Failure Tolerance of Multicore Real-Time Systems scheduled by a Pfair Algorithm

Yves MOUAFO

Supervisors

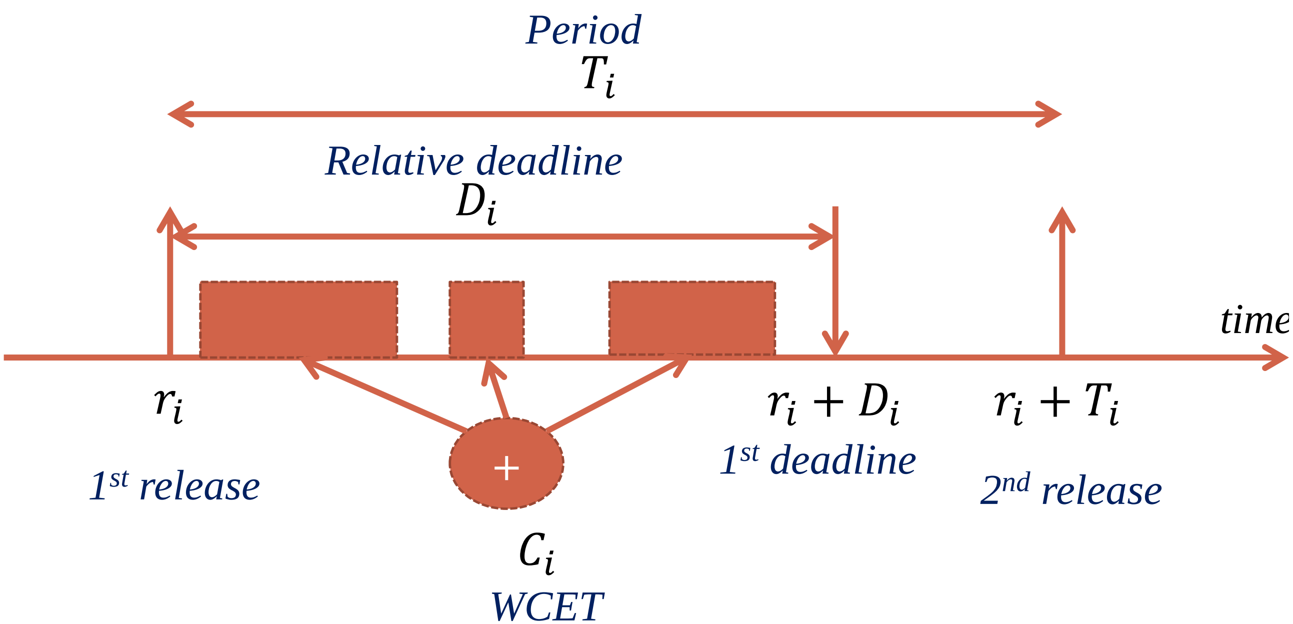
A. CHOQUET-GENIET, G. LARGETEAU-SKAPIN

OUTLINES

1. Context and Problematic
2. State of the art
3. Different scenarios
4. First feasibility result
5. Second feasibility result
6. Future works

The Context

Increased use of multicore platforms in Real-Time System



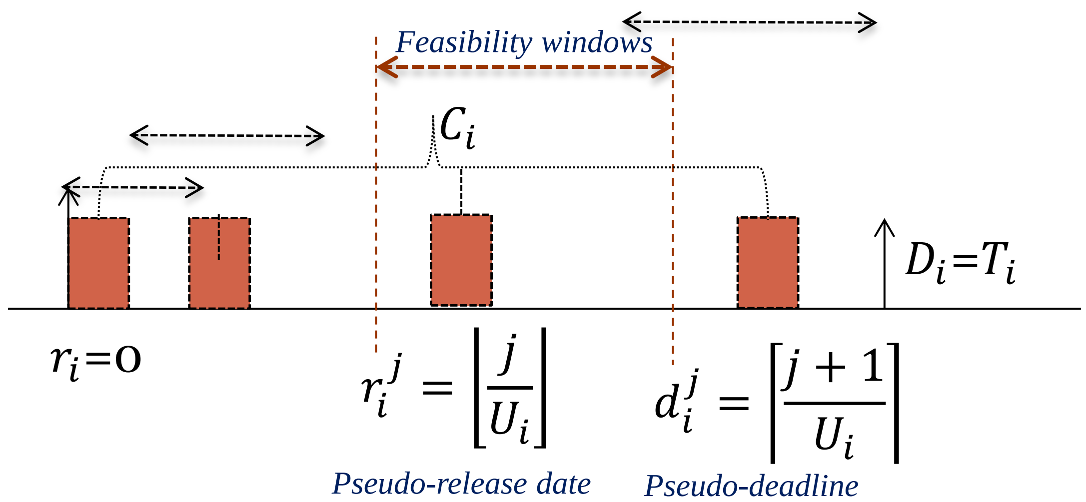
- *Independant*
- *periodic*
- $r_i = 0$
- $D_i = T_i$

<u>Temporal modelling of a periodic task</u>

Context

Scheduling by the Pfair algorithm PD2

- Optimal in our context
- Feasibility condition on m-cores : $U = \sum(U_i = \frac{C_i}{T_i}) \leq m$



$\tau_i < C_i, T_i > \rightarrow C_i$ subtasks $\tau_i^j \left[r_i^j, d_i^j\right)$

- Priority order: increasing pseudo-deadlines + rules for ex-aequo.

Problematic

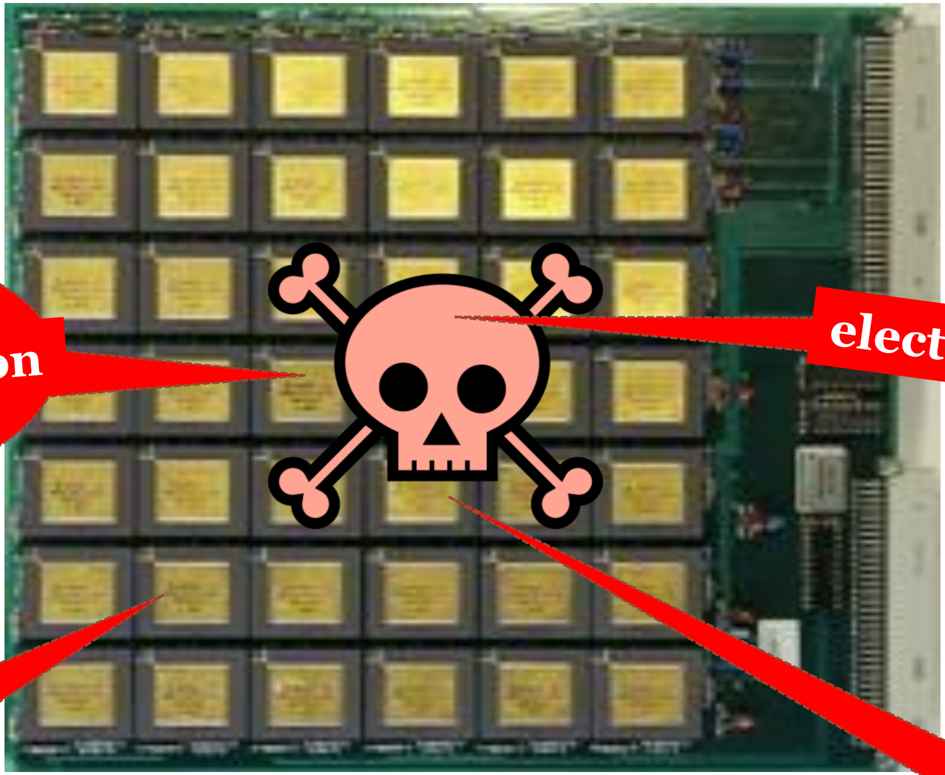
Failure occurrence on any core at any time

Increased processing capacity

Low weight
Small size

Electromigration

electrostatic discharge



Cheap

Low energy consumption

thermal fatigue

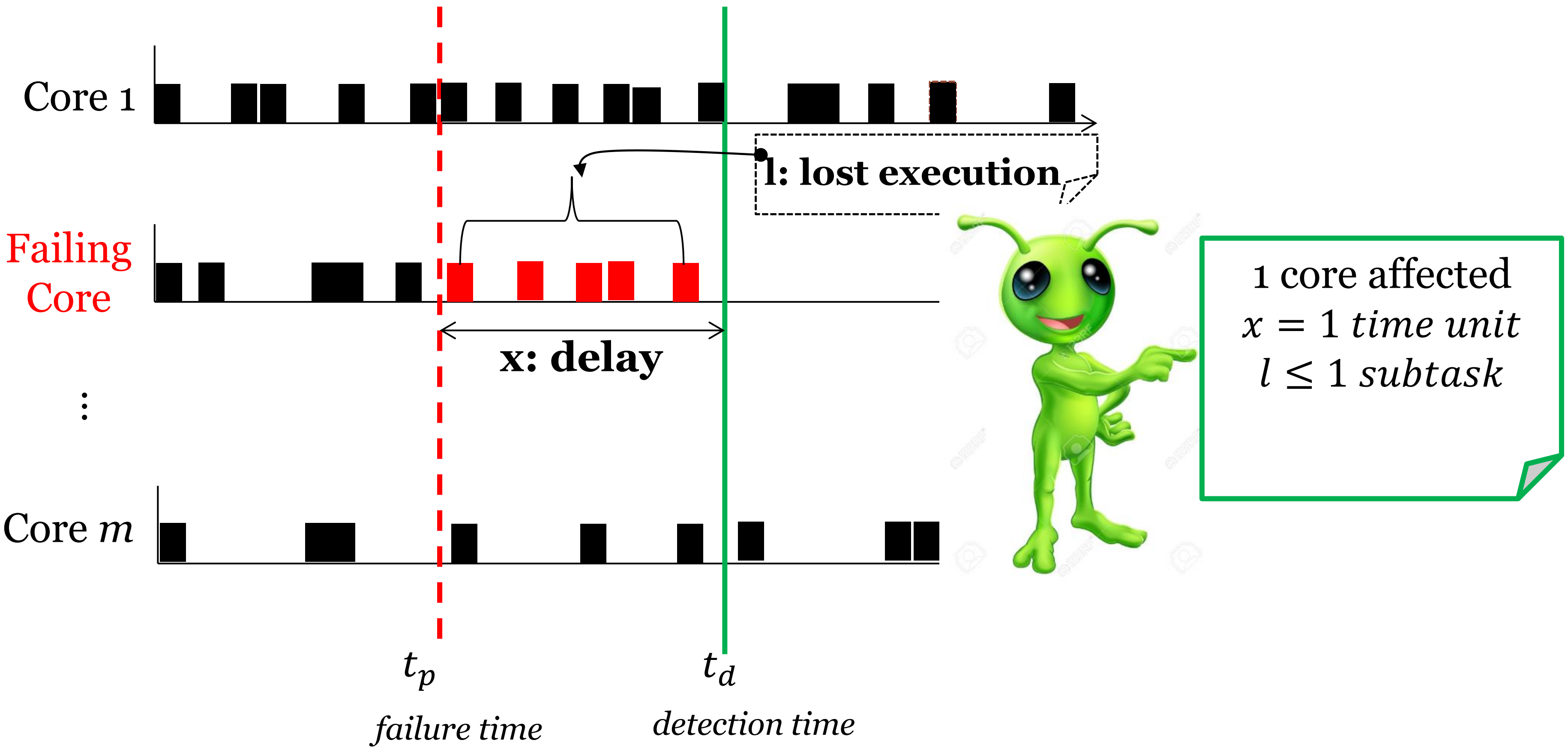
overheat

Several transistors on a single chip

Feasibility condition may not longer be satis

Problematic

Some tasks may be affected by the failure



State of the art
Existing techniques are not suitable

- **Classical approaches**
  - Hardware redundancy *[Pradhan 1996]*
    - Provide each core with a spare or a twin
      - => Over redundant cores as needed
  - Software redundancy *[Koren et al, 2007]*
    - Provide to each task 2 copies: a primary and a backup
      - => Increase of system load
  - Time redundancy *[Kopetz et al. 2003]*
    - Exploit the slack between task completion and deadline
      - => similar to our approach
      - => useful only for transient and intermittant failures
- **Most used in partitioned scheduling**

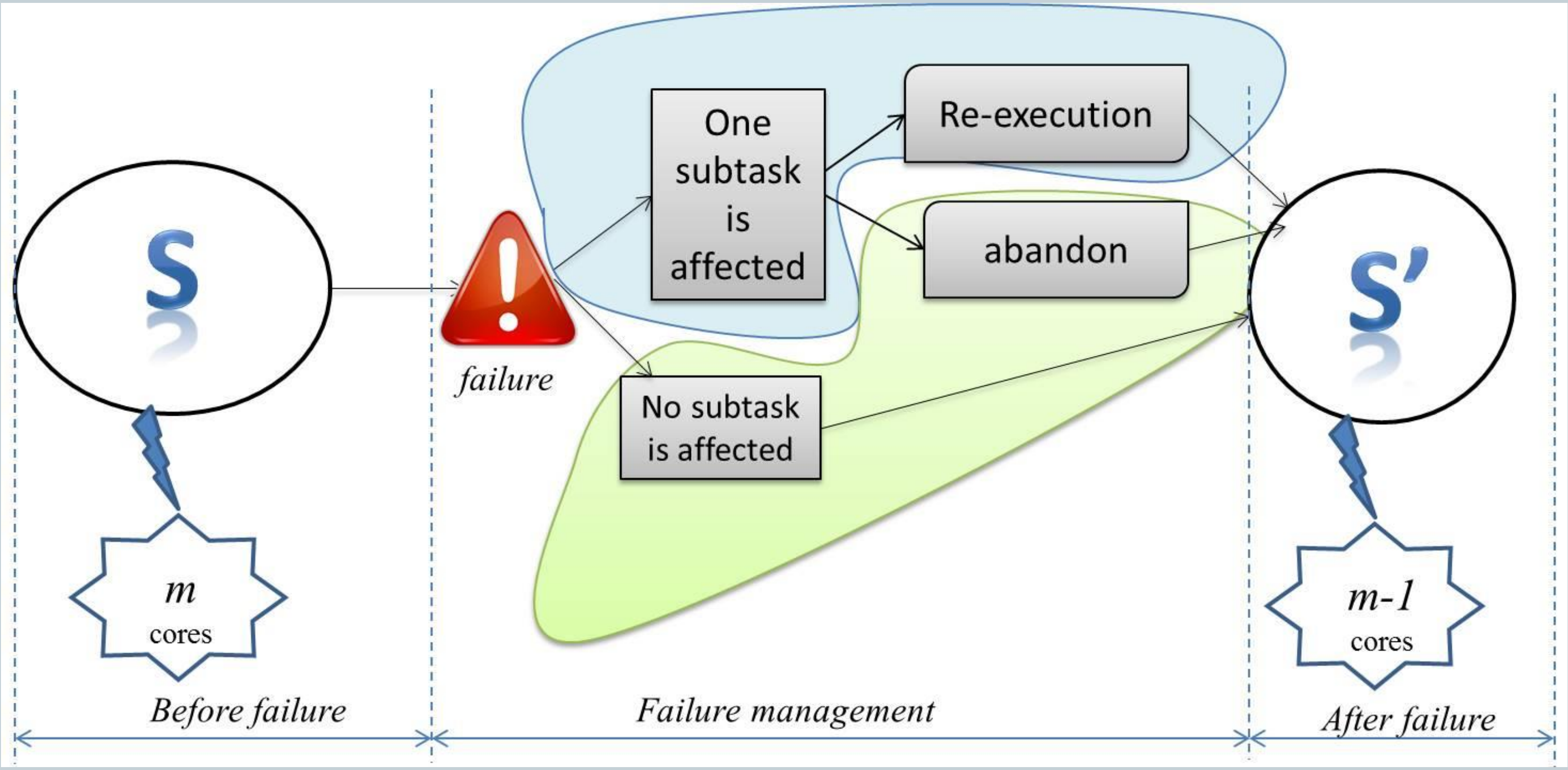
Our goals

Avoid the limitations of the classical techniques

- Provide strictly the number of cores needed
  - $m = \lfloor U \rfloor + 1$
- Limit the hardware redundancy to one core
- Avoid the use of backup copies
- Resume only the lost execution

Different scenarios

- **At any time, a failure may occur on any of the cores**

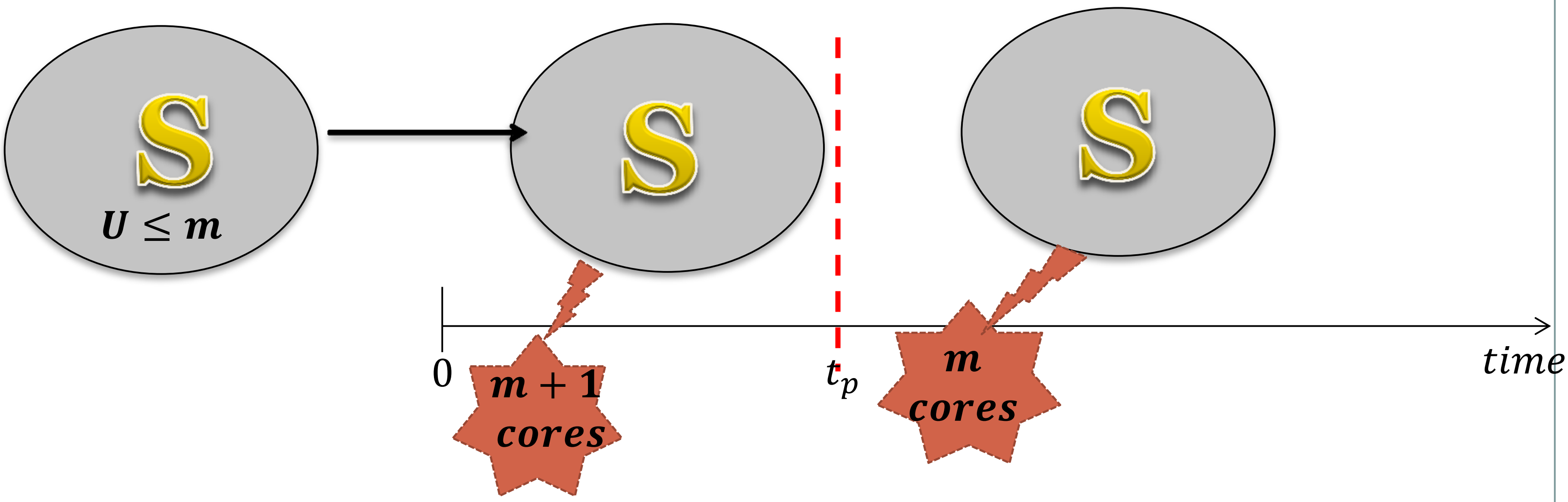


Two Possible Scenarios

Allocate or not additional time to affected tasks

- **No task is affected**
  - Continue the execution
- **One affected task**
  - Partial completion is acceptable
    - **No further time allocation**
    - eg. iterative tasks
  - Full completion is needed
    - **Additional time allocation**

No Further Time Allocation
Limited Hardware Redundancy



- **Provide an additional core**

First Feasibility Result

Limited Hardware Redundancy provide a valid schedule

- **Notations**

- $Sched_m^S$: PD2 schedule of S on a m-core processor
- $Sched_{(m+1)\rightarrow m}^S$: PD2 schedule of S with limited hardware redundancy
- $Pending(Sched, t)$: List of pending subtasks in schedule $Sched$ at time t
- $Exec(\tau_i^j, Sched)$: Execution time of subtask τ_i^j in schedule $Sched$

- **Assumption**

- $U \leq m$ => $Sched_m^S$ and $Sched_{m+1}^S$ are valid and fair

- **Theorem**

The resulting schedule $Sched_{(m+1)\rightarrow m}^S$ is valid and fair

$$\forall \tau_i^j, \qquad r_i^j \leq Exec\left(\tau_i^j, Sched_{(m+1)\rightarrow m}^S\right) < d_i^j$$

Proof

Based on two lemmas

- **Lemma 1**

At any time, subtasks pending in are $Sched^S_{(m+1)\to m}$ also pending in $Sched^S_m$

$$Pending(\tau_i^j, Sched^S_{(m+1)\to m}) \subseteq Pending(\tau_i^j, Sched^S_m)$$

- **Lemma 2**

Any subtask is scheduled earlier in $Sched^S_{(m+1)\to m}$ than in $Sched^S_m$

$$\forall \tau_i^j, Exec(\tau_i^j, Sched^S_{(m+1)\to m}) \leq Exec(\tau_i^j, Sched^S_m)$$

- **Proof of the theorem**

- At $t \leq t_p$ $Sched^S_{(m+1)\to m} = Sched^S_{(m+1)}$ valid and fair

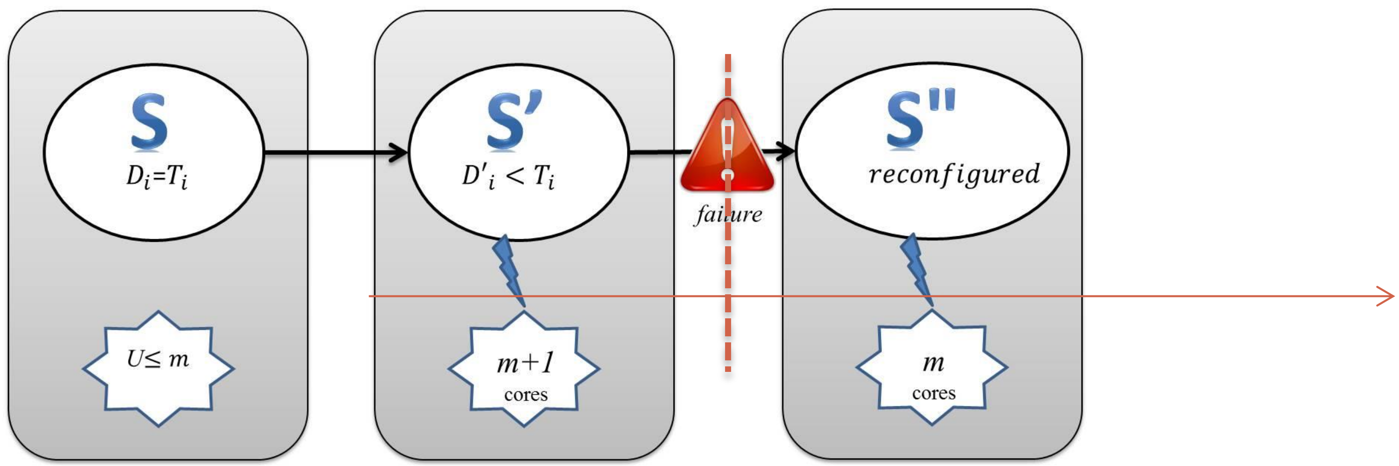
- At $t_p \leq t \leq H, \forall \tau_i^j, r_i^j \leq Exec\left(\tau_i^j, Sched^S_{(m+1)\to m}\right) \underset{(lemma2)}{\leq} Exec(\tau_i^j, Sched^S_m) < d_i^j$

- At $t > H$ $Sched^S_{(m+1)\to m} = Sched^S_{(m)}$ valid and fair

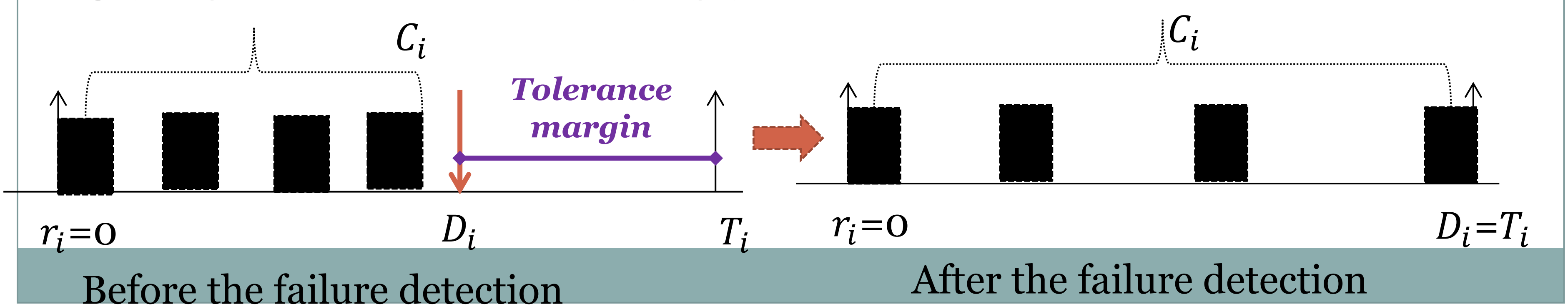
Additional Time Allocated

Two combined techniques

- **Limited hardware redundancy**

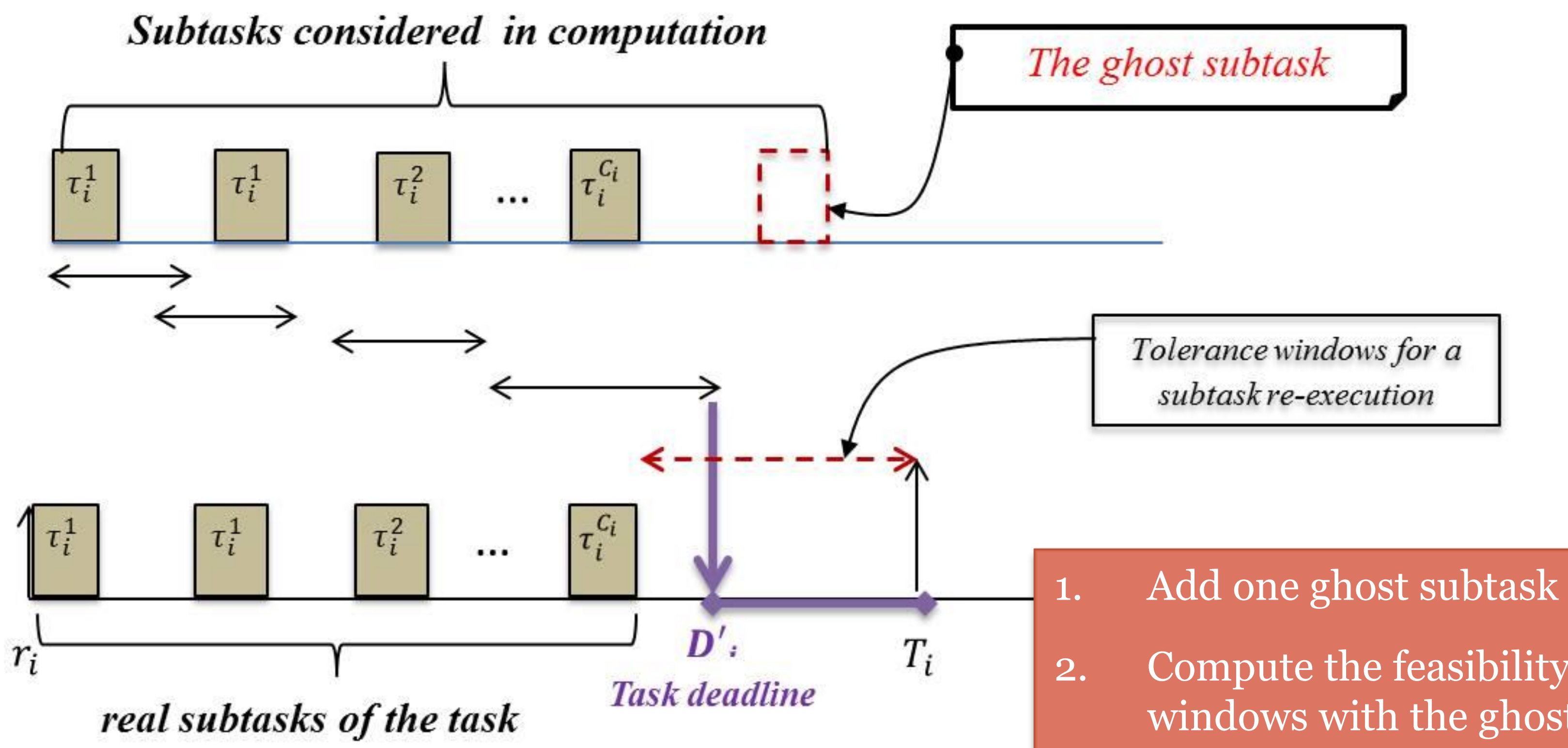


- **Constrain and release**



Deadlines computation

Ghost subtasks method



1. Add one ghost subtask
2. Compute the feasibility windows with the ghost subtasks
3. Task deadline = pseudo-deadline of the subtask before the last

Dynamic Reconfiguration

Subtasks switch from one system parameters to another

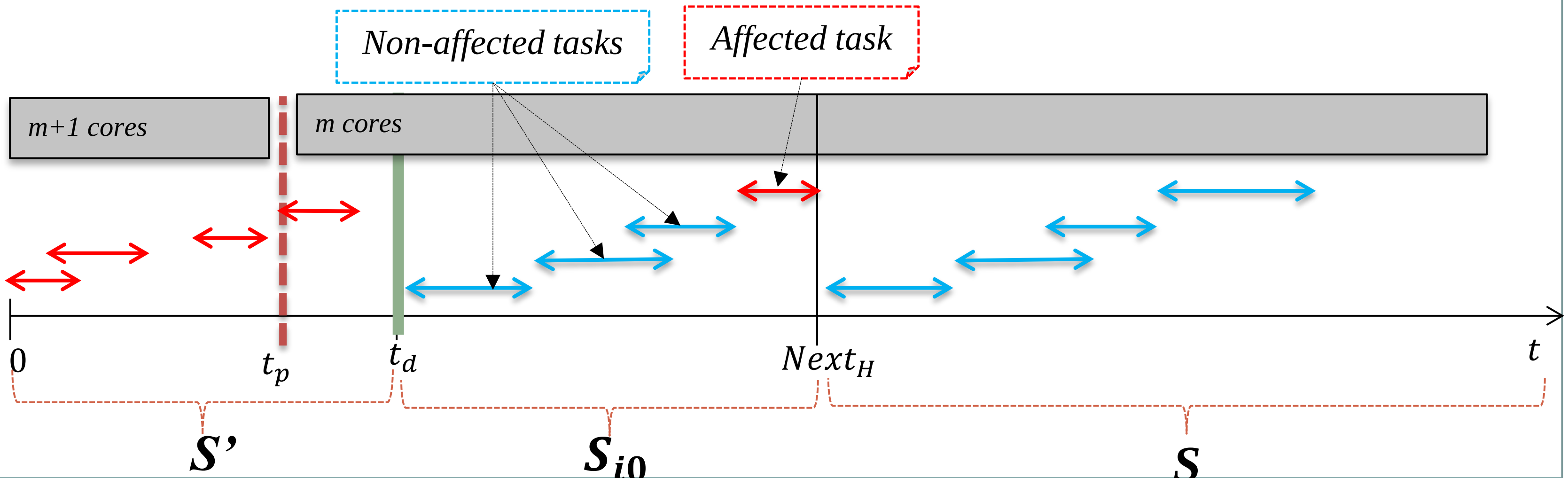
- **Involved systems**
  - *Initial* System S: $\boldsymbol{\tau_i < C_i, D_i = T_i >}$
  - *Constrained* System S': $\boldsymbol{\tau'_i < C_i, D'_i < T_i >}$
  - *Intermediate* System S_{i_0}: $\boldsymbol{\tau_{i \neq i_0} < C_i, D_i = T_i >,}$ $\tau_{i_0} < C_{i_0} + 1, D_{i_0} = T_i$

$$- U_S = \sum \frac{C_i}{T_i} \leq m$$

$$- CH_{S'} = \sum \frac{C_i}{D'_i} \leq m + 1$$

$$- U_{S_{i0}} = \sum_{i \neq i0} \frac{C_i}{T_i} + \frac{C_{i0} + 1}{T_{i0}} \leq m$$

Resulting system notation: $\boldsymbol{S' \rightarrow_{t_p} S_{i_0}}$

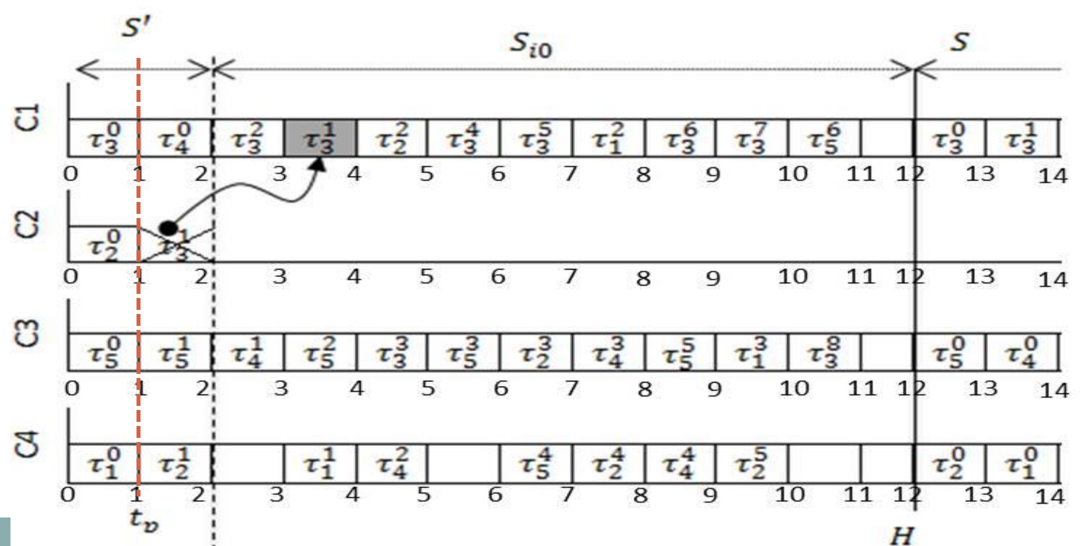


Example

$$t_p = 1,\ x=1, \tau_{i0}^{j0} = \tau_3^1$$

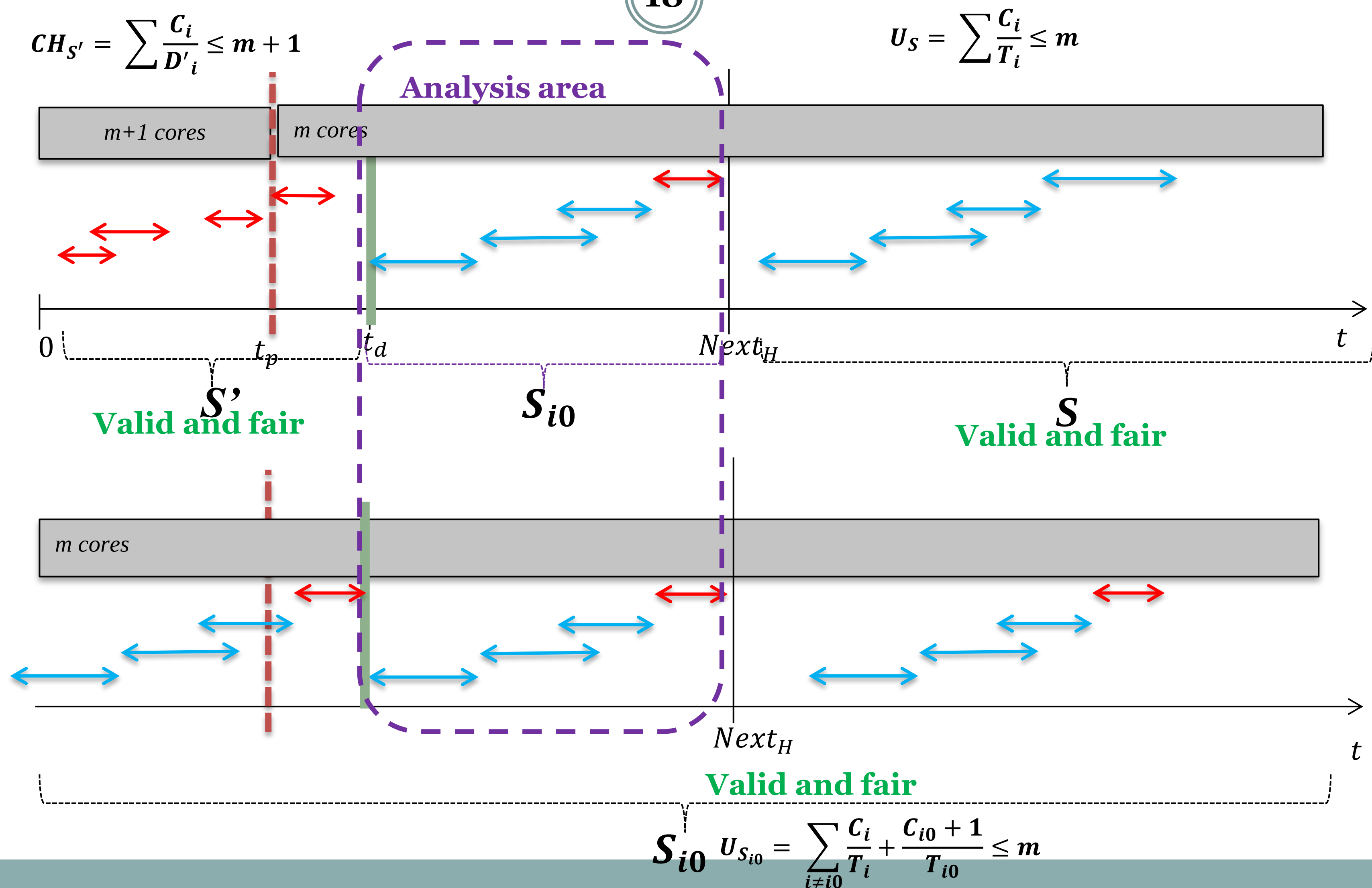
- S: $\tau_1 < 1, 3 >, \tau_2 < 3, 6 >, \tau_3 < 3, 4 >, \tau_4 < 5, 12 >, \tau_5 < 7, 12 >$

		S		S'		S_3		R	
	τ_3^0	0	2	0	1	0	1	0	1
	τ_3^1	1	3	1	2	1	2	1	2
	τ_3^2	2	4	2	3	2	3	2	3
	$\tau_3^{t_1}$	-	-	-	-	3	4	3	4
	τ_3^3	4	6	4	5	4	5	4	5
	τ_3^4	5	7	5	6	5	6	5	6
τ_3	τ_3^5	6	8	6	7	6	7	6	7
	$\tau_3^{t_2}$	-	-	-	-	7	8	-	-
	τ_3^6	8	10	8	9	8	9	8	9



Scheduling analysis

Comparison between $S' \rightarrow_{t_p} S_{i_0}$ and S_{i_0} schedules



An issue

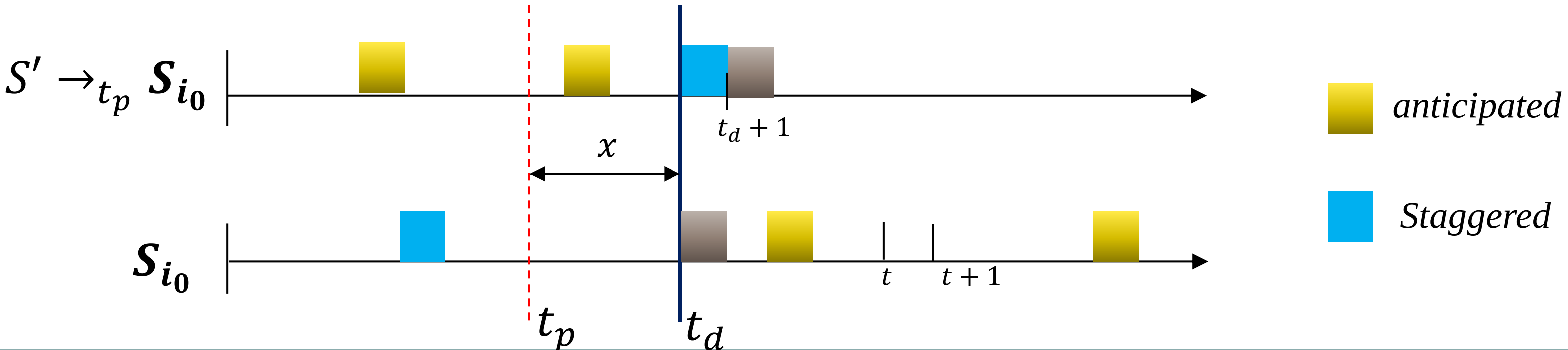
Subtasks priority inversion

- *Pending subtasks at t=0*
  - $S': \tau_3^0\ \tau_2^0\ \tau_5^0\ \tau_1^0\ \tau_4^0$
  - $S_{i_0}: \tau_3^0\ \tau_5^0\ \tau_2^0\ \tau_4^0\ \tau_1^0$

In S'	In $\boldsymbol{S_{i_0}}$
$\tau_2^0 >_{PD2} \tau_5^0$	$\tau_5^0 >_{PD2} \tau_2^0$
$\tau_1^0 >_{PD2} \tau_4^0$	$\tau_4^0 >_{PD2} \tau_1^0$

$>_{\boldsymbol{PD2}}$ i.e *has higher priority over*

- **2 kinds of subtasks at $\boldsymbol{t_d}$**



Example 1

Without staggered subtasks

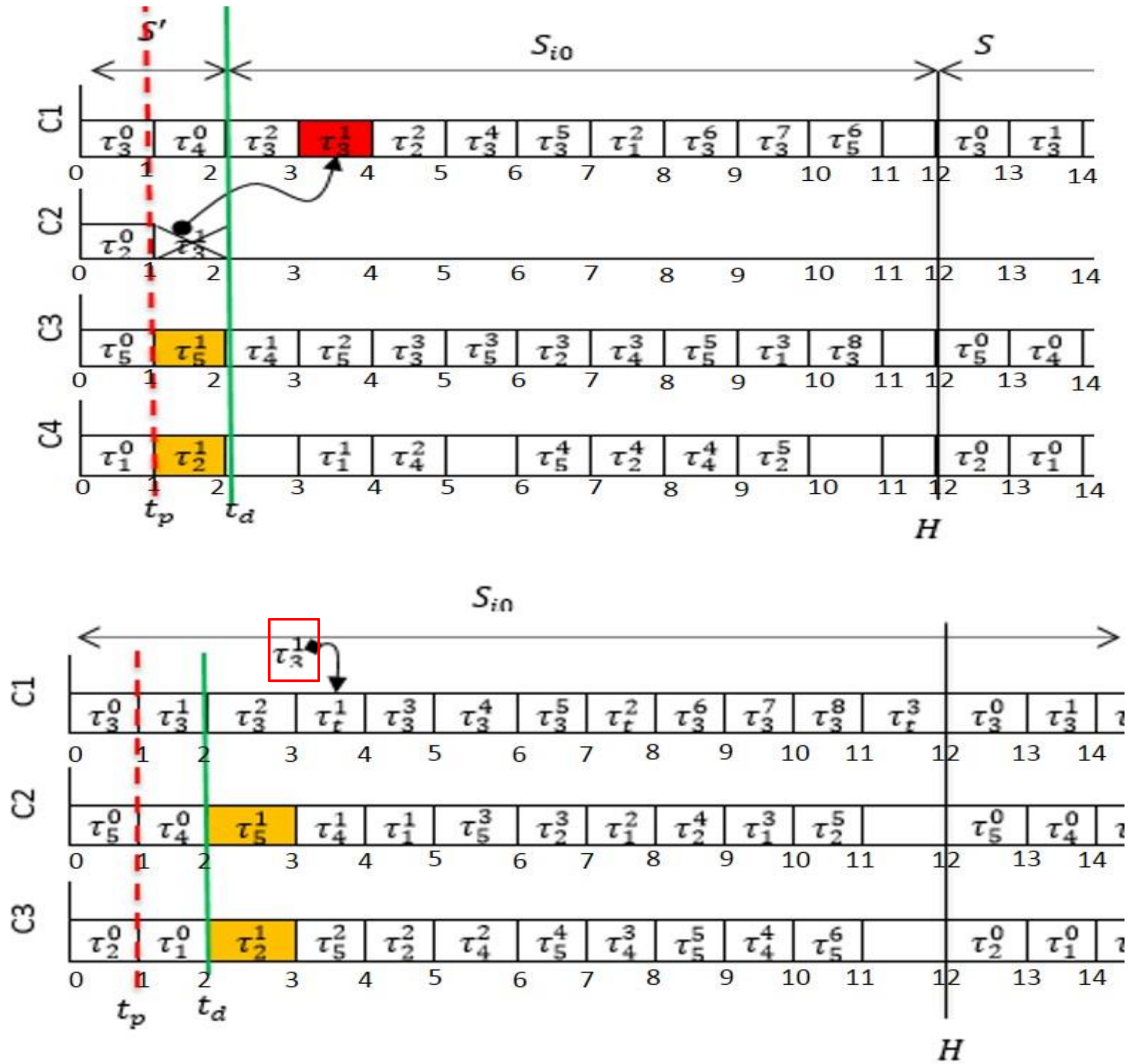
S: $\tau_1 < 1, 3 >$,

$\tau_2 < 3, 6 >$,

$\tau_3 < 3, 4 >$,

$\tau_4 < 5, 12 >$,

$\tau_5 < 7, 12 >$



affected

anticipated

Staggered

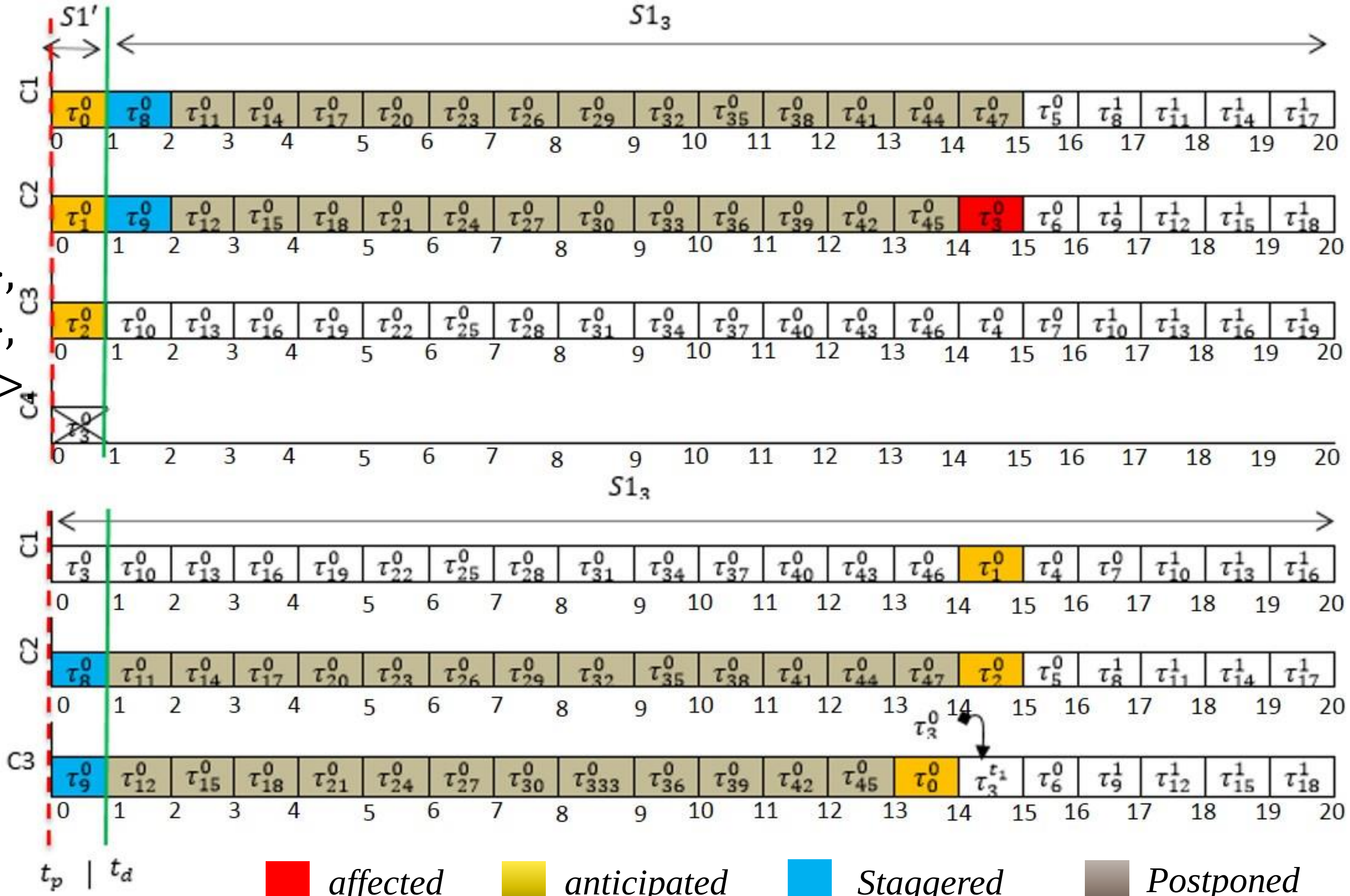
Example 2

With staggered subtasks

$\boldsymbol{S1}$:
$\tau_{0-3} < 1, 20 >$,
$\tau_{4-7} < 1, 36 >$,
$\tau_{8-47} < 2, 38 >$.

$\boldsymbol{S1'}$:
$\tau_{0-3} < 1, 10{,}20 >$,
$\tau_{4-7} < 1, 18{,}36 >$,
$\tau_{8-47} < 2, 26{,}38 >$.

$\boldsymbol{S1_{i_3}}$:
$\tau_{0-2} < 1, 20 >$,
$\tau_3 < 2, 20 >$,
$\tau_{4-7} < 1, 36 >$,
$\tau_{8-47} < 2, 38 >$.



Our Result

The resulting scheduling is valid and fair

- **Assumptions**

- S is feasible on m cores
- S' is feasible on m+1 cores
- S_{i_0} is feasible on m cores
- There is no staggered subtask and t_p is arbitrary
- Or there are some staggered subtasks and $t_p = 0[H]$

- **Theorem**

The resulting scheduling of $S' \rightarrow_{t_p} S_{i_0}$ on $(m+1) \rightarrow m$ cores is valid and fair

$$\forall \tau_i^j, \qquad r_i^j \leq Exec\left(\tau_i^j, S' \rightarrow_{t_p} S_{i_0}\right) < d_i^j$$

Proof

For any t_p with no staggered subtasks

- **Proposition 1** *(Remark 1)*

$R(t)$: *a subtask is not scheduled later in* $S' \rightarrow_{t_p} S_{i_0}$ *than in* S_{i_0}

Proof

At any time $t \geq t_p$:

- **Prop1(t)**: $Pending(\tau_i^j, S' \rightarrow_{t_p} S_{i_0}) \Rightarrow Pending(\tau_i^j, S_{i_0})$
- **Prop2(t)**:
$$\{\exists\ k \text{ subtasks with higher priority than } \tau_i^j \text{ in } S' \rightarrow_{t_p} S_{i_0}\}$$
$$\Rightarrow \{\exists \geq k \text{ subtasks with higher priority than } \tau_i^j \text{ in } S_{i_0}\}$$

- **Conclusion**

$$r_i^j(S_{i_0}) \leq Exec\left(\tau_i^j, S' \rightarrow_{t_p} S_{i_0}\right) \leq Exec(\tau_i^j, S_{i_0}) < d_i^j(S_{i_0})$$

Proof

For $t_p = 0[H]$ with some staggered subtasks

- **Notations**

  τ_s^g: staggered subtask τ_u^p: postponed subtask τ_i^j: any subtask

- **Proposition 2** *(Remark 2)*
  - x staggered subtasks => $x + 1$ anticipated subtasks
  - The staggered subtasks meet their pseudo-deadlines
    $Exec\left(\tau_s^g, S' \rightarrow_{t_p} S_{i_0}\right) = t_d < d_s^g(S_{i_0})$
  - The postponed subtasks meet their pseudo-deadlines
    $\left\{Exec\left(\tau_u^p, S_{i_0}\right) = t\right\} \Rightarrow \left\{Exec\left(\tau_u^p, S' \rightarrow_{t_p} S_{i_0}\right) = t + 1 < d_u^p(S_{i_0})\right\}$
  - When the postponement ends subtasks are scheduled earlier
    <u>If</u> $\left\{Exec\left(\tau_i^j, S_{i_0}\right) \leq t\right\} \Rightarrow \left\{Exec\left(\tau_i^j, S' \rightarrow_{t_p} S_{i_0}\right) \leq t\right\}$
    <u>Then</u> $R(t)$ of Proposition 1 is true.

- **Conclusion**

  $r_i^j(S_{i_0}) \leq Exec\left(\tau_i^j, S' \rightarrow_{t_p} S_{i_0}\right) < d_i^j(S_{i_0})$

Future works

Complete the proof and explore other situations

- Proof: $t_p \neq H$ and there are staggered subtasks
- The failure detection delay x is larger
  - ✓ Use an aperiodic flow
- Several cores are affected
  - ✓ Reduce the system load (delete tasks or subtasks)