Laboratoire d'Informatique Scientifique et Industrielle
École Nationale Supérieure de Mécanique et d'Aérotechnique
1, avenue Clément Ader - BP 40109 - 86961 Futuroscope cedex - France

Uniprocessor Schedulability and Sensitivity Analysis of Multiple Criticality Tasks with Fixed-Priorities

LISI / ENSMA	Université Libre de Bruxelles
François DORIN Pascal RICHARD Michaël RICHARD	Joël GOOSSENS

François DORIN
francois.dorin@lisi.ensma.fr
http://www.lisi.ensma.fr

Summary

- Task models
  - Basic model
  - Multiple criticality model
- State of the Art
- Analysis of Vestal's algorithm
- Sensitivity analysis
- Conclusion

Liu & Layland's model

- Classical characteristics
  - Period: Pi
  - Deadline: Di
  - Execution Time: Ci

- Ci is overestimated for critical tasks
  - ➔ interference of critical tasks on non critical tasks are highly overestimated
  - ➔ oversizing of the system

Aeronautic standards: DO 178B

Level	Criticity	Failure condition	Description
A	5	Catastrophic	Failure may cause a crash
B	4	Hazardous	Failure has a large negative impact on safety of performance, or reduces the ability of the crew to operate the plane due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.
C	3	Major	Failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries)
D	2	Minor	Failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change)
E	1	No effect	Failure has no impact on safety, aircraft operation, or crew workload

Vestal's Model

- New model
  - Vestal: "*Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance*"
  - New notion: criticality level Li
  - One execution time per criticality level
    - Constraint: Ci(l-1) <= Ci(l) for all l
- Example
  - 2 criticality levels : {1, 2}

Task	Di = Ti	Li	Ci(1)	Ci(2)
1	4	2	2	2
2	7	1	2	5

In practice

Task	Di = Ti	Priority	Li	Ci(1)	Ci(2)
1	4	High	2	2	2
2	7	Low	1	2	5

For task 2

Task	Di = Ti	Priority	Ci
1	4	High	2
2	7	Low	2

For task 1

Task	Di = Ti	Priority	Ci
1	4	High	2
2	7	Low	5

Why not just consider Ci(2) for all tasks?

Task	Di = Ti	Priority	Ci
1	4	High	2
2	7	Low	5

→ **Not schedulable**

Constraint: Ci(l-1) <= Ci(l) for all l

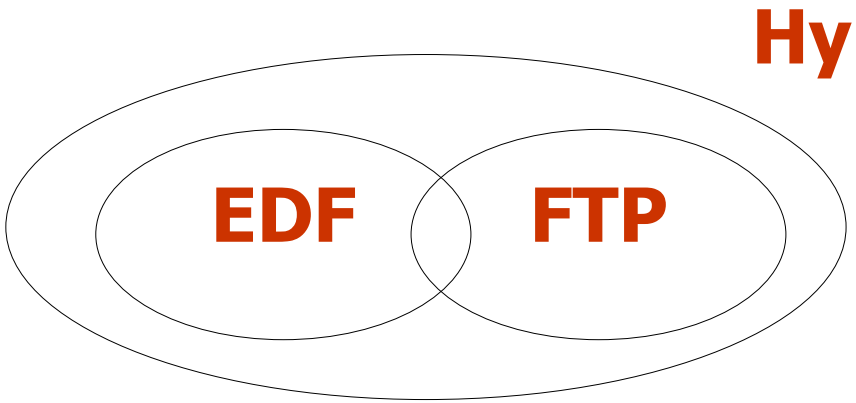
Task	Di = Ti	Priority	Li	Ci(1)	Ci(2)
1	4	High	2	2	2
2	7	Low	1	5	2

- Not schedulable if $L_2=1$
- Schedulable if $L_2=2$

State of the Art

- Vestal [2007]: *Preemptive scheduling of multi-criticality systems with varying degrees of execution times assurance*
  - Vestal's algorithm

- Vestal, Baruah [2008]: S*chedulability analysis of sporadic tasks with criticality specifications*
  - EDF and FTP are not comparable
  - Hybrid algorithm



Vestal's Algorithm

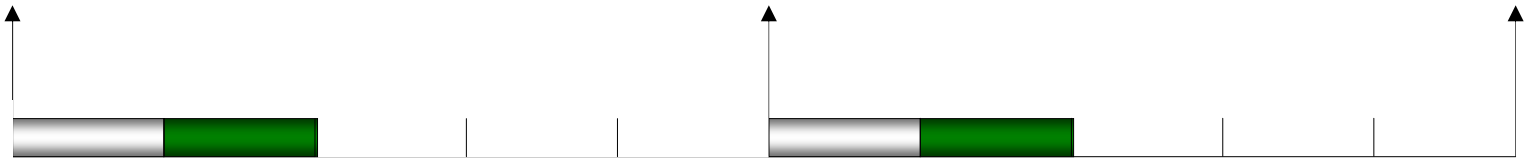
Principles

1) Using Audsley's algorithm
2) Using a tie breaker to choose the task to assign at a given priority level, based on critical scaling factor (Lehoczky 1989)

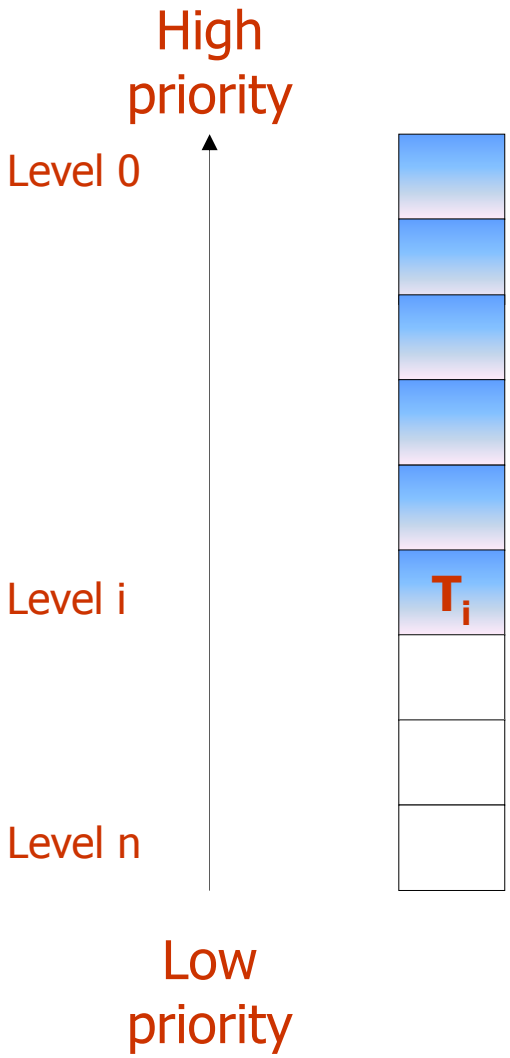
Scaling factor

Maximum reduction factor which can be applied to the processor speed in such a way that the system is still schedulable

Task	Period	Deadline	Priority	Execution Time
	5	5	+	1 → 2.5
	5	5	-	1 → 2.5



Scaling factor of a task



λ_i = **Scaling Factor of Ti and higher priority tasks**

Vestal's algorithm

Task	Pi	Di	Li	Ci(1)	Ci(2)	Priority
1	164	104	1	7	17	2
2	89	44	2	4	4	0
3	191	80	1	12	16	1
4	283	283	2	85	85	3

Priority \ Task	Task 1	Task 2	Task 3	Task 4
3	✗	✗	✗	1.69
2	3.87	1.19	3.47	
1		2.2	5	
0		11		

Tie breaker: scaling factor

Our contributions

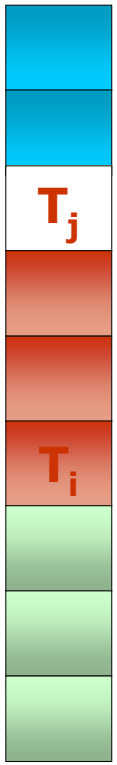
- Audsley's algorithm is optimal
- Vestal's algorithm maximizes the scaling factor
  - Processor speed modulation
  - Reducing power consumption

Priority \ Task	Task 1	Task 2	Task 3	Task 4
3	✗	✗	✗	1.69
2	3.87	1.19	3.47	
1		2.2	5	
0		11		

➔Maximum scaling factor: 1.69

Proof Sketch

High priority



Low priority

Transformation

If the scaling factor of T_j at the priority level of the task Ti is greater than the one of T_i, then affecting the Ti priority to T_j can only increase the scaling factor of the task system.

The scaling factor is unchanged since task of lower priority do not affect these ones

The scaling factor is unchanged since it depends only of the set of higher priority task and not of their relative priorities

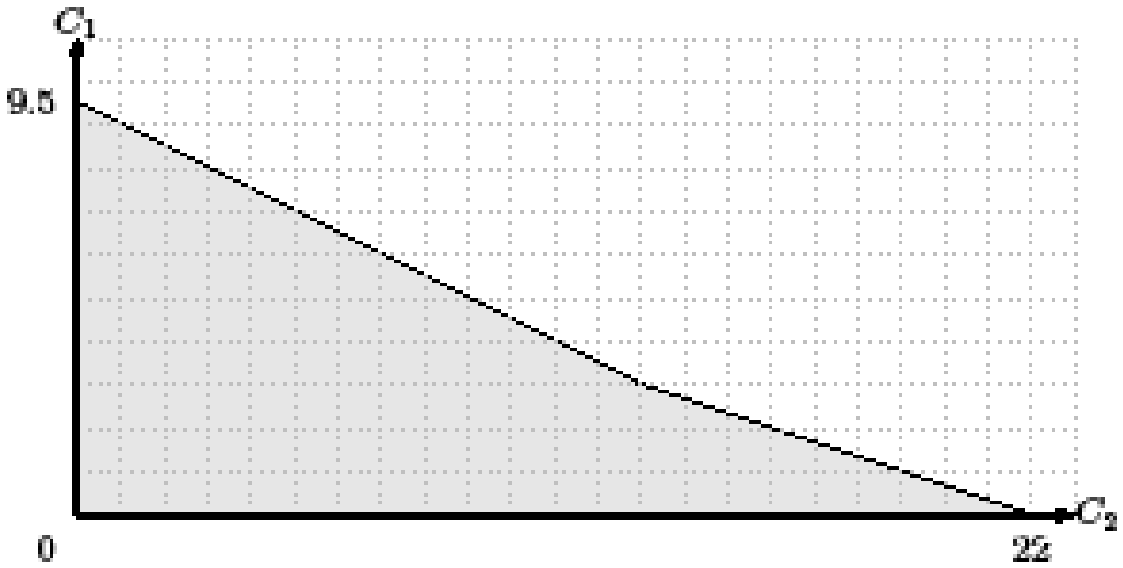
The schaling factor is increased since the priority of each task increased

Sensitivity Analysis

Sensitivity analysis

- Aim : C-Space

Task	Ti	Di
1	9.5	9.5
2	22	22



Existing works

- Bini (*Sensitivity analysis for fixed-priority real-time systems*)
  - Perform a sensitivity analysis on the C-Space, in a given direction:
    - Subset of tasks
    - Ponderation
  - Particular cases
    - Scaling factor (all task with same ponderation)
    - One task (one task)

Our works

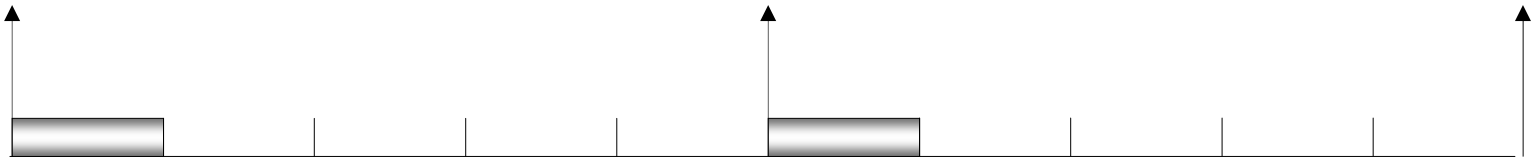
- Adaptation of the Bini's algorithm for multiple criticality tasks system
  - Idea: perform one sensitivity analysis per criticity level

- Example: sensitivity analysis on one task

Sensitivity analysis on one task

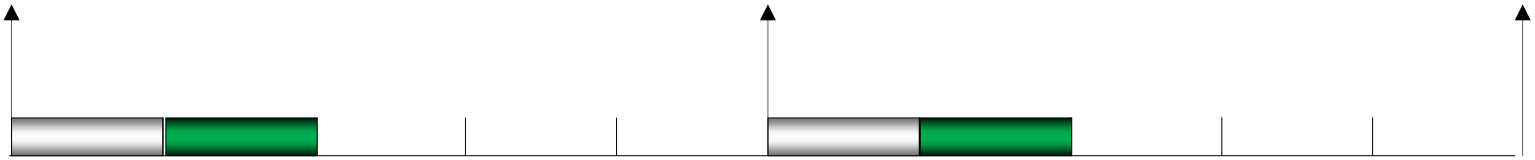
Task	Period	Deadline	Priority	Execution Time
	5	5	High	1
	5	5	Low	1

Grey task



$$\delta C_1 = 4$$

Green task



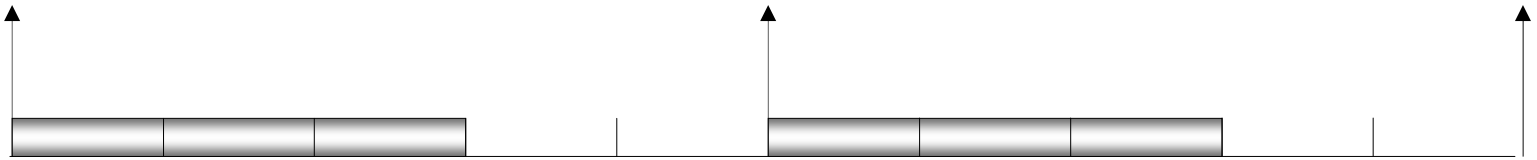
$$\delta C_1 = 3$$

$$\delta C_1 = \min(4,3) = 3$$

Sensitivity analysis on one task

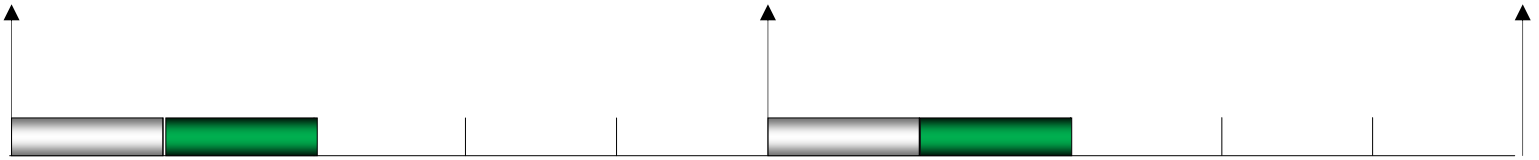
Task	Period	Deadline	Priority	Criticity	Execution Time 1	Execution Time 2
	5	5	High	2	1	3
	5	5	Low	1	1	1

Grey task



$$\delta C_1 = 2$$

Green task



$$\delta C_1 = 3$$

$$\delta C_1 = \min(2,3) = 2$$

$$\delta C_1(1) = \min(3) = 3$$

$$\delta C_1(2) = \min(2) = 2$$

Normalization step

Aim: sensitivity analysis on T2

Task	Ti	Di	Li	Ci(1)	Ci(2)
1	137	65	1	9	29
2	286	139	2	86	86
3	248	168	1	32	160

Results:

$$\delta C_2^{\max}(1) = 32$$

$$\delta C_2^{\max}(2) = 22$$

Task	Ti	Di	Li	Ci(1)	Ci(2)
1	137	65	1	9	29
2	286	139	2	108 ←	108
3	248	168	1	32	160

Need of a normalization step

Conclusion

- Our works
  - Audsley's algorithm is optimal
  - Optimality property of Vestal's algorithm
  - Adaptation of sensitivity analysis of Bini
- Perspectives
  - Adapting our previous work to the multiple criticality model
    - Algorithm minimizing the number of processors (Dorin2008)
  - Sensitivity analysis