RTAS 2019

Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems

Hyunjong Choi, Hyoseung Kim, Qi Zhu[†]

UC RIVERSIDE UNIVERSITY OF CALIFORNIA

RTEN Real-time Embedded and Networked Systems Laboratory

Northwestern University[†]

Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work

Outline

I **Introduction**

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work

Weakly-hard real-time systems

- Many practical systems
  - Tolerable to some deadline misses w/o affecting functional correctness

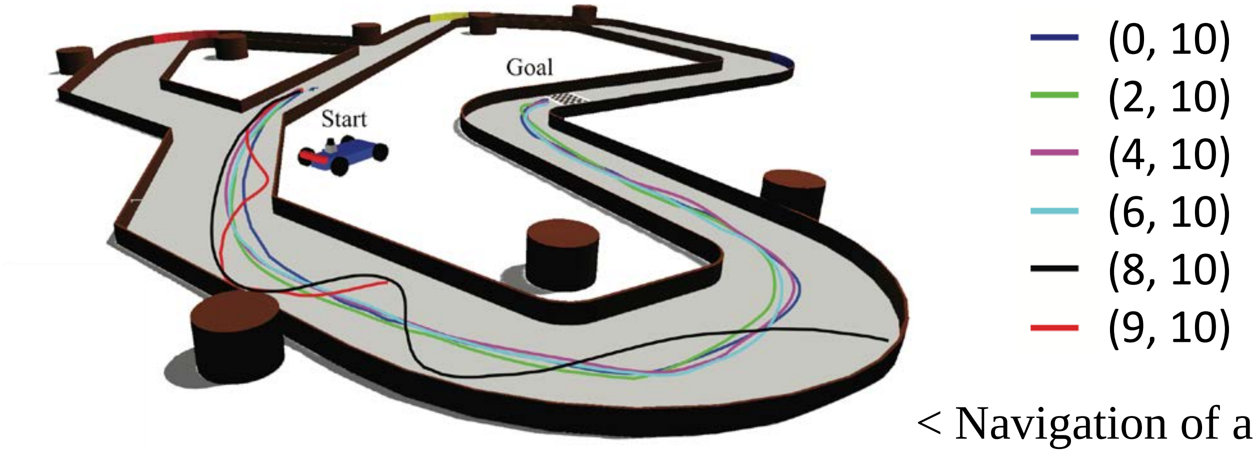
Weakly-hard real-time systems
to improve resource usage efficiency[‡]

$(\boldsymbol{m}, \boldsymbol{K})$: at most m jobs can miss their deadlines
among any K consecutive jobs

[‡] G. Bernat, A. Burns, and A. Liamosi, “Weakly hard real-time systems,” IEEE transactions on Computers, 2001

Effectiveness of weakly-hard real-time systems

- Navigation of an autonomous vehicle in *Gazebo with ROS*
  - A periodic task: *ControlTask*†
  - Mission: Drive from start to end points
  - Injected deadline misses w.r.t. weakly-hard constraints



< Navigation of an autonomous vehicle – *ControlTask* exp. >

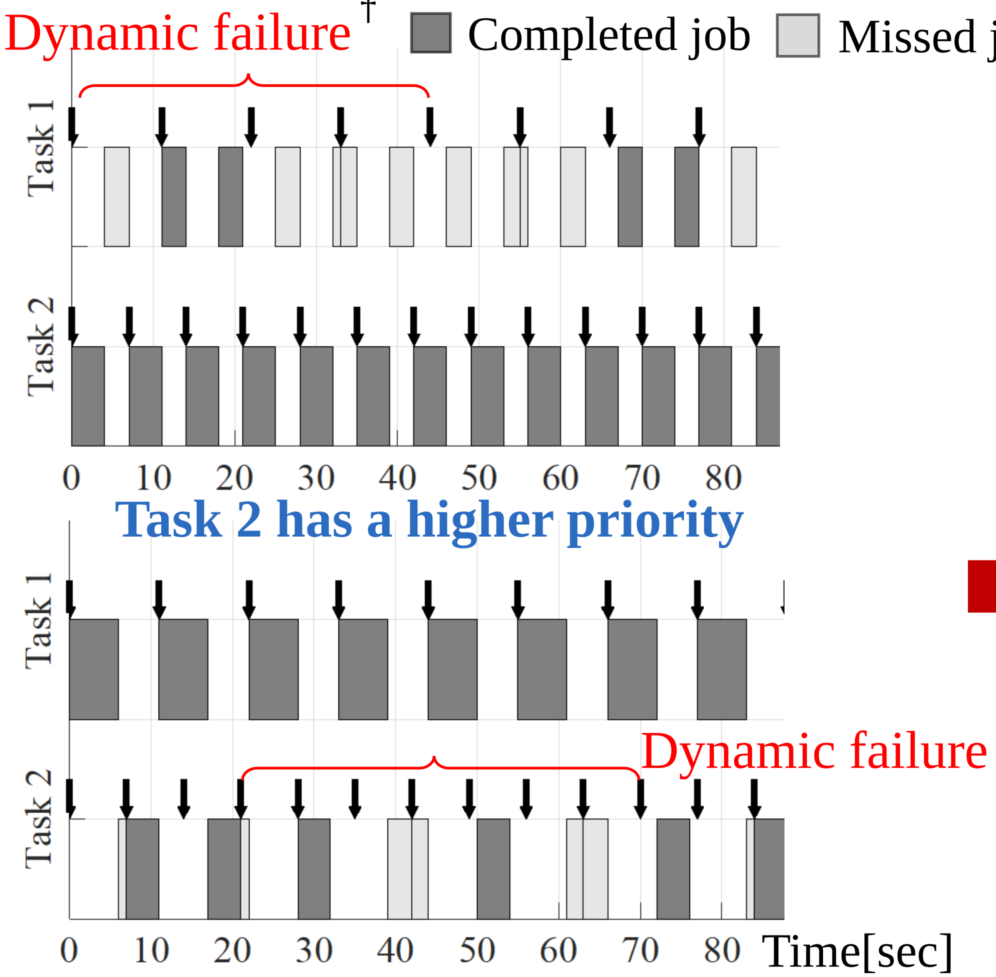
Tasks with bounded deadline misses can produce a functional correctness

Resource can be reserved for the other tasks

† It sends velocity command to robot base(actuator) at the specified rate defined as a control frequency.

Limitation of task-level fixed-priority scheduling

- Simple taskset with weakly-hard constraints



Task 2 has a higher priority

Task 1 has a higher priority

	Specifications
Task 1	$T_1 = 11, C_1 = 6, m_1 = 2, K_1 = 4$
Task 2	$T_2 = 7, C_2 = 4, m_2 = 4, K_2 = 7$

< A taskset example >

No matter which task has a higher priority, NOT schedulable !

New approach

[†] Task experiences more than *m* deadline misses in a window of *K* jobs.

Contributions

- Main contributions
  - Propose **a new job-class-level fixed-priority scheduler** based on ***meet-oriented classification*** of jobs of tasks
  - Present the **schedulability analysis framework** for our proposed scheduler
  - Generalization of task-level fixed-priority scheduling
  - Outperforms the latest work in terms of **task schedulability**, **analysis running time**
  - **Implement our scheduler in the Linux kernel** running on Raspberry Pi

Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work

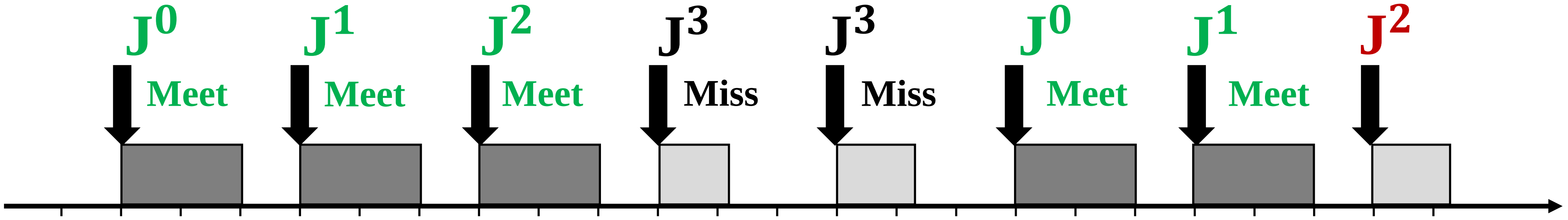
System Model

- Task model
  - $\tau_i \coloneqq (C_i, D_i, T_i, (m_i, K_i))$
    - ✓ C_i: The worse-case execution time
    - ✓ D_i: The relative deadline
    - ✓ T_i: The minimum inter-arrival time
    - ✓ $(m_i,\ K_i)$: The weakly-hard constraints ($m_i < K_i$). For a hard real-time task, $m_i = 0$ and $K_i = 1$.
- Preemptive scheduling
- Uniprocessor system

Job-Class-Level Fixed-Priority Scheduling

- Job classification
  - Assign different priorities to individual job-classes

Meet-oriented : the number of prior deadlines consecutively met



- For instance, (m, K) = (2, 4) can have job classes: J^0, J^1, and J^2
- Priority of a job-class decrease monotonically

Bounding consecutive deadline misses

- Miss threshold w_i
  - Limit the distance from the current job to the previous deadline-met jobs to bound the number of consecutive deadline misses

$$w_i = \max\left(\left\lfloor\frac{K_i}{K_i - m_i}\right\rfloor - 1, 1\right)$$

  - Ensure enough number of jobs running with the highest priority job-class
  - For instance, $(m, K) = (5, 7)$ where $w_i = 2$ allows 2 consecutive deadline misses

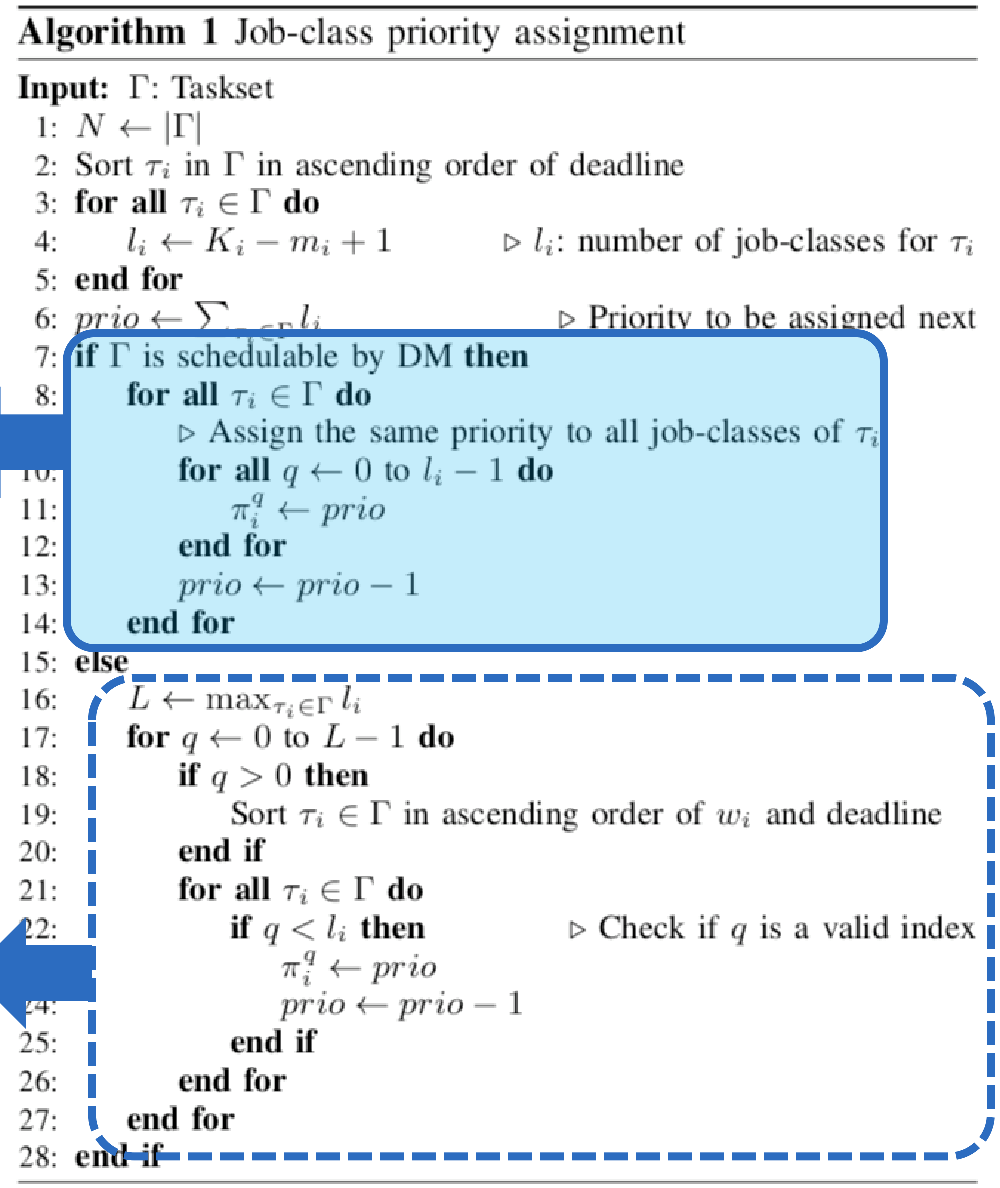
Priority assignment

- A heuristic priority assignment
  - An **extension of the deadline monotonic (DM)** priority assignment

Lemma 3.

Subsumes the task-level DM priority assignment

  - Rule.
    - ✓ Assign higher priority to a job-class with a smaller index
    - ✓ For job-classes with the same index,
      - Higher priority to shorter deadline (q = 0)
      - Higher priority to shorter miss threshold with deadlines for tie-breaking (q > 0)



Algorithm 1 Job-class priority assignment

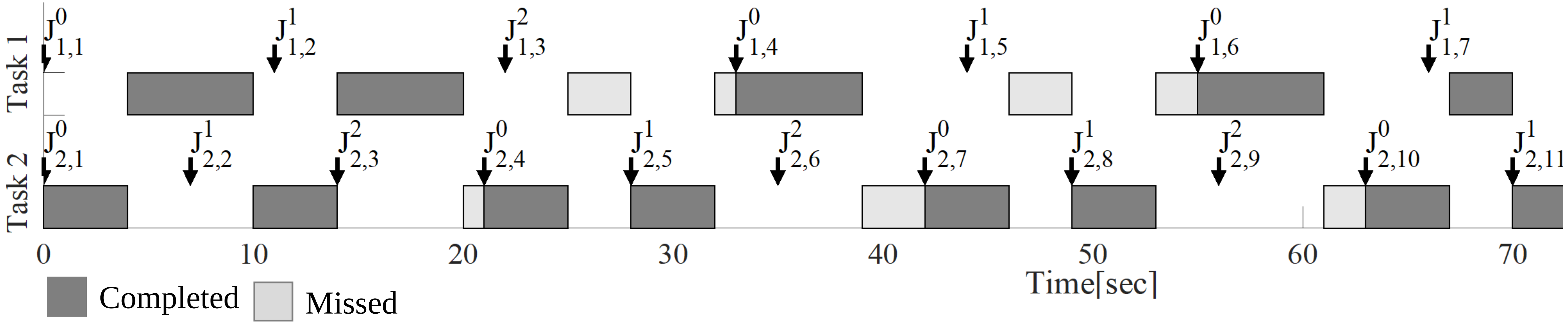
```
Input: $\Gamma$: Taskset
$N \leftarrow |\Gamma|$
Sort $\tau_i$ in $\Gamma$ in ascending order of deadline
for all $\tau_i \in \Gamma$ do
    $l_i \leftarrow K_i - m_i + 1$            ▷ $l_i$: number of job-classes for $\tau_i$
end for
$prio \leftarrow \sum_{\tau_i \in \Gamma} l_i$            ▷ Priority to be assigned next
if $\Gamma$ is schedulable by DM then
    for all $\tau_i \in \Gamma$ do
        ▷ Assign the same priority to all job-classes of $\tau_i$
        for all $q \leftarrow 0$ to $l_i - 1$ do
            $\pi_i^q \leftarrow prio$
        end for
        $prio \leftarrow prio - 1$
    end for
else
    $L \leftarrow \max_{\tau_i \in \Gamma} l_i$
    for $q \leftarrow 0$ to $L - 1$ do
        if $q > 0$ then
            Sort $\tau_i \in \Gamma$ in ascending order of $w_i$ and deadline
        end if
        for all $\tau_i \in \Gamma$ do
            if $q < l_i$ then            ▷ Check if $q$ is a valid index
                $\pi_i^q \leftarrow prio$
                $prio \leftarrow prio - 1$
            end if
        end for
    end for
end if
```

An example of job-class-level scheduling

- With the same taskset at page 6.

	Specifications
Task 1	$T_1 = 11, C_1 = 6, m_1 = 2, K_1 = 4$
Task 2	$T_2 = 7, C_2 = 4, m_2 = 4, K_2 = 7$

< A taskset example >



Schedulable !!

Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

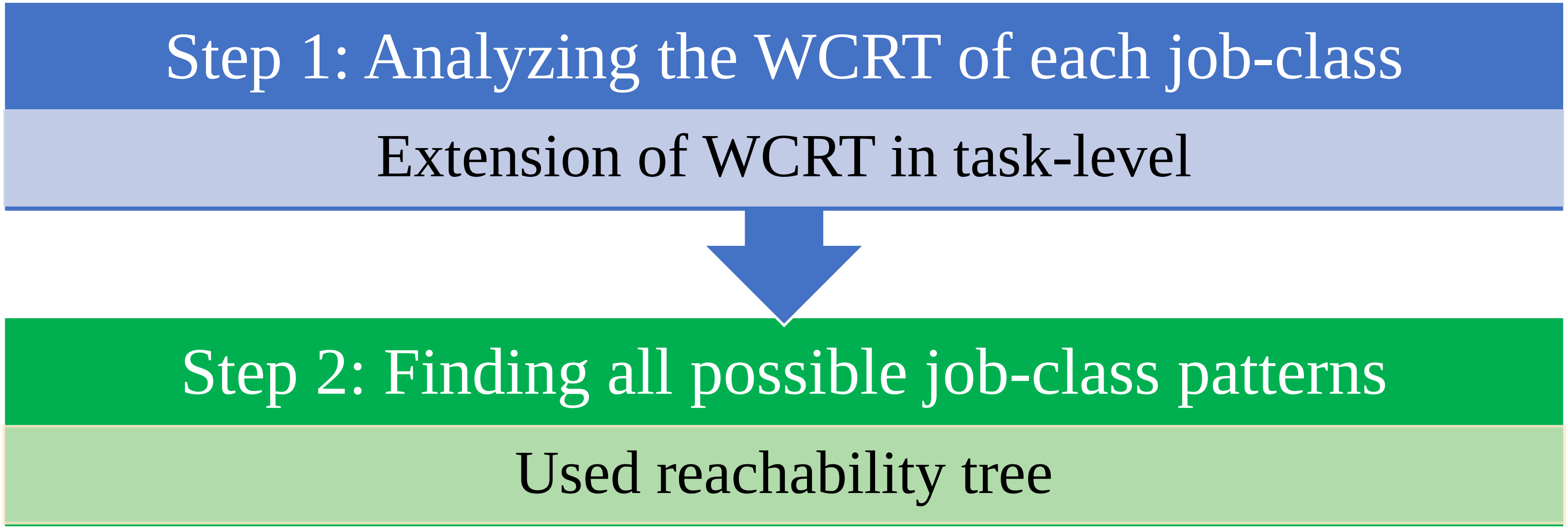
IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work

Schedulability Analysis

- The schedulability analysis of tasks with weakly-hard constraints under job-class-level scheduling decompose



Step 1: Analyzing the WCRT of each job-class

Extension of WCRT in task-level

Step 2: Finding all possible job-class patterns

Used reachability tree

< Schedulability analysis process of job-class-level scheduler >

Worse-case response time of job-classes

- Worse-case response time of J_i^q is bounded by the recurrence:

$$R_i^{q,n+1} \leftarrow C_i + \sum_{\tau_k \in \Gamma - \tau_i} W_i^q(R_i^{q,n}, \tau_k)$$

✓ W_i^q is an upper-bound of interference imposed on J_i^q

$$W_i^q(t, \tau_k) = \min\left(\sum_{\forall\, p:\pi_i^q < \pi_k^p} \left\lceil \frac{t + J_k}{\eta(J_k^p)} \right\rceil \times C_k \,, \left\lceil \frac{t + J_k}{T_k} \right\rceil \cdot C_k \right)$$

✓ Each job-class has a *different* *minimum job-class inter-arrival time*, $\eta(J_k^p)$

Lemma 8.

Generalization of the task-level iterative response time test for hard real-time tasks.

[†] M. Josephand P. Pandya, "Finding response times in a real-time system," The Computer Journal, 1986.

Schedulability check

- Schedulability test of a task with $m_i/K_i \geq 0.5$

Lemma 10.

*A task τ_i is **always schedulable** if the **ratio of m_i/K_i is greater than or equal to 0.5** and it **satisfies the prerequisite** given by Lemma 9.*

Step 1: Show at least 1 deadline met in K_i window by using a **<u>necessary condition</u>**

$$(w_i + 1) \cdot \alpha \leq K_i \qquad \mathrm{WCRT}(\mathrm{J}_i^0) \leq D_i$$

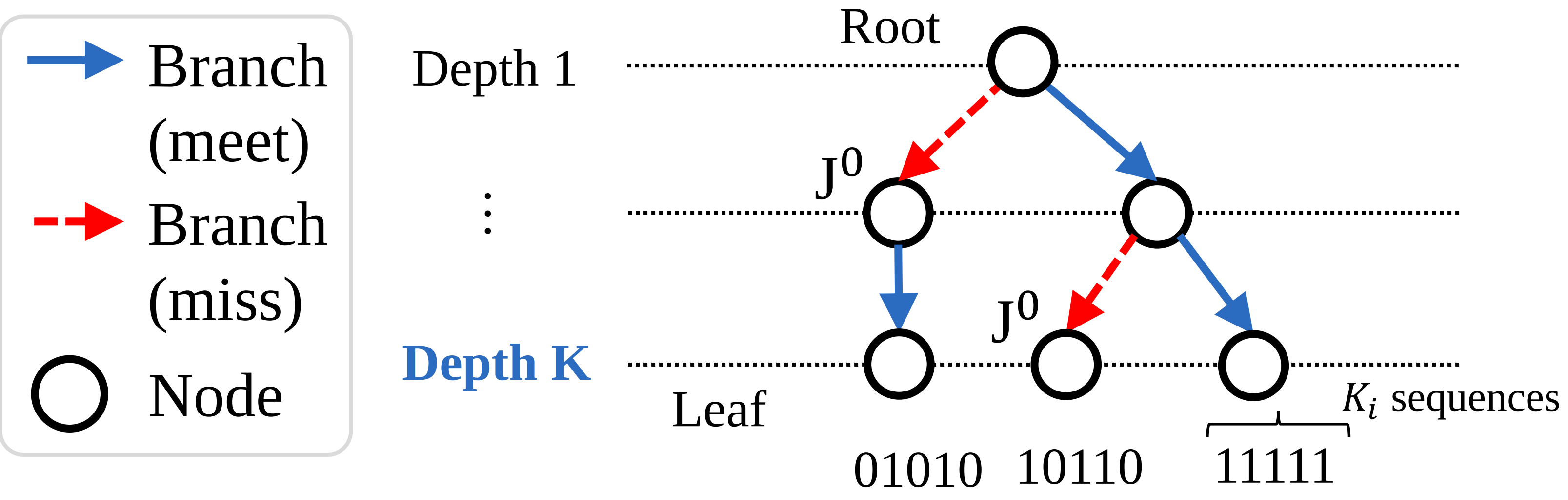
Step 2: Show that the number of deadline met satisfies the constraint

$$\frac{1}{w_i+1} \geq \frac{K_i-m_i}{K_i} \Longrightarrow \left\lfloor \frac{K_i}{K_i-m_i} \right\rfloor \leq \frac{K_i}{K_i-m_i}$$

Always true as $m_i \leq K_i - 1$

Reachability tree

- For tasks with $m_i/K_i < 0.5$, find all possible job-class patterns for K_i job executions using ***reachability tree***



< Reachability tree >

Lemma 13.

The reachability trees of a task τ_i ***represent all possible job-class patterns*** *that the task can experience at its runtime for K_i* execution window

Outline

I Introduction

II Related Work & Motivation

III Job-class-level Scheduling

IV Schedulability Analysis

V Evaluation

VI Conclusion and Future Work

Implementation cost

- Measure runtime overhead of the proposed scheduler implementation
- Experimental setup
  - Linux kernel v4.9.80 running on Raspberry Pi 3
  - ARM Cortex-A53 @ clock frequency of 1.2 GHz
  - Run 5 tasks with period of 20ms to 40ms for 10 minutes (118,569 jobs)

Type		Mean	Max	Min	99%th
Updating μ-pattern[†]		0.3002	1.1460	0.1040	0.6250
Updating job-class index		1.5035	11.8750	0.5210	2.5000
Changing task priority		4.7633	28.9580	3.0210	11.3020
Rollback	Checkpointing	1.9413	9.3230	1.2500	3.2290
	Recovery	6.1257	24.8430	0.4680	8.3146

< Runtime overhead [μs] >

[†] Represents a sequence of deadline met and missed jobs of a task, (G. Bernat, A. Burns, and A. Liamosi. “Weakly hard real-time systems”, 2001)

Schedulability experiments

- The evaluation is conducted in two ways:
  - **Comparison** with other weakly-hard scheduling schemes (WSA[†], RTO-RM[*])
    - ✓ WSA: delayed completion for deadline-missed jobs
    - ✓ RTO-RM: job abort for deadline-missed jobs
  - **Exploration** of the proposed scheduler under diverse experimental conditions
  - Performance metric : *percentage of schedulable taskset*, *analysis running time*

- Taskset generation

	Number of tasksets	Task utilization (UUniFast algorithm[#])	Task period [ms]	K range
Value	1,000	[0.8, 1.8]	[10, 1000]	{5, 10, 15}

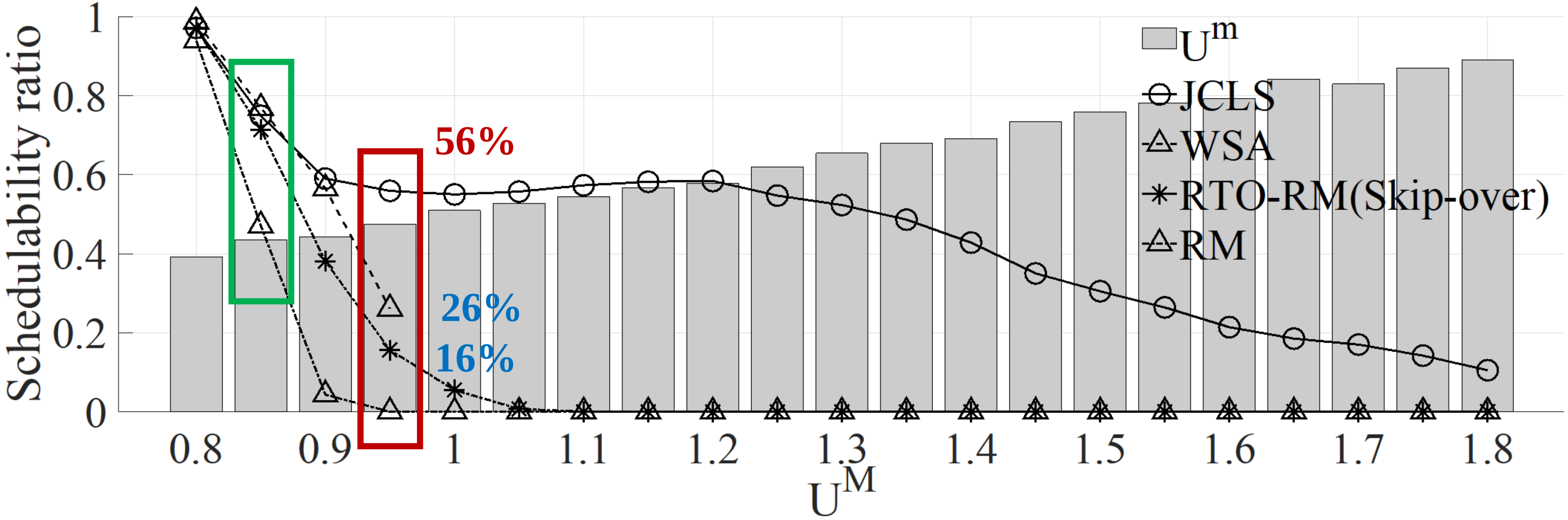
[†] Y. Sun and M. D. Natale, “Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks,” *TECS*, 2017

[*] G. Koren and D. Shasha. “Skip-over: Algorithms and complexity for overloaded systems that allow skips”, RTSS, 1995

[#] E.Biniand G.C.Buttazzo. “Measuring the performance of schedulability tests”, Real-Time Systems, 2005

Taskset schedulability

- Comparison of schedulability ratio with other schemes
  - 1,000 tasksets with 20 tasks
  - $K_i = 10, m_i = [1, 9]$, common (m, K) for a taskset



JCLS better utilizes CPU resource when there are overloaded weakly-hard tasksets

Analysis running time

- Time to determine the schedulability of a given taskset
  - By the number of tasks in a taskset (10, 30, and 50 tasks)
  - 1,000 tasksets, $K_i = 10, m_i = [1, 9]$
  - JCLS (on Raspberry Pi 3), WSA (on Intel Core-i7 for CPLEX Optimizer)

Number of tasks	Approach	Mean	Max
10	JCLS	0.0010	0.0046
	WSA	0.2739	114.2892
30	JCLS	0.0112	0.0432
	WSA	25.7284	1800.5996
50	JCLS	0.0331	0.1463
	WSA	78.5982	3002.5189

< Analysis running time [sec] >

The analysis time of JCLS is shorter than that of WSA
More applicable to runtime admission control

Conclusion & Future work

- Conclusion
  - **New job-class-level fixed-priority scheduling** and **analysis** for weakly-hard real-time systems
  - Proposed scheduler **outperforms prior work** with respect to taskset schedulability and analytical complexity
  - Proposed approach is effective in overloaded situations (e.g., maximum utilization is higher than 1)

- Future work
  - Address the pessimism of our schedulability analysis when the ratio of m_i/K_i is less than 0.5

Thank you

Job-Class-Level Fixed Priority Scheduling of Weakly-Hard Real-Time Systems

Hyunjong Choi, Hyoseung Kim, Qi Zhu

Q & A

Appendix

1. Related work
2. Utilizations
3. Benefits of the meet-oriented classification
4. Minimum time interval of a job-class
5. Interference of job-class-level analysis
6. Schedulability check
7. Complexity of reachability tree
8. An example of reachability tree

Related work

- Goals in weakly-hard systems : **guarantee** & **improve schedulability**
  - Scheduling: task-level fixed-priority scheduling
  - Assumptions : *initial offset is known*[†]*, periodic task*[◊] *with no jitter*[‡]

➡ **Limits applicability to recent cyber physical systems**

- [†, ‡]Bernat et al. works on the schedulability of periodic tasks with weakly-hard constraints under fixed-priority scheduling (RTSS’2001)
- Typical worst-case analysis (TWCA) approaches significantly contributes to weakly-hard systems (DATE’2012, DATE’2013, EMSOFT’2014, ECRTS’2015)
  - **Assume exact arrival patterns of task instances is known**
- [◊]Sun et al. relaxed the assumption on offset and jitter (TECS’2017)
- [†,◊, ‡] Goossens et al. distanced-based dynamic-priority scheduling (RTNS’2008)

Utilizations

- Represent the resource usage
  - Maximum utilization

Definition 1.

Maximum utilization of a task τ_i, U_i^M, is the maximum amount of CPU resource that τ_i can utilize, defined as $U_i^M = \frac{C_i}{T_i}$

*Maximum total utilization: $U^M = \sum_{i=1}^{N} C_i/T_i$

  - Minimum utilization

Definition 2.

Minimum utilization of a task τ_i, U_i^m, is the CPU resource used by τ_i when it experiences the maximum deadline misses allowed by its (m_i, K_i) constraint, i.e., $U_i^m = \frac{C_i}{T_i} \times \frac{K_i - m_i}{K_i}$

*Minimum total utilization: $U^m = \sum_{i=1}^{N} \frac{C_i}{T_i} \times \frac{K_i - m_i}{K_i}$

Benefits of the *meet-oriented* classification

- Benefits of meet-oriented classification
  - It reduces interferences imposed by higher priority jobs by modulating consecutive meets
  - Enables to avoid a pessimism when we evaluate WCRT of a job.

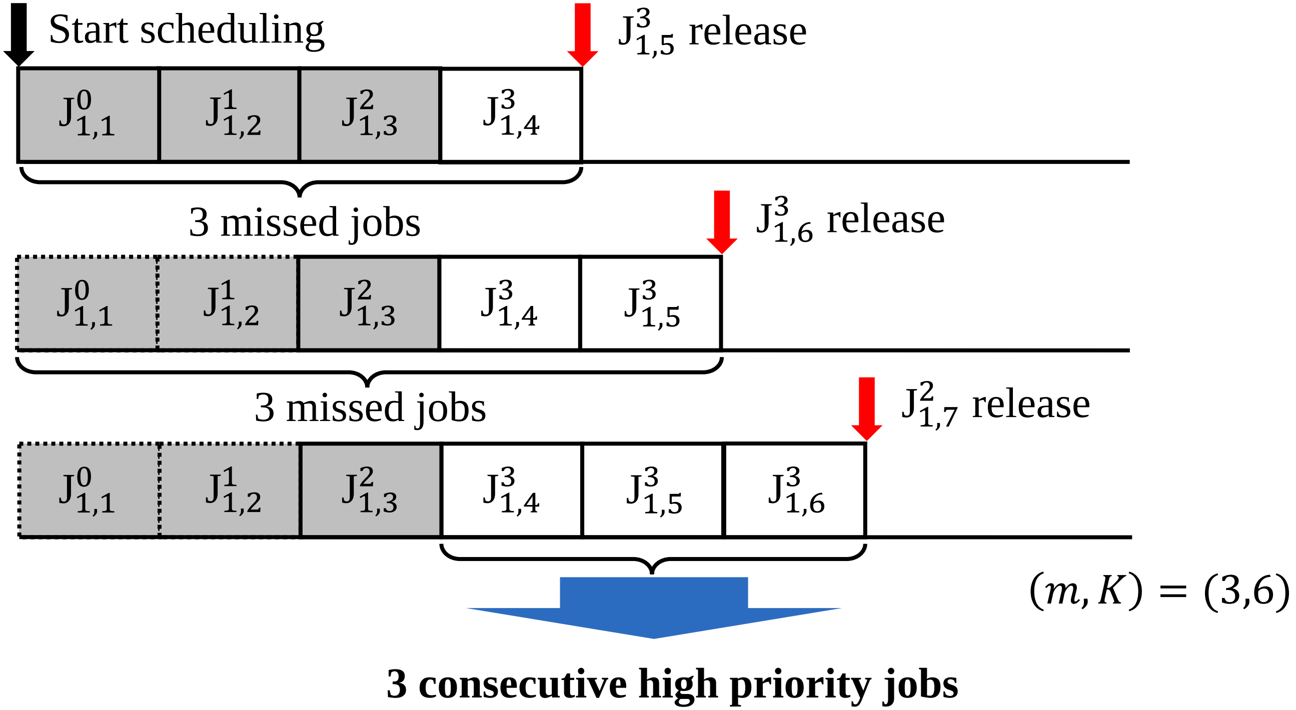
Job-classes	J^0	J^1	J^2	J^3
Meet/Miss	△	△	△	○
Priorities	1	3	5	7

Low ⟷ High

< After scheduling >

△ May miss or meet ▨ Deadline missed

○ Always meet □ Deadline met

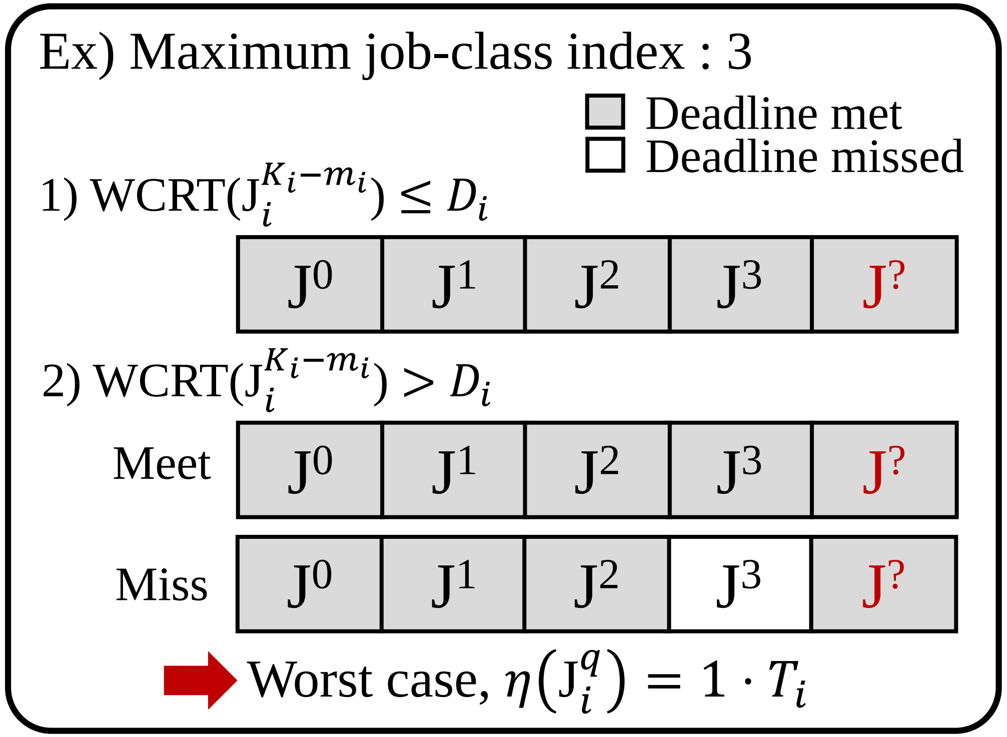


< Consecutive execution of high-priority jobs under *miss-oriented* job classification >

Minimum job-class inter-arrival time (1/4)

- As a first step, analyzing the WCRT of individual job-classes
- Upper bound the maximum interference imposed by the jobs of other tasks with higher-priority job-classes

$q = K_i - m_i$	• $\eta(\mathrm{J}_i^q) = 1 \cdot T_i$
$q < K_i - m_i$ & WCRT(J_i^q) $> D_i$	• $\eta(\mathrm{J}_i^q) = (q+1) \cdot T_i$, if $w_i = 1$ • $\eta(\mathrm{J}_i^q) = 1 \cdot T_i$, if $w_i > 1$
$q < K_i - m_i$ & WCRT(J_i^q) $\leq D_i$	• $\eta(\mathrm{J}_i^q) = (w_i + 1) \cdot T_i$, if $q = 0$ • $\eta(\mathrm{J}_i^q) = (q+2) \cdot T_i$, if $q > 0$



Ex) Maximum job-class index : 3

▨ Deadline met
□ Deadline missed

1) WCRT($\mathrm{J}_i^{K_i - m_i}$) $\leq D_i$

▨	▨	▨	▨	▨
J^0	J^1	J^2	J^3	$\mathrm{J}^?$

2) WCRT($\mathrm{J}_i^{K_i - m_i}$) $> D_i$

Meet	▨ J^0	▨ J^1	▨ J^2	▨ J^3	▨ $\mathrm{J}^?$
Miss	▨ J^0	▨ J^1	▨ J^2	□ J^3	▨ $\mathrm{J}^?$

Worst case, $\eta(\mathrm{J}_i^q) = 1 \cdot T_i$

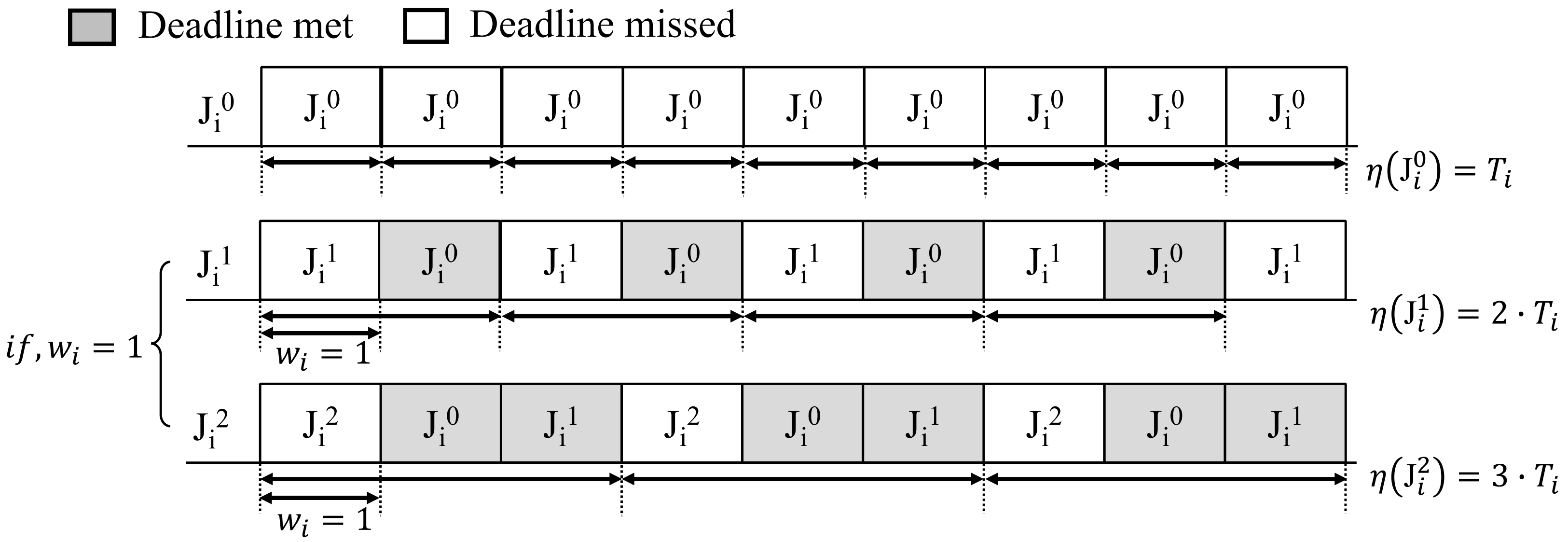
Minimum time interval of a job-class (2/4)

- A job-class whose the WCRT > D_i,

Lemma 5.

The minimum inter-arrival time of J_i^q where $q < K_i - m_i$ and the WCRT of J_i^q is greater than D_i is given by

$$\eta\left(J_i^q\right) = \begin{cases} (q+1) \cdot T_i, if\ w_i = 1 \\ 1 \cdot T_i, if\ w_i > 1 \end{cases}$$



▨ Deadline met
□ Deadline missed
J_i^0
J_i^0 J_i^0 J_i^0 J_i^0 J_i^0 J_i^0 J_i^0 J_i^0 J_i^0
$\eta(J_i^0) = T_i$
J_i^1
J_i^1 J_i^0 J_i^1 J_i^0 J_i^1 J_i^0 J_i^1 J_i^0 J_i^1
$\eta(J_i^1) = 2 \cdot T_i$
$w_i = 1$
$if, w_i = 1$
J_i^2
J_i^2 J_i^0 J_i^1 J_i^2 J_i^0 J_i^1 J_i^2 J_i^0 J_i^1
$\eta(J_i^2) = 3 \cdot T_i$
$w_i = 1$

WCRT > D_i and $w_i = 1$

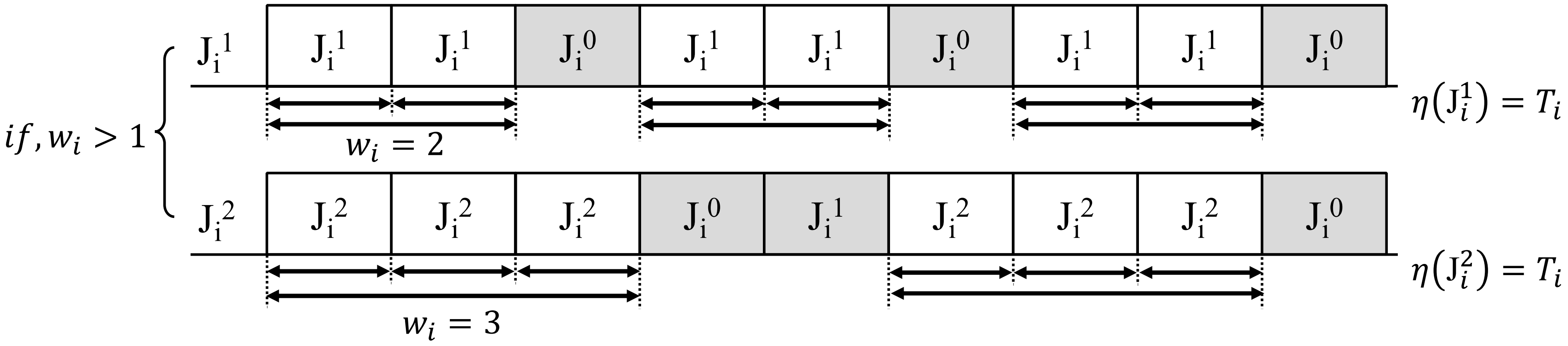
Minimum time interval of a job-class (3/4)

- A job-class whose the WCRT > D_i,

Lemma 5.

The minimum inter-arrival time of J_i^q where $q < K_i - m_i$ and the WCRT of J_i^q is greater than D_i is given by

$$\eta(\mathrm{J}_i^q) = \begin{cases} (q+1) \cdot T_i, if\ w_i = 1 \\ 1 \cdot T_i, if\ w_i > 1 \end{cases}$$



$if, w_i > 1$
$\mathrm{J_i}^1$
$\mathrm{J_i}^1$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$
$w_i = 2$
$\eta(\mathrm{J}_i^1) = T_i$
$\mathrm{J_i}^2$
$\mathrm{J_i}^2$ $\mathrm{J_i}^2$ $\mathrm{J_i}^2$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^2$ $\mathrm{J_i}^2$ $\mathrm{J_i}^2$ $\mathrm{J_i}^0$
$w_i = 3$
$\eta(\mathrm{J}_i^2) = T_i$

WCRT > D_i and $w_i > 1$

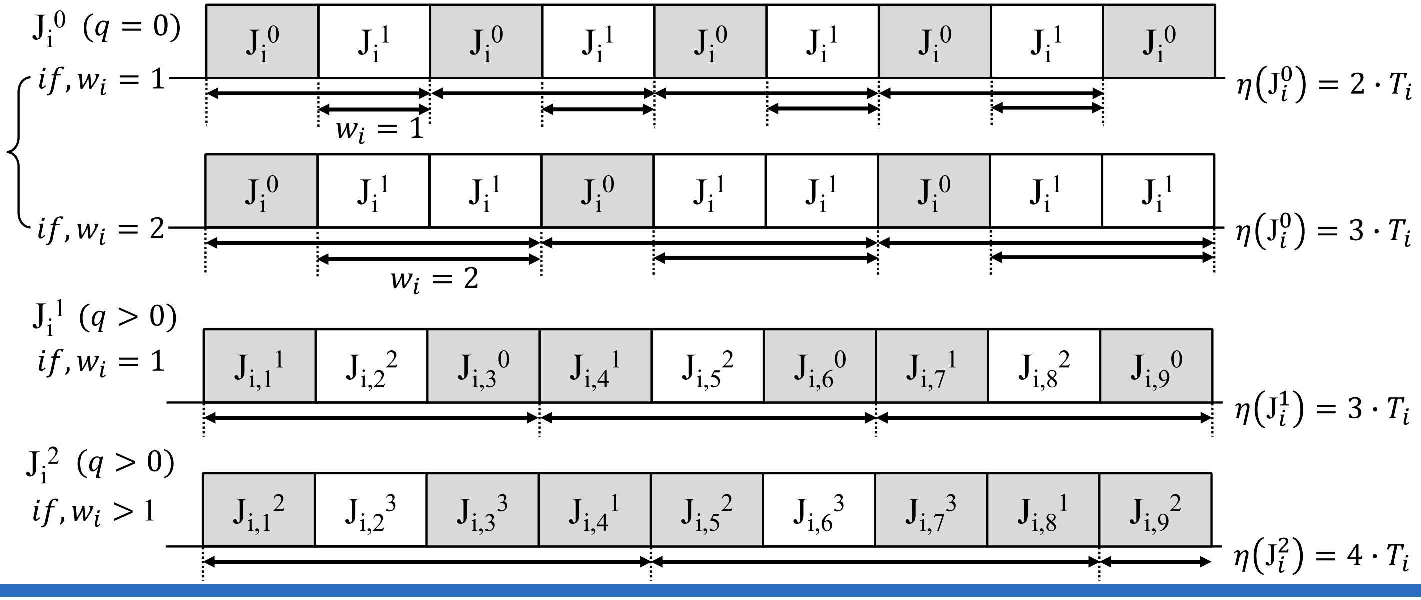
Minimum time interval of a job-class (4/4)

- A job-class whose the WCRT $\leq D_i$,

Lemma 6.

The minimum inter-arrival time of J_i^q where $q < K_i - m_i$ and the WCRT of J_i^q is less than or equal to D_i is given by

$$\eta\left(\mathrm{J}_i^q\right) = \begin{cases}(w_i + 1) \cdot T_i, if\ q = 0 \\ (q + 2) \cdot T_i, if\ q > 0\end{cases}$$



$\mathrm{J_i}^0$ $(q = 0)$
$if, w_i = 1$
$\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$
$w_i = 1$
$\eta(\mathrm{J}_i^0) = 2 \cdot T_i$
$if, w_i = 2$
$\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^1$ $\mathrm{J_i}^0$ $\mathrm{J_i}^1$ $\mathrm{J_i}^1$
$w_i = 2$
$\eta(\mathrm{J}_i^0) = 3 \cdot T_i$
$\mathrm{J_i}^1$ $(q > 0)$
$if, w_i = 1$
$\mathrm{J_{i,1}}^1$ $\mathrm{J_{i,2}}^2$ $\mathrm{J_{i,3}}^0$ $\mathrm{J_{i,4}}^1$ $\mathrm{J_{i,5}}^2$ $\mathrm{J_{i,6}}^0$ $\mathrm{J_{i,7}}^1$ $\mathrm{J_{i,8}}^2$ $\mathrm{J_{i,9}}^0$
$\eta(\mathrm{J}_i^1) = 3 \cdot T_i$
$\mathrm{J_i}^2$ $(q > 0)$
$if, w_i > 1$
$\mathrm{J_{i,1}}^2$ $\mathrm{J_{i,2}}^3$ $\mathrm{J_{i,3}}^3$ $\mathrm{J_{i,4}}^1$ $\mathrm{J_{i,5}}^2$ $\mathrm{J_{i,6}}^3$ $\mathrm{J_{i,7}}^3$ $\mathrm{J_{i,8}}^1$ $\mathrm{J_{i,9}}^2$
$\eta(\mathrm{J}_i^2) = 4 \cdot T_i$

WCRT $\leq D_i$

Interference of job-class-level analysis

- An upper-bound of interference imposed on J_i^q by the higher priority jobs J_k^p of another tasks during arbitrary time t
  - Extension of previous work[†]

$$R_i = C_i + I_i, \qquad I_i = \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Jitter

$$W_i^q(t,\tau_k) = \min\left(\sum_{\forall\, p:\pi_i^q<\pi_k^p} \left\lceil \frac{t+\boxed{\mathcal{J}_k}}{\eta(J_k^p)} \right\rceil \times C_k, \boxed{\left\lceil \frac{t+\boxed{\mathcal{J}_k}}{T_k} \right\rceil \times C_k} \right)$$

Overcome pessimism of job-class-level analysis

Equation 1

✓ $\mathcal{J}_k$ is a jitter of a higher priority job

[†] M. Josephand P. Pandya, “Finding response times in a real-time system,” The Computer Journal, 1986.

Worse-case response time of job-classes

- Worse-case response time of J_i^q is bounded by the recurrence:

$$R_i^{q,n+1} \leftarrow C_i + \sum_{\tau_k \in \Gamma - \tau_i} W_i^q(R_i^{q,n}, \tau_k)$$

Theorem 1.

✓ Γ is the entire taskset

✓ Starts with $R_i^{q,0} = C_i$ and terminates when $R_i^{q,n} + \mathcal{J}_i > D_i$ or $R_i^{q,n+1} = R_i^{q,n}$

Lemma 8.
The job-class-level response time test for weakly-hard tasks given in Theorem 1 is a ***generalization of the task-level iterative response time test for hard real-time tasks****.*

Schedulability check

Theorem 2.

A task is guaranteed to be schedulable if the μ*-patterns at all leaf nodes in its reachability trees satisfy the weakly-hard constraint.*

Complexity of a reachability tree

- Inspecting all possible patterns is an inefficient way ?
  - However, in a reachability tree, the upper-bound on the number of nodes follows the ***Fibonacci sequence*** ($f_{i+2} = f_{i+1} + f_i$)
  - For a task τ_i, the upper-bound of complexity of computing all the reachability trees is represented as

Theorem 4. [†]

$$O_i \leq \ (K_i - m_i + 1) \times \frac{\rho^{\,K_i+1} - (1-\rho)^{K_i+1}}{\sqrt{5}}$$

Where $\rho = \frac{1+\sqrt{5}}{2}$ which is golden ratio and $K_i - m_i + 1$ is the number of job-classes

[†] Verner E. Hoggatt, *Fibonacci and Lucas Numbers*. Boston:Houghton Mifflin Co., 1969

An example of reachability tree

- A job-class J_1^0 of Task 1

J^q, m

: μ−patterns

q : index of a job-class

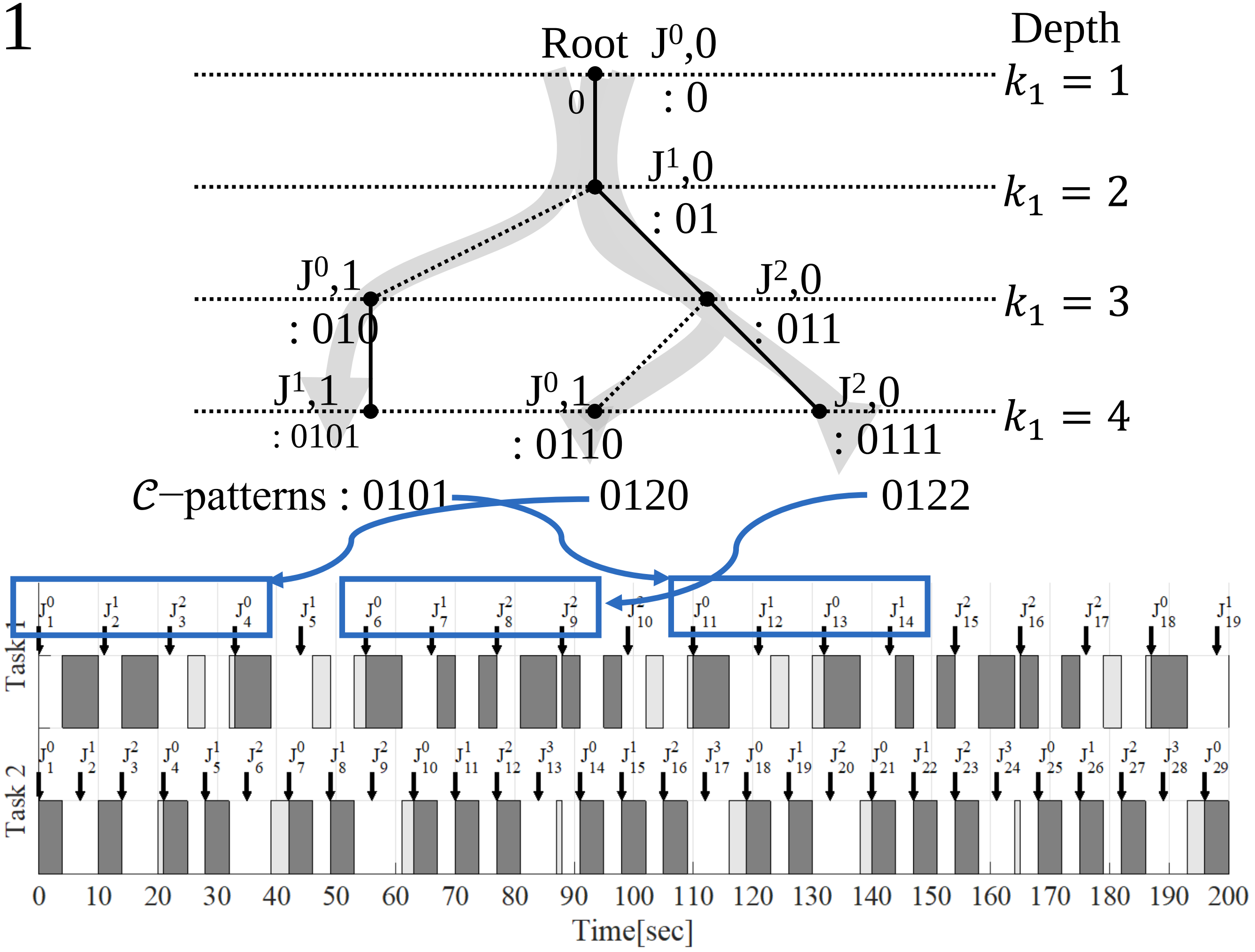
m : number of misses

Miss Meet

0 : miss

1 : meet

$\mathcal{C}$−patterns



Root J^0,0
: 0
0
Depth
$k_1 = 1$
J^1,0
: 01
$k_1 = 2$
J^0,1
: 010
J^2,0
: 011
$k_1 = 3$
J^1,1
: 0101
J^0,1
: 0110
J^2,0
: 0111
$k_1 = 4$
$\mathcal{C}$−patterns : 0101
0120
0122
Task 1
J_1^0 J_2^1 J_3^2 J_4^0 J_5^1 J_6^0 J_7^1 J_8^2 J_9^2 J_{10} J_{11}^0 J_{12}^1 J_{13}^0 J_{14}^1 J_{15}^2 J_{16}^2 J_{17}^2 J_{18}^0 J_{19}^1
Task 2
J_1^0 J_2^1 J_3^2 J_4^0 J_5^1 J_6^2 J_7^0 J_8^1 J_9^2 J_{10}^0 J_{11}^1 J_{12}^2 J_{13}^3 J_{14}^0 J_{15}^1 J_{16}^2 J_{17}^3 J_{18}^0 J_{19}^1 J_{20}^2 J_{21}^0 J_{22}^1 J_{23}^2 J_{24}^3 J_{25}^0 J_{26}^1 J_{27}^2 J_{28}^3 J_{29}^0
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200
Time[sec]