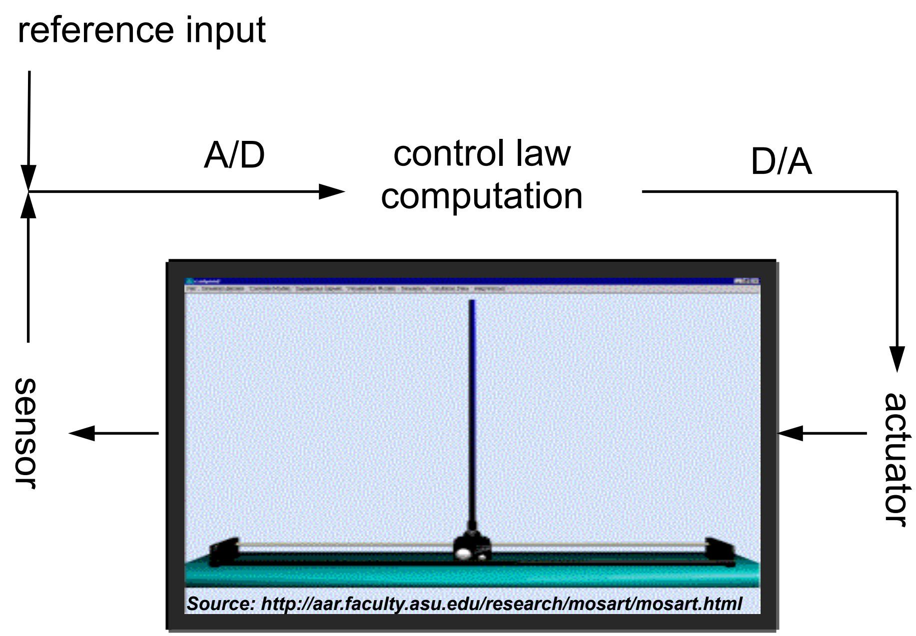
A New Computation Task Model for Cyber-Physical Systems

Kang G. Shin
Real-Time Computing Laboratory
EECS/CSE, University of Michigan@Ann Arbor

This is joint work with Jinkyu Lee

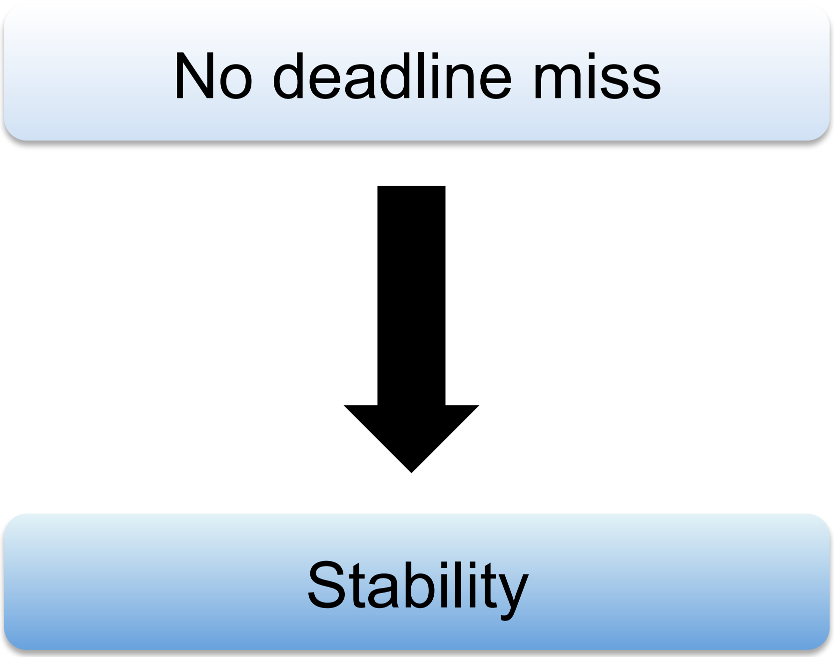
How does *cyber* control *physical*?

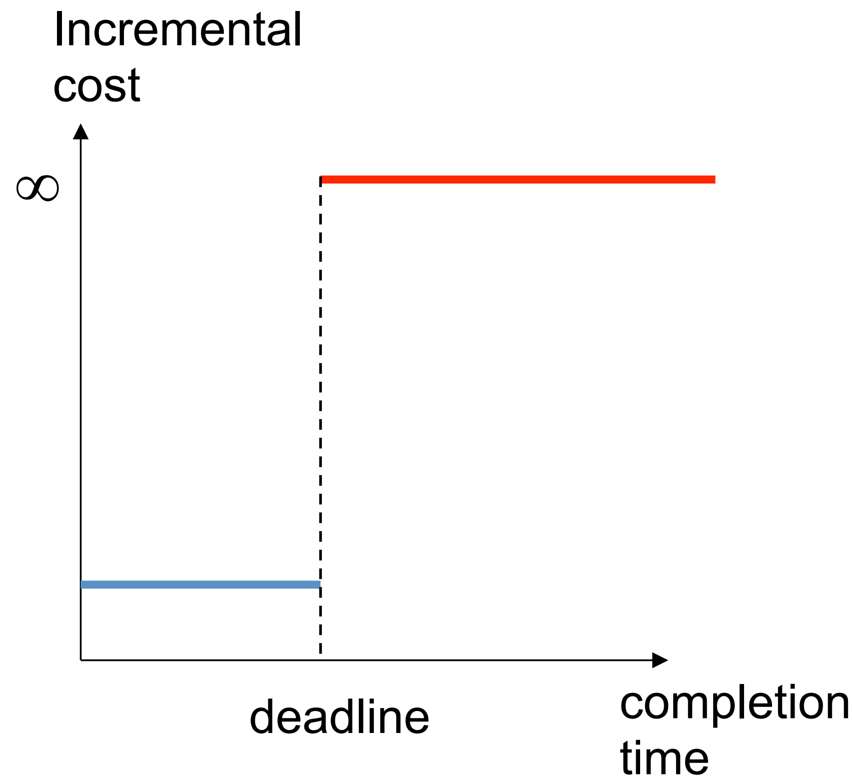
- Feedback control with periodic computation tasks



How does *cyber* control *physical*?

- How to guarantee stability?





How does *cyber* control *physical*?

- Is “no deadline miss” a must?

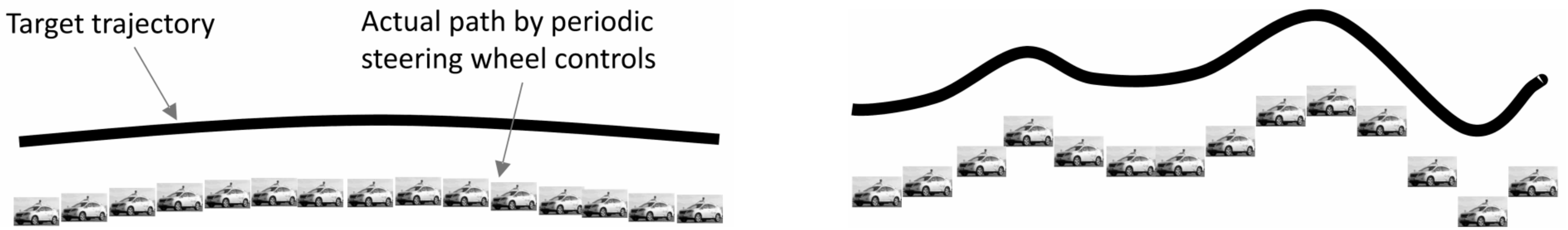
Q1. Is “no deadline miss” always and absolutely required for every task?

Q2. What’s the price we pay to meet the “no deadline miss” requirement?

How does *cyber* control *physical*?

Q1. Is “no deadline miss” always and absolutely required for every task?

No. It depends on tasks and situations.



(a) Even, almost straight road, e.g., highway

(b) Unpaved, winding road, e.g., off-road

How does *cyber* control *physical*?

Q2. What's the price we pay to meet the "no deadline miss" requirement?

Efficiency: we can accommodate more tasks if we relax the requirements.

New CPS task model

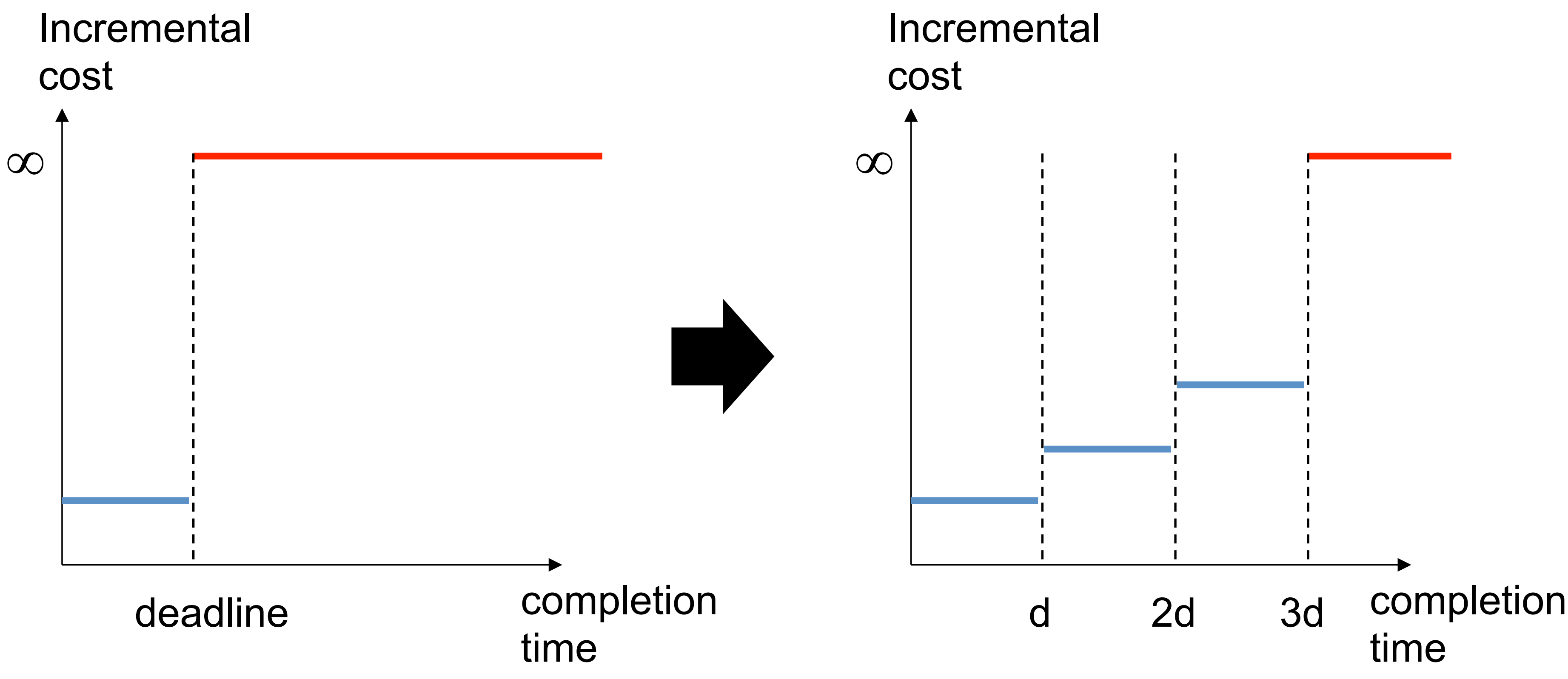
Stability ✚ Efficiency

R1. Capture tolerable job deadline misses without system instability

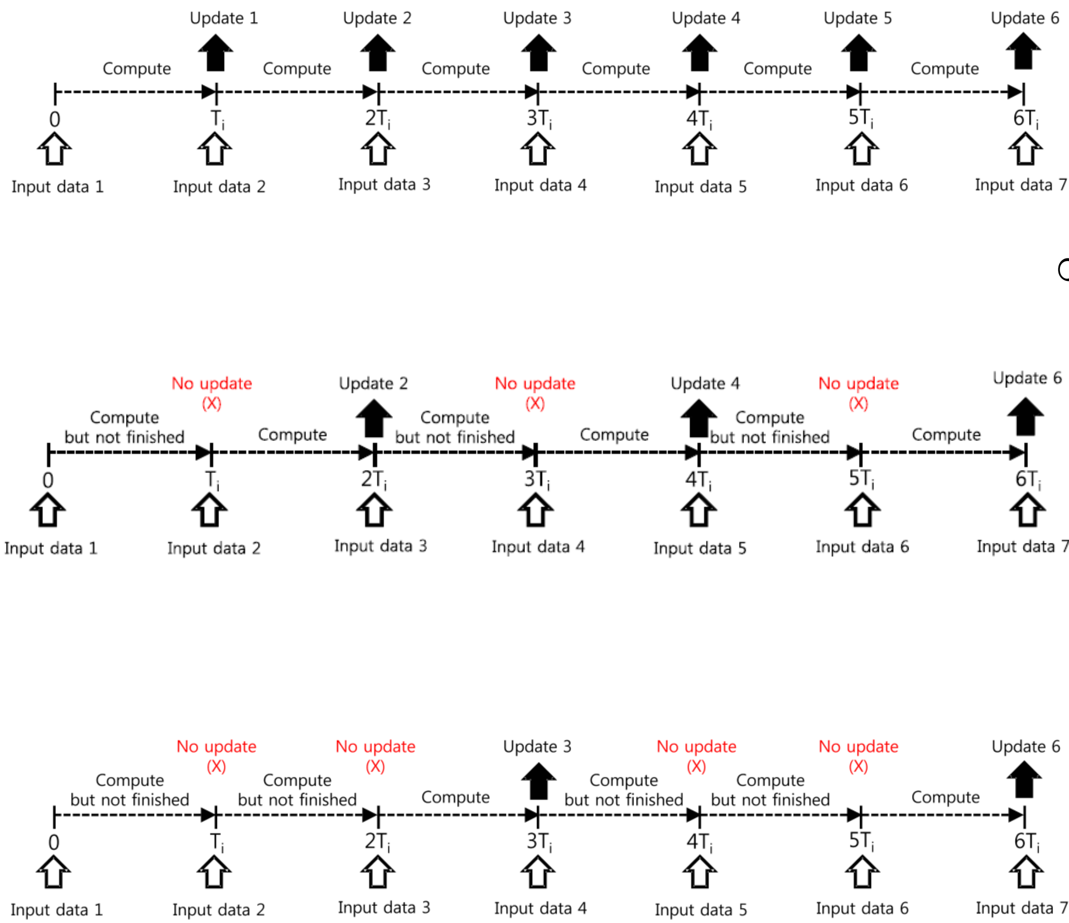
R2. Capture the control cost associated with job deadline misses

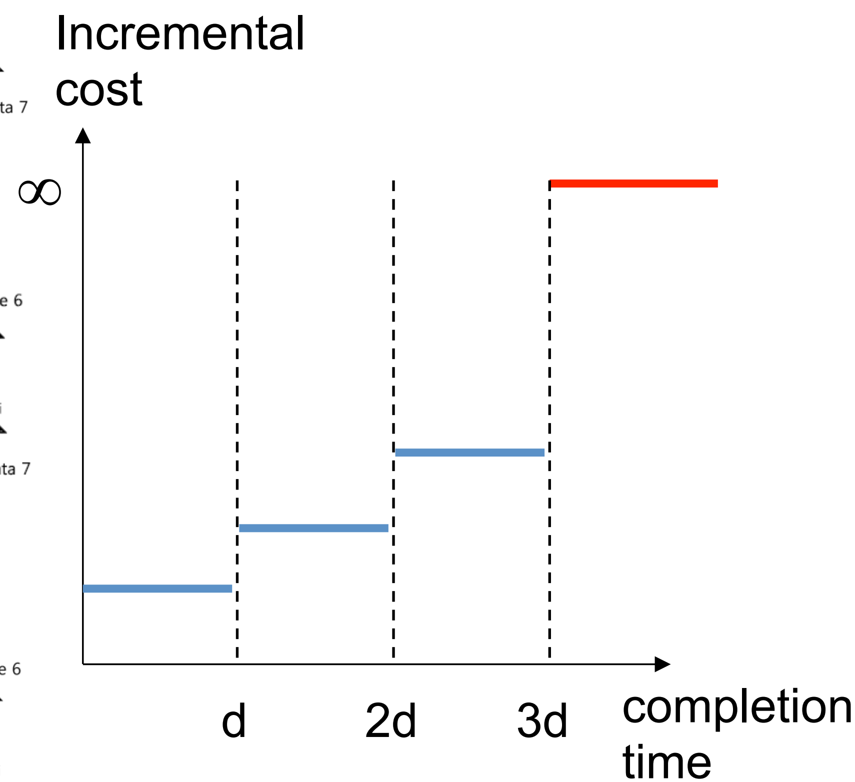
R3. Express a number of job deadline misses with finite states, capturing the coupling between cyber and physical subsystems

New CPS task model



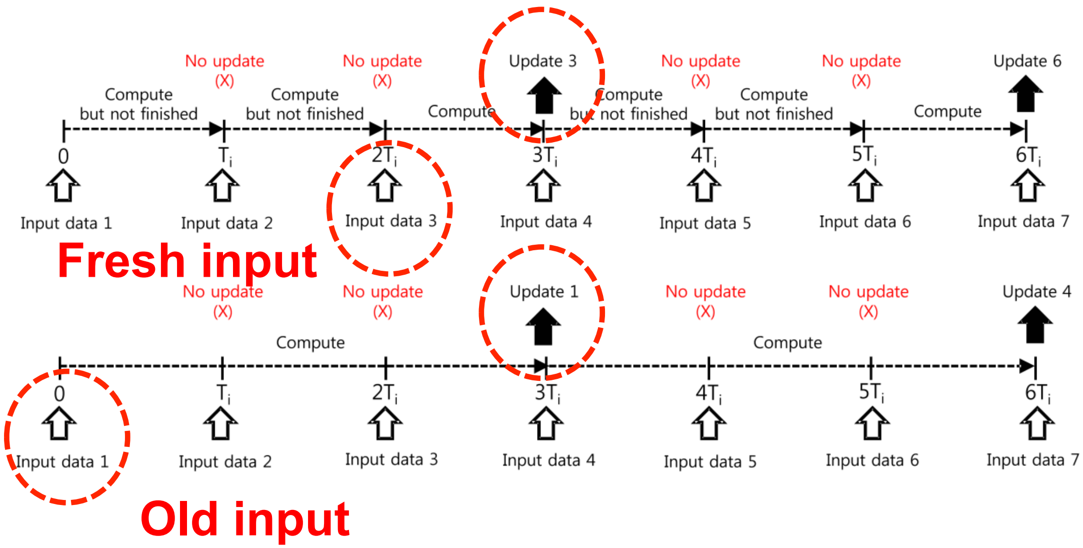
New CPS task model





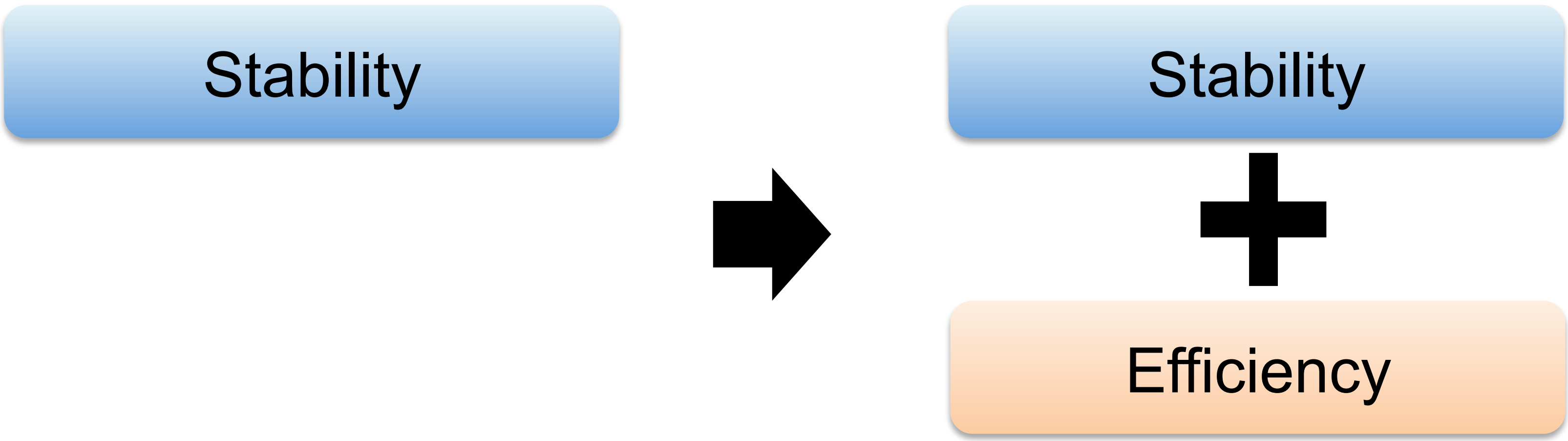
New CPS task model

- What's new here?
  - Existing models cannot capture the control cost associated with job deadline misses
    - Change sampling frequency [9,17,18,26,27]



    - Deadline-miss-tolerance models [11,12,13,14,15,16]
  - Generalization of existing models

Scheduling and analysis



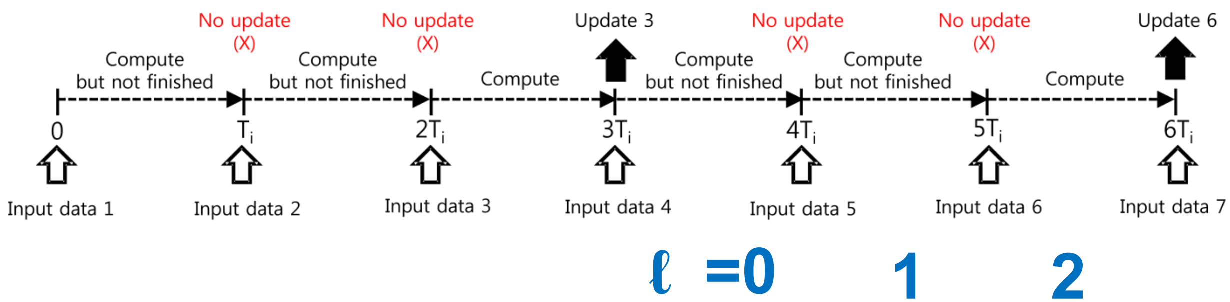
- Scheduling for no deadline miss
- Schedulability analysis

- Scheduling for minimizing incremental cost without any deadline miss
- Schedulability and cost analysis

More complex problem!

Scheduling

- Job state ℓ: the number of consecutive deadline misses



- Job state ℓ is a key parameter that determines both stability and efficiency.
  - ℓ: cyber subsystem state (CSS)

Scheduling

Traditional model → **New model**

Task-level fixed-priority → CSS-level fixed-priority

e.g., Task 1 > Task 2

e.g., Task 1 ($\ell=2$) > Task 2 ($\ell=1$)
> Task 1 ($\ell=1$) > Task 2 ($\ell=0$)
> Task 1 ($\ell=0$)

1. **How to guarantee stability and efficiency with a given priority?** Analysis
2. **How to find the best priority in terms of stability and efficiency?** Priority assignment

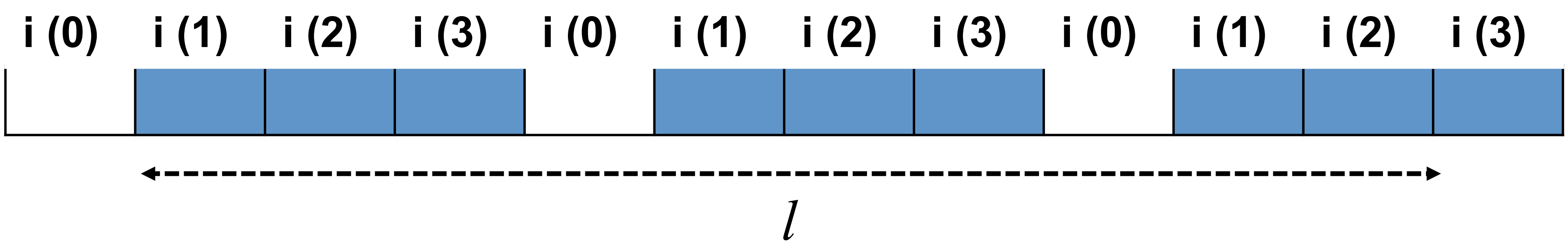
Analysis

Each task	→	Each task, each CSS
Deadline miss or not		Deadline miss or not + Cost upper-bound

- Worst-case release patterns of other tasks
- Worst-case release patterns of other tasks with different CSSes

Analysis

- An upper-bound of the amount of execution of task i's jobs with priority strictly higher than p in an interval of length *l* such that the interval starts at one of the release times of task i's jobs.



$$W_i\,(l, p_i^{\,0}) = 8 * C_i$$

$$W_i(l,p) = \left\lfloor \frac{l}{(m_i+1)\cdot T_i} \right\rfloor \cdot n_i(p) \cdot C_i + \min\left(\left\lceil \frac{l \bmod ((m_i+1)\cdot T_i)}{T_i} \right\rceil, n_i(p) \right) \cdot C_i. \quad (2)$$

Analysis

- Synchronous release is the worst case.
  - Higher-priority execution upper-bounded by $\sum_{\tau_i \in \tau} W_i(l, p)$.
- Response-time analysis

$$R^{x+1} \leftarrow C_k + \sum_{\tau_i \in \tau - \{\tau_k\}} W_i(R^x, p_k^\ell).$$

- Testing order: Task i(0) -> Task i(1) -> Task i(2) …
  - If Task i(x) is schedulable, Tasks i(y>x) are not feasible, and incremental cost is no larger than Task i(x)'s cost.

Improved analysis

- Reduce pessimism by observing that all worst-case situations cannot happen coincidently.
- Details will be available upon request.

Priority assignment

- The number of combinations: n!
  - *n*: the number of all CSSes in a task set
- Addressing time-complexity
  - The lowest -> the highest
  - Observation: Task i(x)'s priority affect Task i(y>x)'s response tir
  - Greedy approach: Try a task w incremental cost difference bet i(x+1).
  - Linear time-complexity

Algorithm 2 Priority assignment for CFP

1: $p_{curr} \leftarrow 1$, i.e., we determine the lowest priority first.
2: $p_i^\ell \leftarrow p_{max}, \forall J_i^\ell$ where $\tau_i \in \tau$ and $1 \leq \ell \leq m_i$.
3: **while** there exists J_i^ℓ such that $p_i^\ell = p_{max}$ **do**
4:   $\mathcal{J} \leftarrow \emptyset$.
5:   **for** $\forall \tau_i \in \tau$ such that $\exists p_i^\ell = p_{max}$ **do**
6:     $\hat{\ell} \leftarrow$ the smallest ℓ such that $p_i^\ell = p_{max}$.
7:     $\mathcal{J} \leftarrow \mathcal{J} \cup \{J_i^{\hat{\ell}}\}$ if $\hat{\ell} < m_i + 1$.
8:     Calculate an upper-bound of the response time of $J_i^{\hat{\ell}}$ in case it has the priority of p_{curr} using Theorem 2.
9:     **if** the upper-bound is smaller than or equal to T_i **then**
10:       $p_i^\ell \leftarrow p_{curr}, \forall \hat{\ell} \leq \ell \leq m_i + 1$.
11:       Exit for-loop and go to Step 19.
12:     **end if**
13:   **end for**
14:   **if** $\mathcal{J} = \emptyset$ **then**
15:     **return** INSTABLE
16:   **else**
17:     Find $J_i^\ell \in \mathcal{J}$ which has the smallest $I_i^{\ell+1} - I_i^\ell$, and then $p_i^\ell \leftarrow p_{curr}$.
18:   **end if**
19:   $p_{curr} \leftarrow p_{curr} + 1$.
20: **end while**
21: **return** STABLE with $\{p_i^\ell\}$.

Evaluation

- Randomly generated 10,000 task sets based on [29]
- Ours(m): allowing at most m consecutive deadline miss, applying CSS-level fixed-priority scheduling with our priority assignment method

Task model	# of task sets proven stable
Classical task model= Ours(0)	1906
Ours(1)	2892
Ours(2)	3201
Ours(3)	3336
Ours(4)	3397

- Compared to the classical task model, our model yields more schedulable task sets.

Evaluation

- Elas(m): disallowing any deadline miss, but period extension by (m+1), applying deadline monotonic scheduling

m	Control cost: Ours(m) / Elas(m)
0	1.0
1	0.63
2	0.59
3	0.49
4	0.46

- Compared to frequency change, our model yields less control costs.

Conclusion

Need of a new CPS task model

Development of the model

Addressing both stability and efficiency

Algorithm, analysis and priority assignment