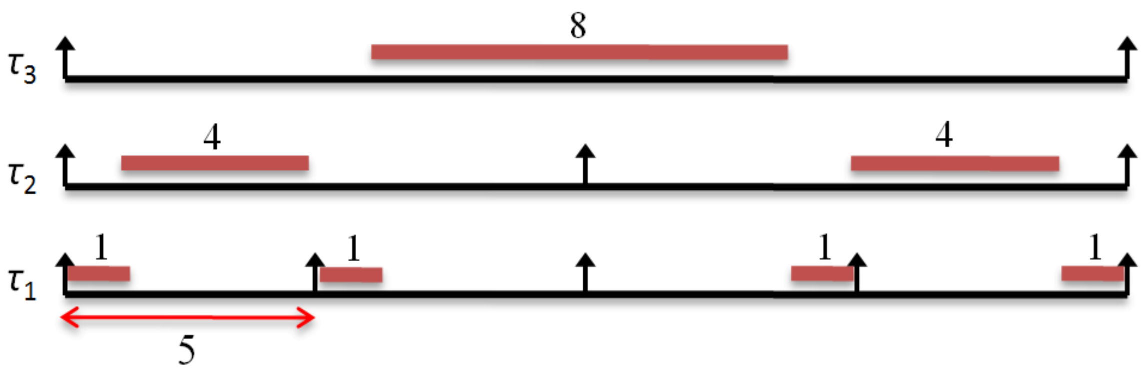
Preemption is expensive

- Context switch overheads
- Destructing cache affinity
- Shared resources (need mutual exclusion)

Application's Desire

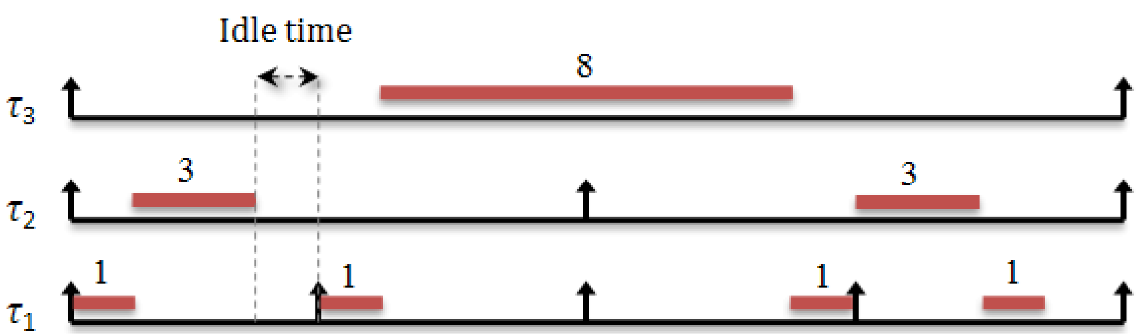
- Control applications are affected by I/O delay (preemption length)

Why Non-preemptive Scheduling is Hard?



Schedulable by npEDF

Not schedulable by npEDF



Schedulable by a non-work conserving scheduling algorithm

Why Non-preemptive Scheduling is Hard? (cont.)

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Without considering idle times in the schedule, we cannot find a solution.

- **No known <u>optimal scheduling policy</u>**
- **No known strategy for <u>idle time insertion</u>**

Needs an exhaustive search
over <u>all jobs</u> and <u>all possible values/locations of idle times</u>

Preemptive, D=T

Preemptive, D<T

Non-Preemptive

State of the Art

Non-Preemptive Scheduling

Schedulability Test
(npEDF, npRM, FP)

Kim 1980, Jeffay 1991, George 1996, Park 2007, Andersson 2009, Marouf 2010, …

Non-Work Conserving Algorithms
(for Harmonic Tasks)

Deogun 1986, Cai 1996, Nasri 2014

Heuristic Algorithms
(gEDF, cEDF)

Ekelin 2006, Li 2007

Deogun 1986

No deadline miss if

- Constant integer period ratio $K \geq 3$

Cai 1996

No deadline miss if tasks have

- Integer period ratio $K_i \geq 3$
- Constant period ratio $K = 2$

Exponential complexity

Nasri 2014

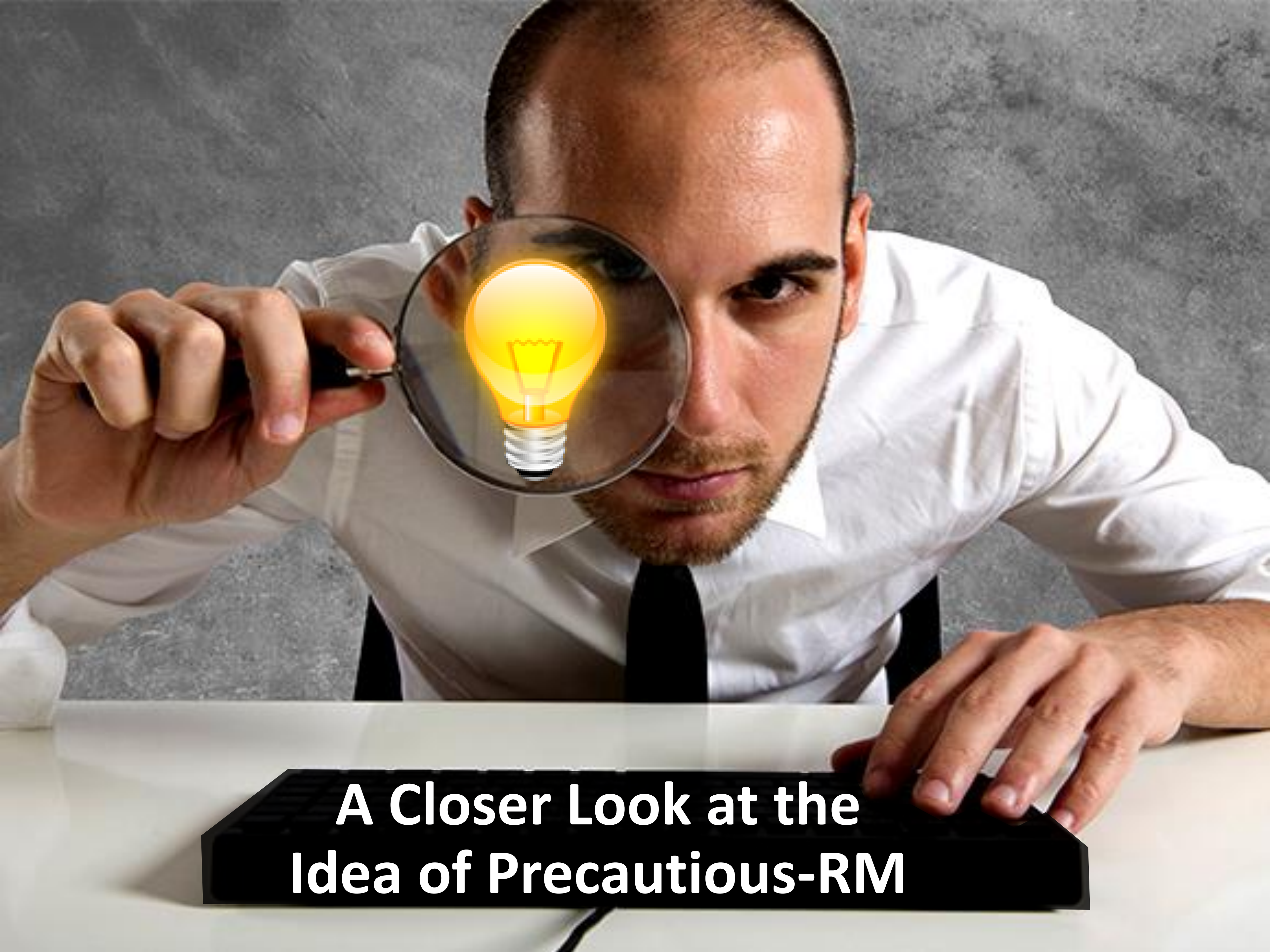
Precautious-RM

No deadline miss if tasks have

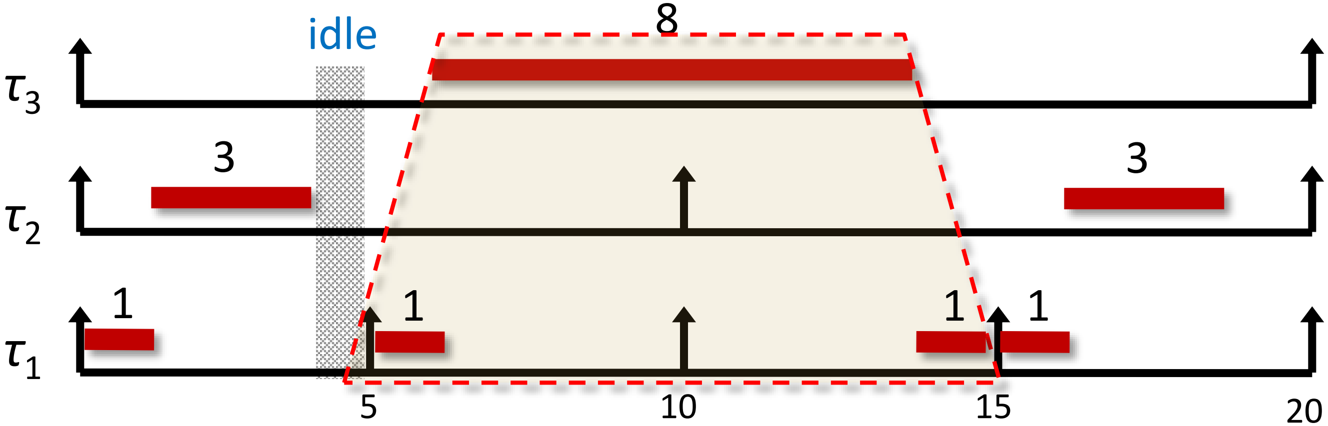
- Integer period ratio $K_i \geq 3$
- Constant period ratio $K = 2$
- Integer period ratio $k_i \geq 1$ and enough vacant intervals

Period Ratio

$$k_i = \frac{T_i}{T_{i-1}}$$


A Closer Look at the
Idea of Precautious-RM

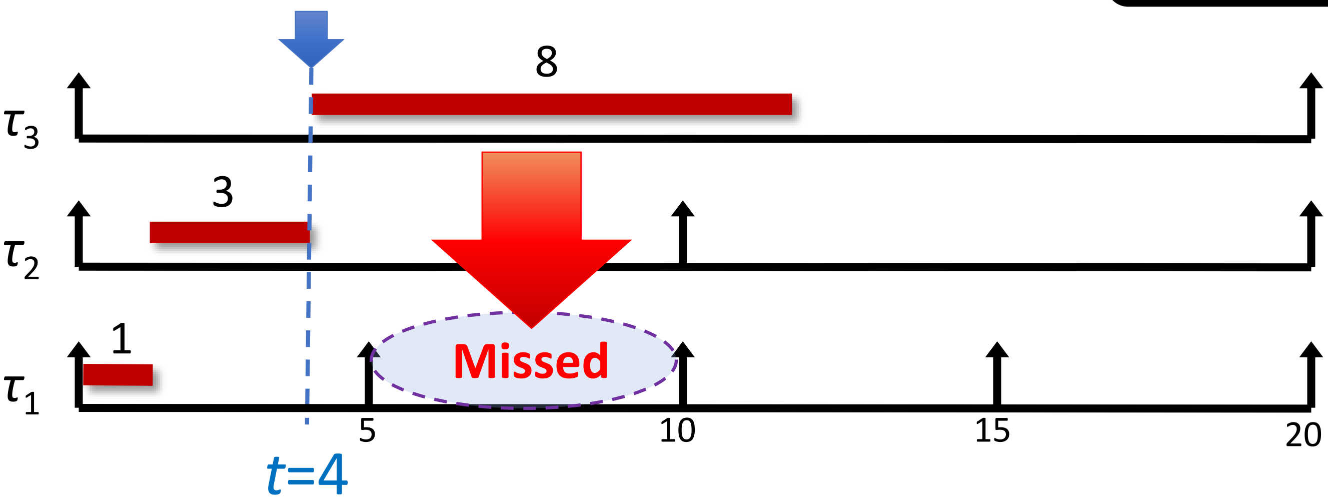
Precautious-RM Idea: An Efficient Decision



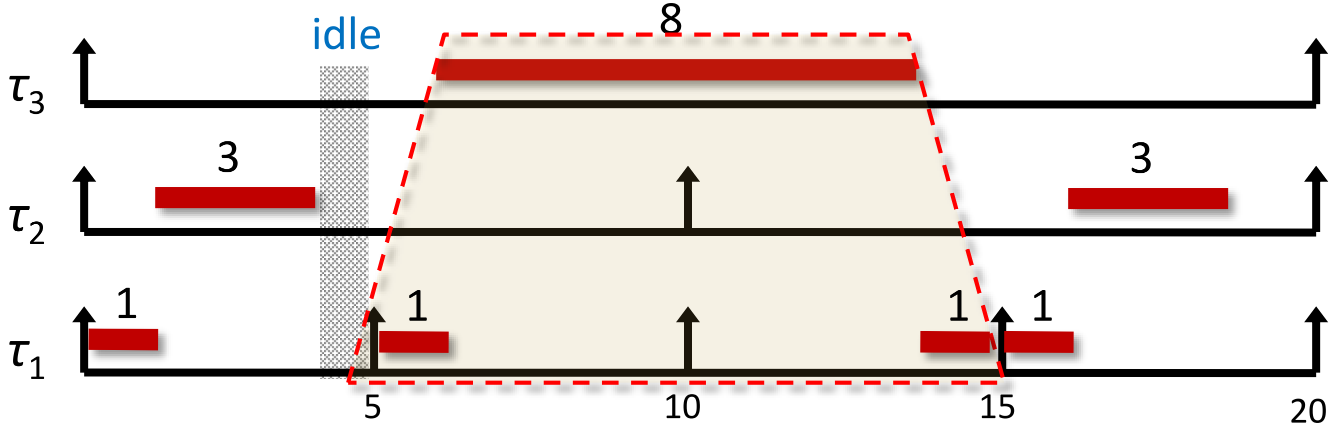
Feasible by a non-work conserving scheduling algorithm

$$c_i \leq 2(T_1 - c_1)$$

Necessary Condition



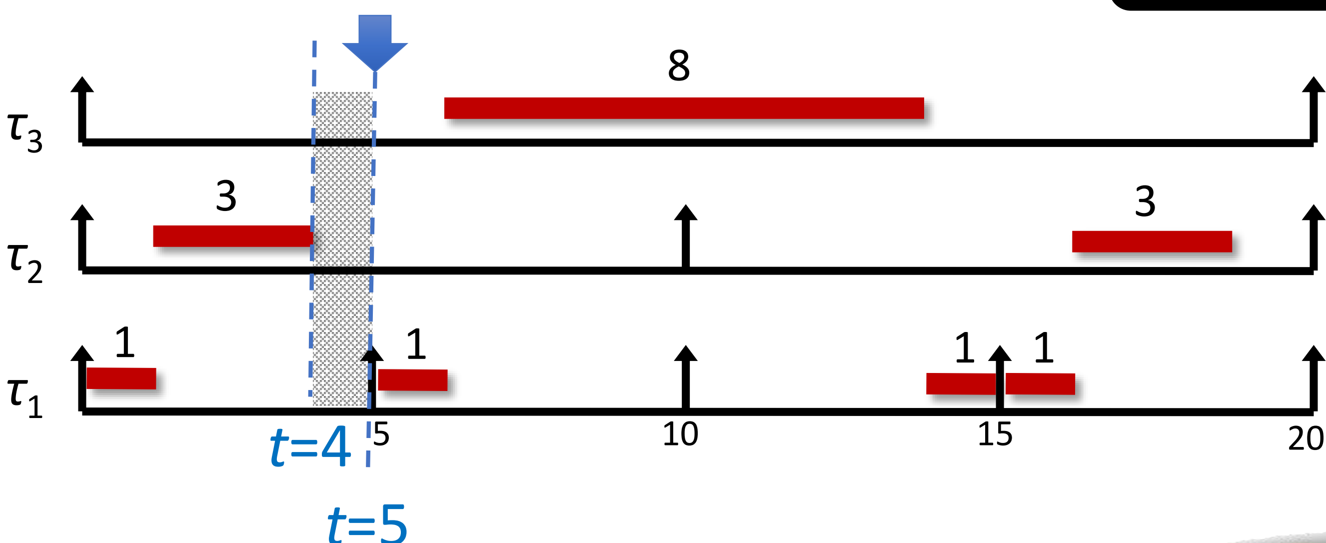
Precautious-RM Idea: An Efficient Decision



Feasible by a non-work conserving scheduling algorithm

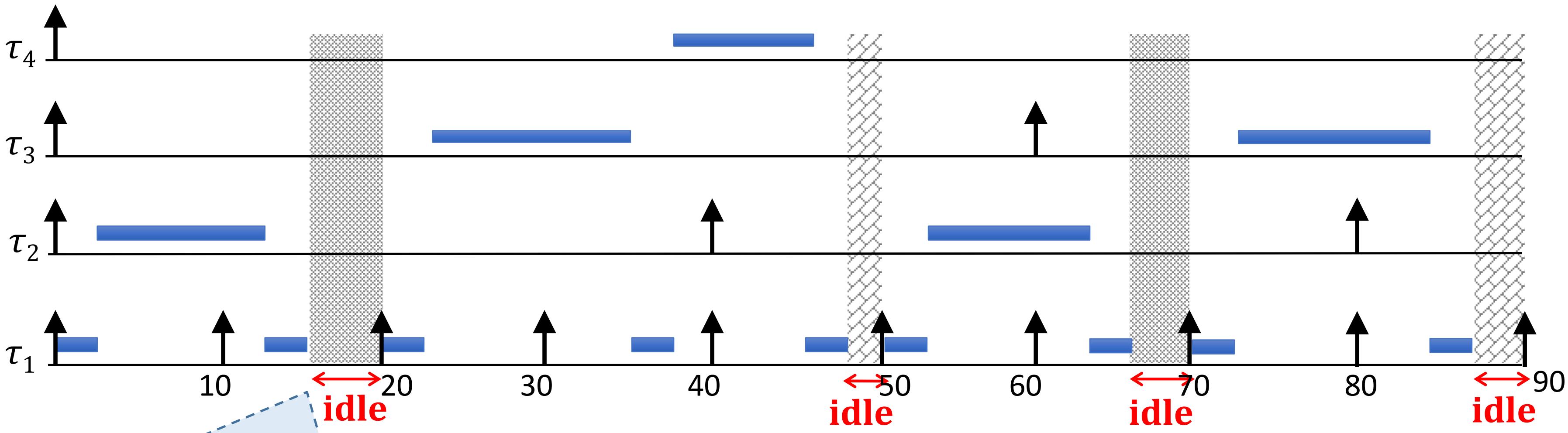
$$c_i \le 2(T_1 - c_1)$$

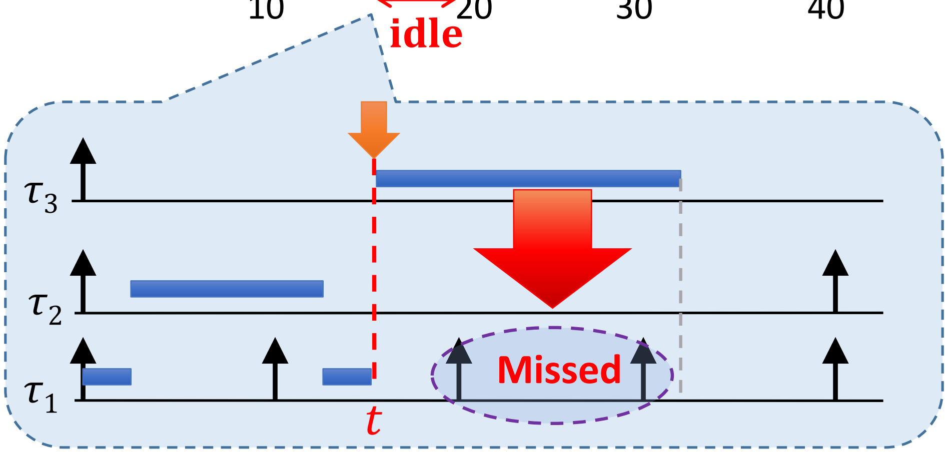
Necessary Condition



How Precautious-RM Works

- Rule 1: Use RM priorities (shorter periods have higher priority)
- Rule 2: Schedule a task only if it will not cause a deadline miss for the next instance of τ_1, otherwise, insert an idle interval until the next release of τ_1





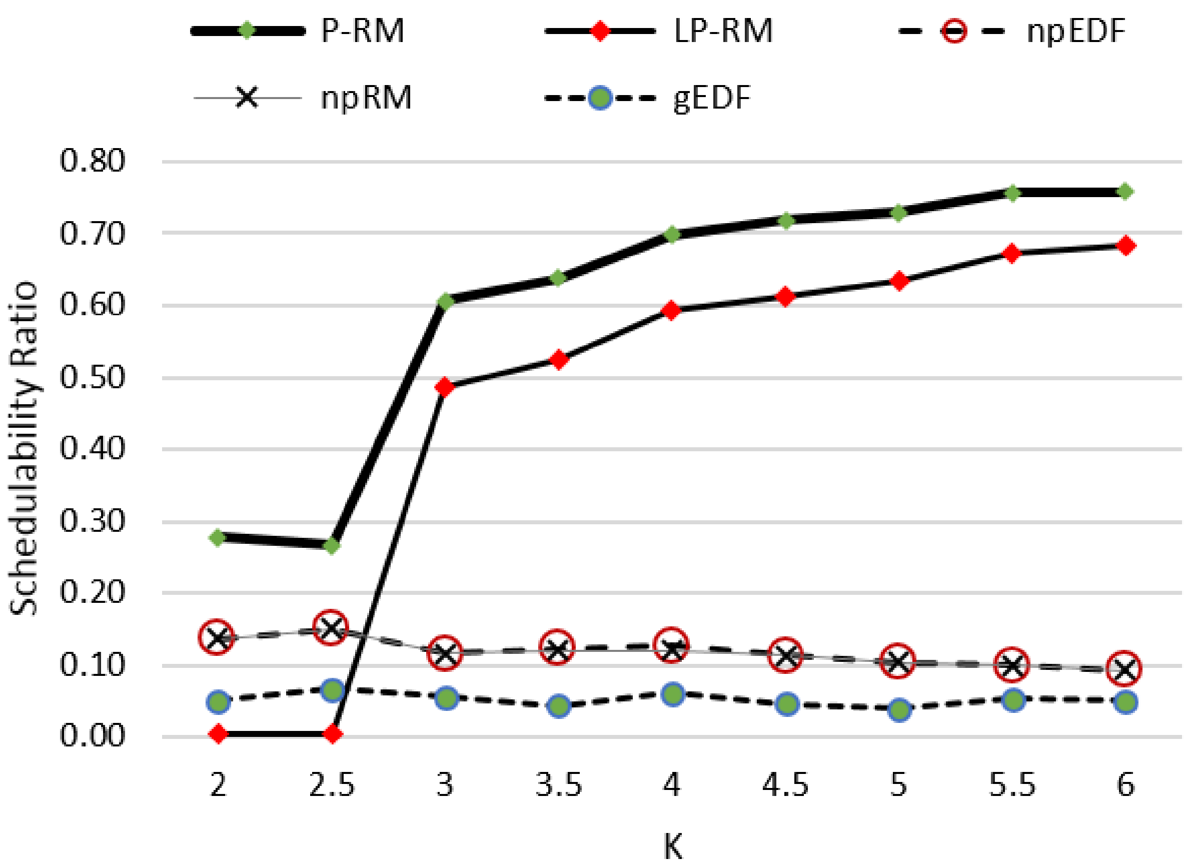
- Online algorithm (online decisions)
- $O(1)$ computational complexity

- **Limitations of the existing schedulability test**
  - It is only for harmonic tasks
  - It is pessimistic
    - it assumes each task has $c_i = 2(T_1 - c_1)$

Simple Idea, Interesting Results

Technische Universität Kaiserslautern

- How good is this idea?



$K = \max\{k_i\}$, where k_i is the individual period ratio in the task set

Simple Idea, Interesting Results

- How good is this idea?

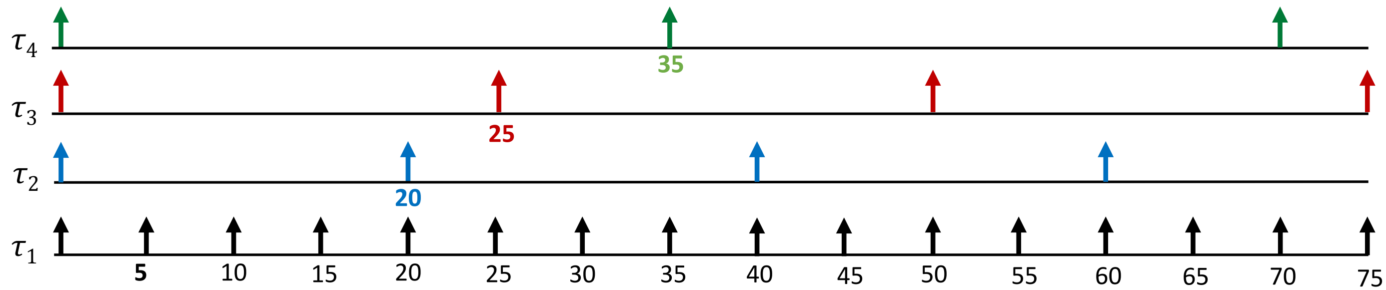
It is a big progress!

Contributions of This Work

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

- Extending the schedulability of **Precautious-RM** to **Loose Harmonic** tasks
  - Loose harmonic tasks:

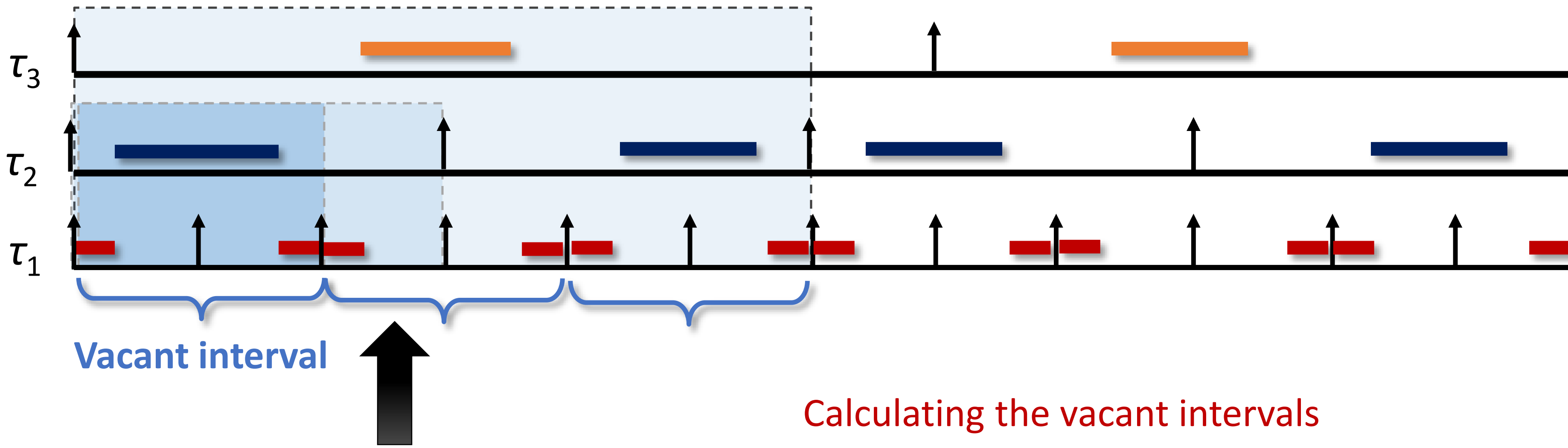
$$\frac{T_i}{T_1} \in \mathbb{N}$$



- Improving the schedulability by **priority grouping**
  - Tasks are assigned to priority groups and they are only allowed to be scheduled if the head of the group is scheduled

- Presenting a priority grouping algorithm which theoretically dominates schedulability test for Precautious-RM
  - The wise fit!

Precautious-RM Schedulability Test's Idea



$v_1 = 0.5$

$v_2 = 3×0.5 - 1 =$ **0.5**

$v_3 = 2×0.5 - 1 = 0$

Calculating the vacant intervals

$$v_i = \begin{cases} \lfloor k_i \rfloor v_{i-1} - 0.5, & c_i \leq T_1 - c_1 \ \textbf{\textit{and}}\ 1 < i \leq n \\ \lfloor k_i \rfloor v_{i-1} - 1, & c_i > T_1 - c_1 \ \textbf{\textit{and}}\ 1 < i \leq n \end{cases}$$

$$v_1 = 0.5$$

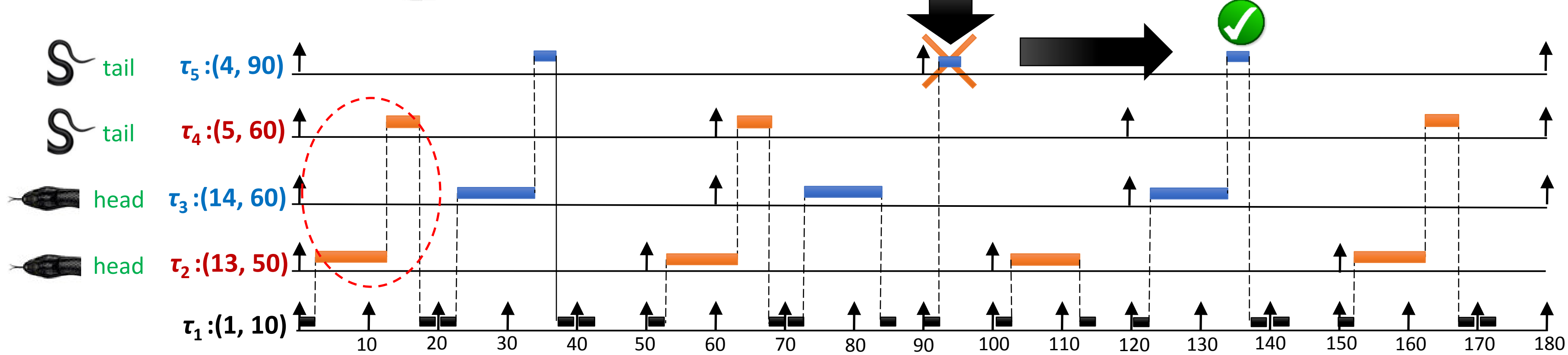
The schedulability test

$$v_i \geq 0.5 \qquad \forall i, 1 < i < n$$
$$v_n \geq 0$$

Next Improvement: Priority Grouping

- **Priority grouping**
  - It helps to improve the schedulability by wasting less vacant intervals

- **The restriction**



$$2(T_1 - c_1) = 18 \quad \begin{cases} \{\tau_1\} & \\ \{\tau_2 + \tau_4\} & c_2 + c_4 = 18 \\ \{\tau_3 + \tau_5\} & c_3 + c_5 = 18 \end{cases}$$

The solution

Permit the tail tasks to be executed only if the head task is scheduled in the same vacant interval.

Schedulability of the Priority Groups

- Each group has $C_i \leq 2(T_1 - c_1)$, thus we can use Precautious-RM schedulability.
  - We need $V_i \geq 0.5$

Easy proof for head tasks

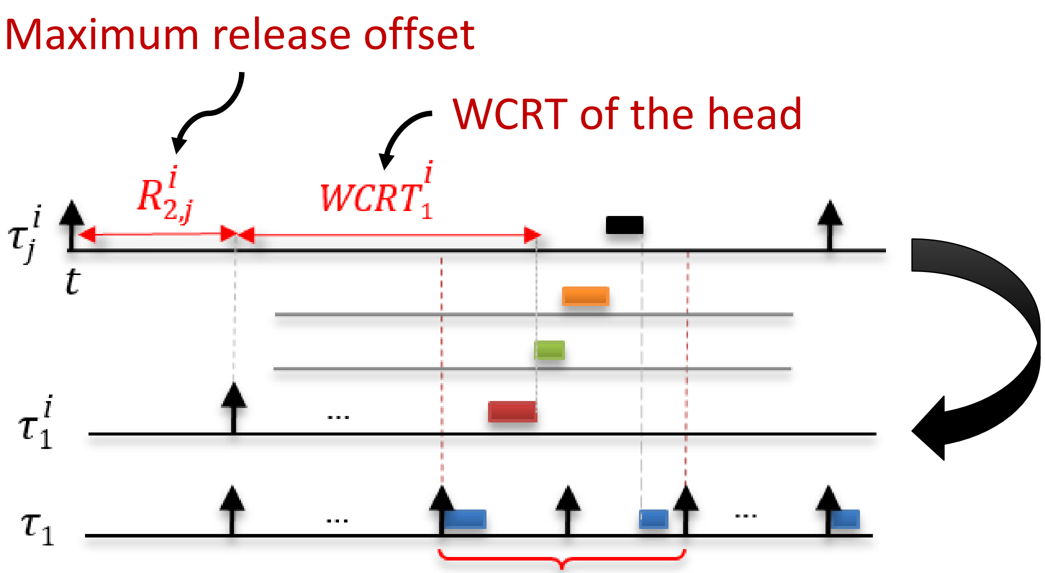
How can we guarantee schedulability of the tail tasks?

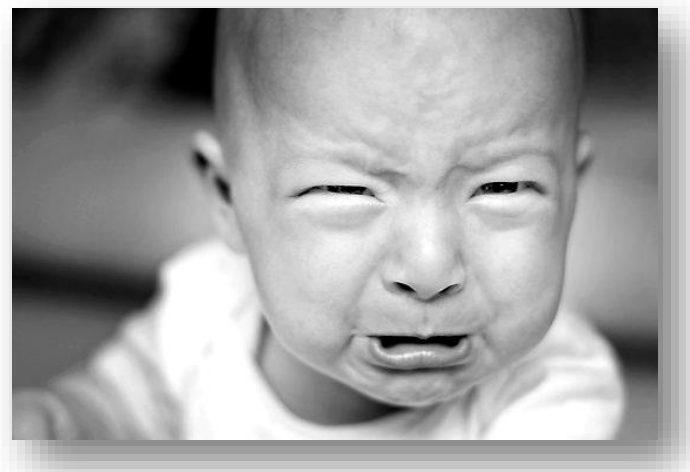
Schedulability of the Priority Groups, cont.

- How to guarantee the schedulability of the tail tasks?

WCRT analysis?

$$R^i_{1,j} + WCRT^i_1 + \sum_{x=2}^{X_i} c^i_x \le T^i_j$$

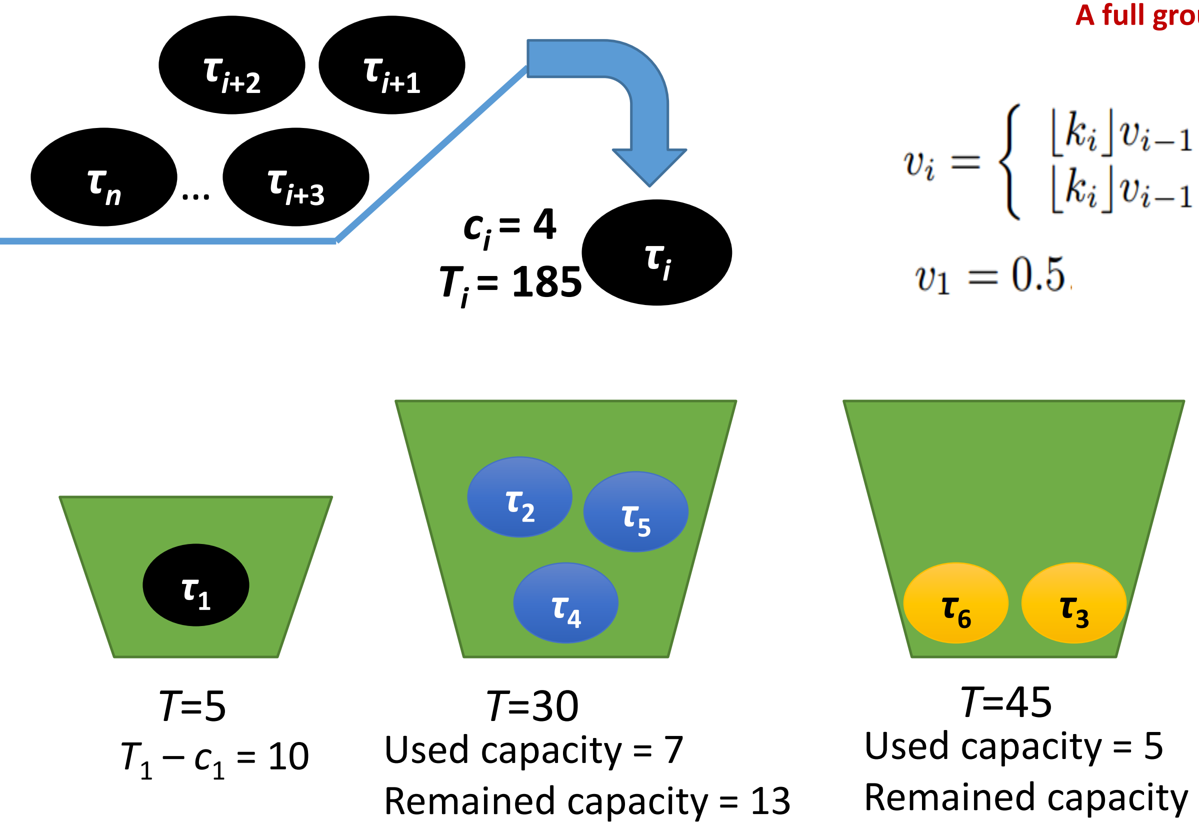




Period of each tail ≥ 2×T_{head}

The Wise-Fit Approach

- Wise-Fit
  - Picks the first ungrouped task
  - Finds the first group with enough capacity (based on the execution times)
  - Verifies the schedulability of the existing groups if this task is added to the group
  - If there is no such group, it creates a new group



A full group is the one with $C_i > (T_1 - c_1)$

$$v_i = \begin{cases} \lfloor k_i \rfloor v_{i-1} - 0.5, & c_i \leq T_1 - c_1 \text{ and } 1 < i \leq n \\ \lfloor k_i \rfloor v_{i-1} - 1, & c_i > T_1 - c_1 \text{ and } 1 < i \leq n \end{cases}$$

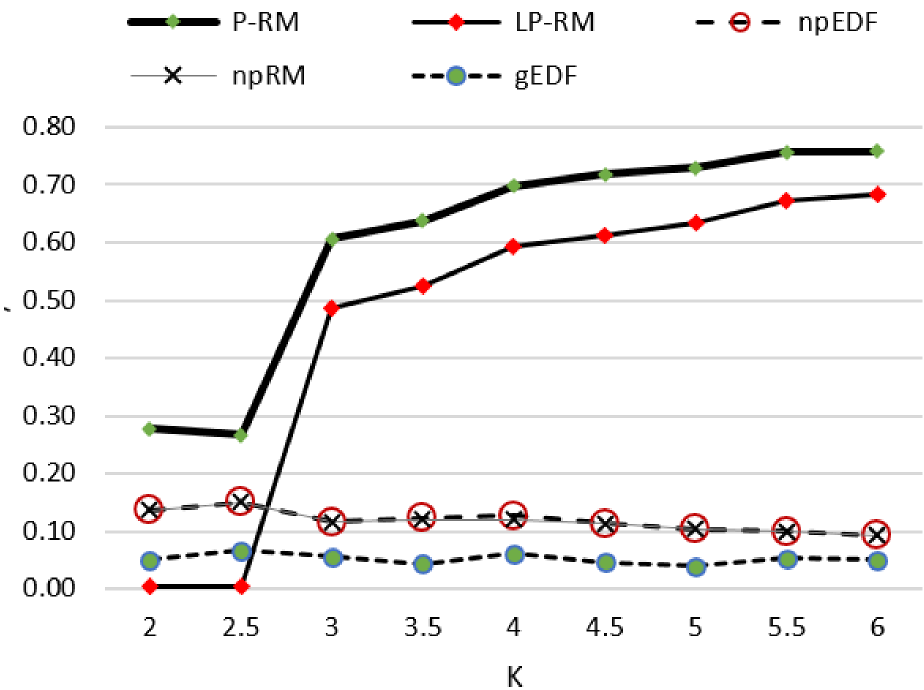
$$v_1 = 0.5$$

Sometimes it is better to leave a group half-empty instead of making it totally full.

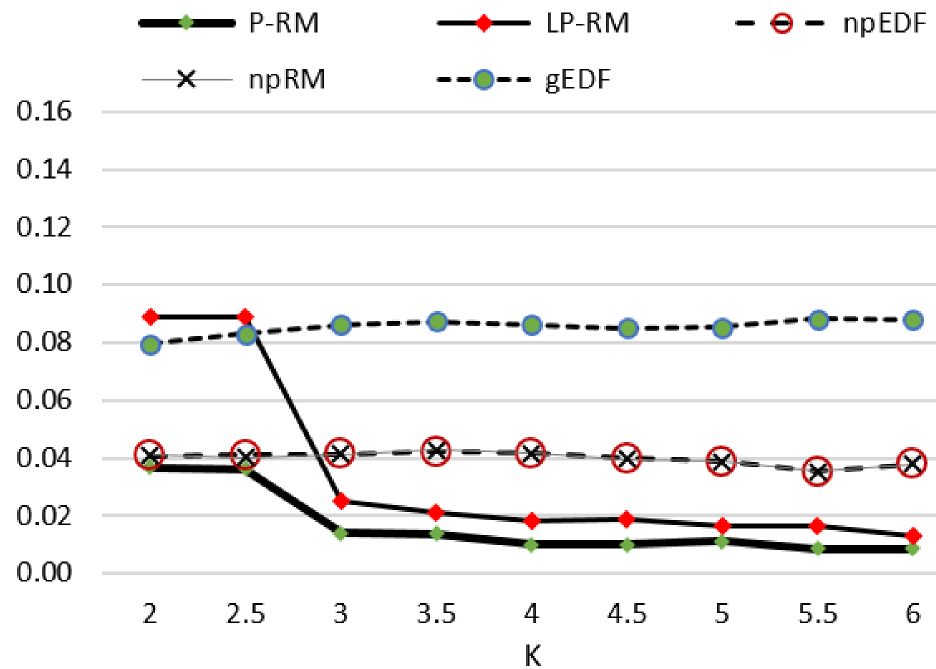
Evaluations

The Effect of Period Ratio

Schedulability Ratio

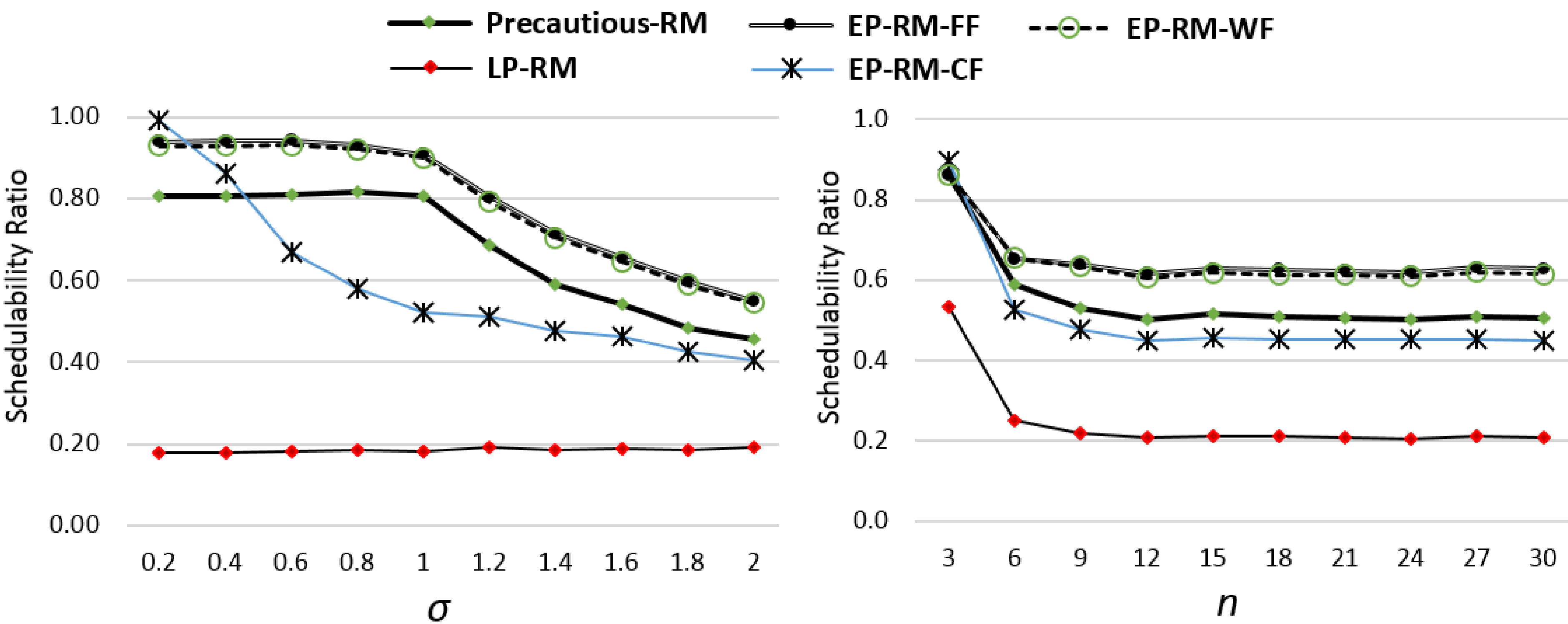


Job Miss Ratio



- $K = \max\{k_i\}$, where k_i is the individual period ratio in the task set
- Tasks with random execution time smaller than $2(T_1\text{-}c_1)$
- Not necessarily feasible task sets
- 7 tasks

The Effect of Other Parameters



- k_i is selected randomly from {1, 2, 3, 4}
- $c_i \leq \sigma \times 2(T_1\text{-}c_1)$
- Not necessarily feasible task sets
- 10 tasks

- k_i is selected randomly from {1, 2, 3, 4}
- $c_i \leq 2(T_1\text{-}c_1)$
- Not necessarily feasible task sets

Conclusions

Non-Preemptive Scheduling

- The only applicable solution in many systems
- Reduced overheads by avoiding preemptions
- More timing predictability for the tasks
- Necessary for many applications

A Non-Work Conserving Solution
(based on Precautious-RM)

- $O(1)$ online complexity
- $O(n)$ schedulability test
- High schedulability ratio

New Schedulability Test for Non-Harmonic Tasks

Improving the Schedulability by Priority Grouping

Future Work

Schedulability Analysis in General Case

- D < T
- Periodic tasks with no condition on periods

Appling Precautious-RM in Different Systems

- In CAN networks
- In real-time GPU applications

Extending it to multiprocessor systems

- Both partitioning and global approaches


We broke an old wall
Thank you

An Example

$\tau_1=(3, 10)$

$\tau_2=(2, 20)$

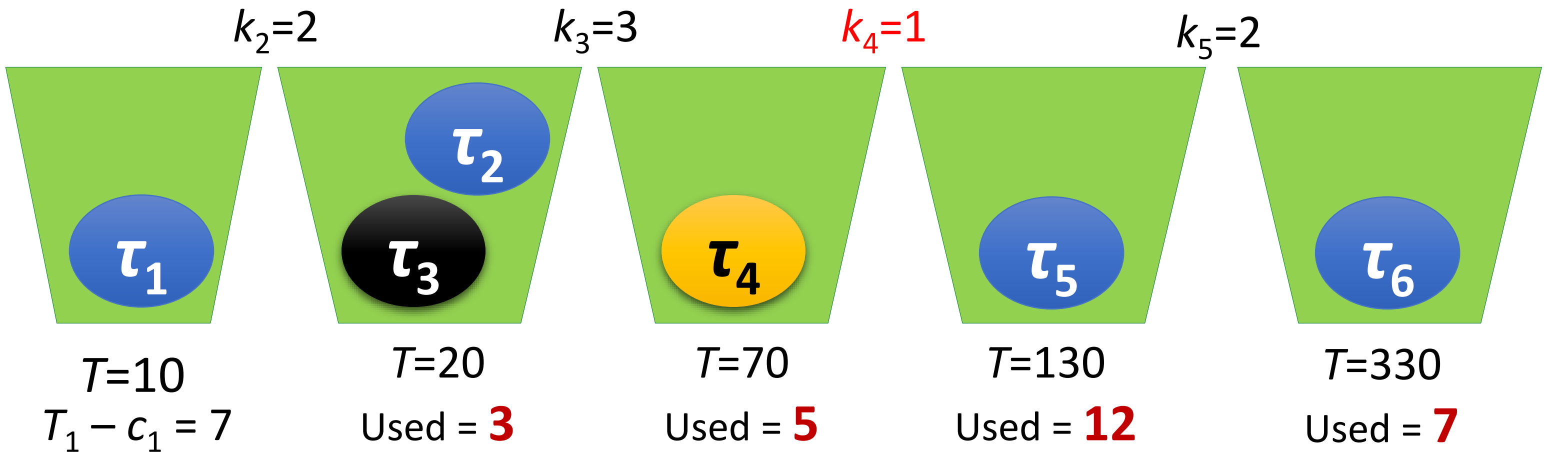
$\tau_3=(1, 40)$

$\tau_4=(5, 70)$

$\tau_5=(12, 130)$

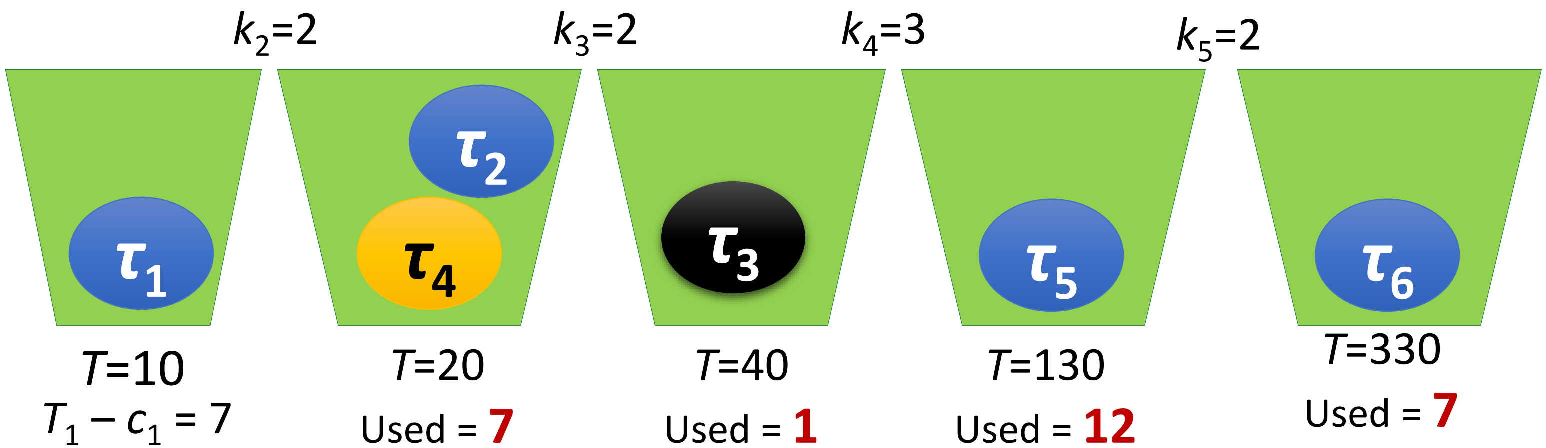
$\tau_6=(7, 330)$

- **First-Fit**



Rejected by the schedulability test

- **Wise-Fit**



Accepted by the test: Schedulable