The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems

Sanjoy Baruah[1] Vincenzo Bonifaci[2]
Gianlorenzo D'Angelo[3]
Haohan Li[1] Alberto Marchetti-Spaccamela[4] Suzanne Van Der Ster[5]
Leen Stougie[5]

[1]The University of North Carolina, Chapel Hill, USA.

[2]IASI — CNR, Italy.

[3]MASCOTTE project — INRIA, France

[4]Sapienza University of Rome, Italy.

[5]Vrije Universiteit, The Netherlands.

Mixed-Criticality scheduling — motivations

In Safety-critical embedded systems, there is an increasing trend towards implementing multiple functionalities upon a single shared computing platform

Examples: Integrated Modular Avionics (IMA) and AUTomotive Open System ARchitecture (AUTOSAR)

This can force tasks of different importance (i.e. criticality) to share a processor and interfere with each other

Mixed-Criticality scheduling — motivations

Example

In **unmanned aerial vehicles** functionalities are classified into two levels of criticality:

Level 1: mission-critical functionalities

- Certified by clients or vendors
- Less rigorous: consider low WCET
- Interested in Level 2 functionalities

Level 2: flight-critical functionalities

- Certified by civilian Certification Authorities (CA)
- CAs are very conservative: consider high *Worst Case Execution Time* (WCET)
- CAs are not interested in Level 1 functionalities

Each CA has its own rules to determine the WCET of a job

The WCET of the same job of a flight critical functionality has two values:

- One lower value: WCET if we consider mission critical functionalities
- One higher value: WCET if we restrict to flight critical functionalities

In this paper

We analyze an algorithm EDF-VD for scheduling mixed-criticality task systems proposed in [Baruah et al, European Symposium on Algorithms 2011]

- we show that its speed-up factor is $4/3$ (instead of ϕ)
- we show that it is optimal w.r.t. speed-up factor
- we show how to implement it in logarithmic computational time
- we derive utilization bounds and simulate its behavior

Outline

Outline

Sporadic Task Systems

A *Sporadic Task System* consists of a set of tasks $\tau_1, \tau_2, \ldots, \tau_n$

A task $\tau_i = (c_i, d_i, p_i)$ generates a potentially infinite sequence of jobs, where:

- c_i is the execution requirement
- d_i is the relative deadline: time between a job's arrival and its deadline
- p_i is the period: minimum time between two successive arrivals of jobs from this task

In an implicit-deadline tasks system it holds that $d_i = p_i$ for all tasks

Mixed-Criticality Sporadic Task Systems

A *Mixed-Criticality Sporadic Task System* with 2 levels consists of a set of tasks $\tau_1, \tau_2, \ldots, \tau_n$

Each task $\tau_i = (\chi_i, c_i(LO), c_i(HI), T_i)$ generates a potentially infinite sequence of jobs, where:

$\chi_i \in \{LO, HI\}$ is the criticality level

$c_i(LO)$ is the execution requirement at level LO

$c_i(HI)$ is the execution requirement at level HI

T_i is the relative deadline and period

We assume $c_i(LO) \leq c_i(HI)$

Mixed-Criticality Sporadic Task Systems

The amount of execution time a that job of task τ_i needs is not known, but discovered by executing the job until it signals completion. A collection of realized values for the execution time is called a behavior

By executing the tasks, we learn in which level the system is. This may change over time

When the system is in level HI we need to process only the tasks that are of criticality level HI, the tasks of criticality level LO are omitted

Utilization

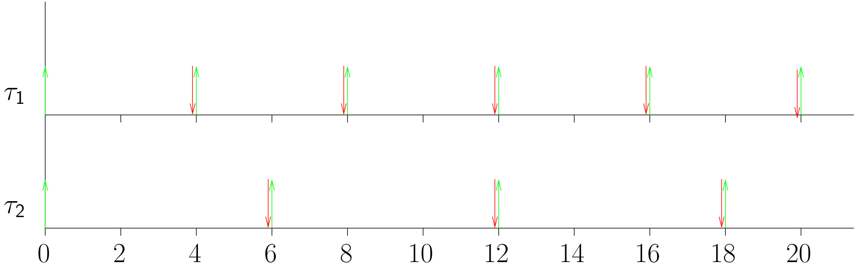
For $x, y \in \{LO, HI\}$,

$$U_x^y = \sum_{\chi_i = x} \frac{c_i(y)}{T_i}$$

For example, U_{HI}^{LO} is the utilization of HI criticality tasks, assuming a level-LO behavior

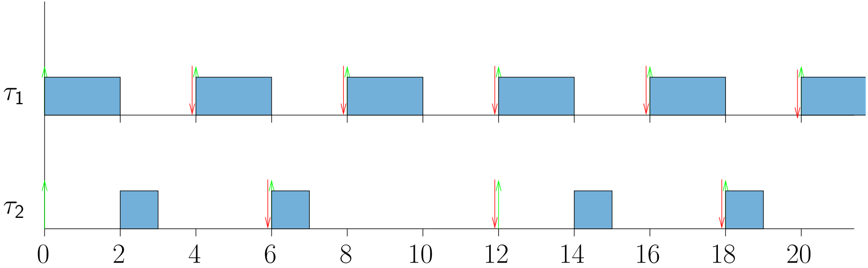
Example

τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	$1/2$	
τ_2	1	5	6	2	$1/6$	$5/6$
Total					$2/3$	$5/6$



Example

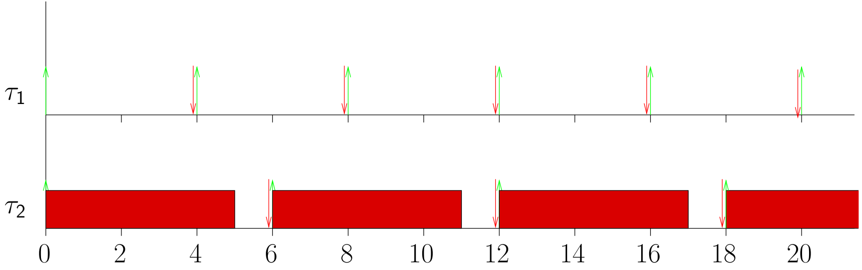
τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	1/2	
τ_2	1	5	6	2	1/6	5/6
Total					2/3	5/6



LO-criticality behavior

Example

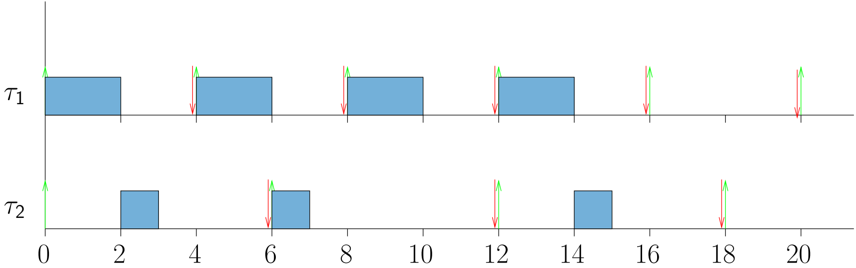
τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	1/2	
τ_2	1	5	6	2	1/6	5/6
Total					2/3	5/6



HI-criticality behavior

Example

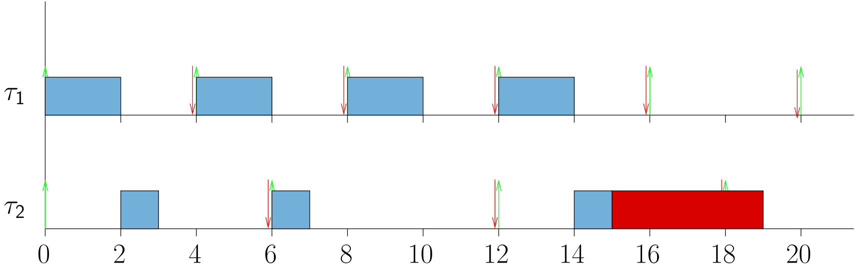
τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	$1/2$	
τ_2	1	5	6	2	$1/6$	$5/6$
Total					$2/3$	$5/6$



HI-criticality behavior

Example

τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	1/2	
τ_2	1	5	6	2	1/6	5/6
Total					2/3	5/6



HI-criticality behavior

Outline

Overview of the Algorithm EDF-VD

Preprocessing

1. Compute x as:

$$x \leftarrow \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}$$

2. If $\left(xU_{LO}^{LO} + U_{HI}^{HI} \leq 1\right)$ then declare success and compute

$$\hat{T}_i \leftarrow xT_i \text{ for each } \tau_i \text{ s.t. } \chi_i = HI$$

   Else declare failure and exit

$\hat{T}_i$: modified period

Overview of the Algorithm EDF-VD

Run-time

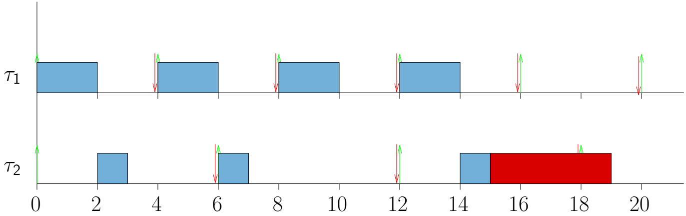
1. If the system is at level *LO* schedule according to EDF where *HI* criticality tasks τ_i have period $\hat{T}_i$
2. If some job executes beyond its *LO*-criticality WCET:
   - discard all *LO*-criticality jobs
   - schedule *HI*-criticality tasks according to EDF with actual periods

Example

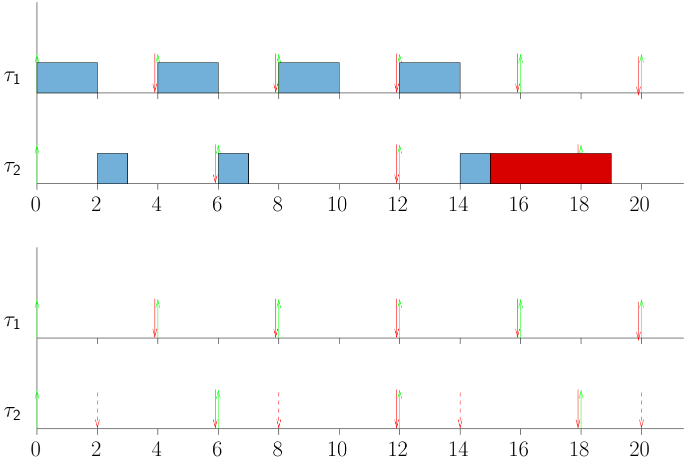
τ_i	$c_i(LO)$	$c_i(HI)$	T_i	χ_i	U^{LO}	U^{HI}
τ_1	2	2	4	1	1/2	
τ_2	1	5	6	2	1/6	5/6
Total					2/3	5/6

$$x = \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}} = \frac{1/6}{1 - 1/2} = \frac{1}{3}$$

$$xU_{LO}^{LO} + U_{HI}^{HI} = \frac{1}{3} \cdot \frac{1}{2} + \frac{5}{6} = 1$$

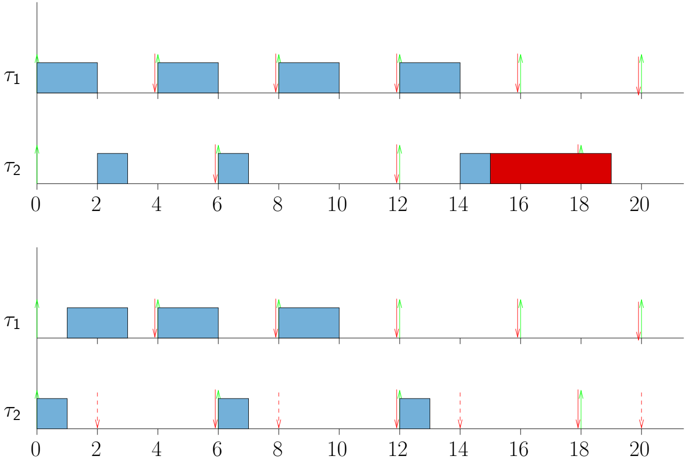


Example

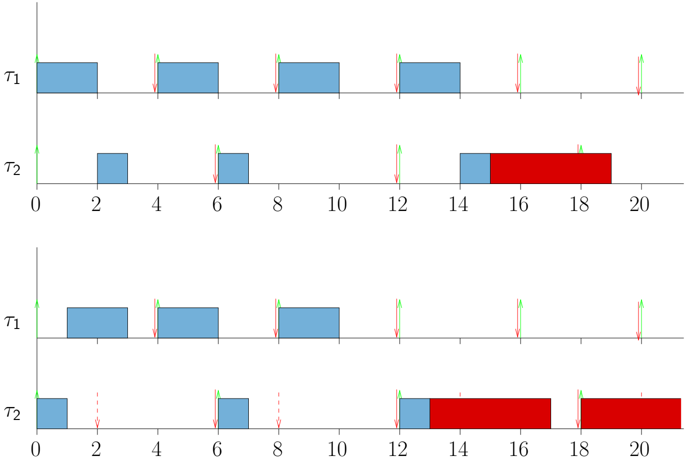


τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20
τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20

Example


τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20
τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20

Example



τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20
τ_1
τ_2
0 2 4 6 8 10 12 14 16 18 20

Preprocessing

Theorem

The following condition is sufficient for ensuring that EDF-VD successfully schedules all LO-criticality behaviors:

$$x \geq \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}$$

EDF-VD chooses the smallest value of x such that the above condition is satisfied

Theorem

The following condition is sufficient for ensuring that EDF-VD successfully schedules all HI-criticality behaviors:

$$xU_{LO}^{LO} + U_{HI}^{HI} \leq 1$$

Dispatching

1. $\Gamma = LO$ (criticality level indicator)
2. while $(\Gamma \equiv LO)$
   1. when some job of task τ_i arrives at time t
      - if $\chi_i \equiv LO$ the deadline is $t + T_i$
      - if $\chi_i \equiv HI$ the deadline is $t + \hat{T}_i$
   2. at each instant the job with the earliest deadline is scheduled
   3. if the current scheduled job executes for more than its LO-criticality WCET

$$\Gamma \leftarrow HI$$

3. Once $(\Gamma \equiv HI)$
   1. add $T_i - \hat{T}_i$ to the deadline of active jobs of task τ_i
   2. when some job of task τ_i arrives at time t, its deadline is $t + T_i$
   3. LO-criticality jobs do not receive any execution

Dispatching – a $O(\log(n))$ implementation

We have to implement three operation corresponding to three events:

1. Arrival of a job
2. Completion of a job
3. Switching of Γ to HI

We use two priority queues:

- Q_{LO}
- Q_{HI}

J_c: current executed job
We use a timer that indicate whether J_c executed more than its LO-criticality WCET

Dispatching – a $O(\log(n))$ implementation

Arrival of a job of task τ_i at time t_c

- If $\chi_i = LO$, the job is inserted only in Q_{LO}
- If $\chi_i = HI$, the job is inserted in both Q_{LO} (with modified deadline) and Q_{HI} (with actual deadline)
- If the new job is the one with the earliest deadline, set the timer to $t_c + C_i(LO)$

Completion of job J_c at time t_c

- Delete J_c from both Q_{LO} and Q_{HI}
- Reset the timer to t_c + the LO-criticality WCET of the minimum job in Q_{LO}

The timer goes off

We schedule according to Q_{HI}

Outline

Comparison with Worst-Case Reservation

The worst case reservation approach (WCR) maps a mixed-criticality task system into a traditional task system

$$\tau_i = (\chi_i, c_i(LO), c_i(HI), T_i) \rightarrow \tau_i' = (c_i(\chi_i), p_i)$$

and schedules according to regular EDF

Necessary condition

$$U_{LO}^{LO} + U_{HI}^{HI} \leq 1$$

Theorem

Any task system that is correctly scheduled using WCR is also scheduled by EDV-VD

Proof.

As $U_{LO}^{LO} + U_{HI}^{HI} \leq 1$ and $x \leq 1$, then $xU_{LO}^{LO} + U_{HI}^{HI} \leq 1$ □

System where the *LO*-criticality WCET of *HI*-criticality task is 0 ($U_{HI}^{LO} = 0$)

In this systems $x = 0$ and then

- the sufficient condition is
$$U_{LO}^{LO} \leq 1$$
$$U_{HI}^{HI} \leq 1$$
i.e. *LO*-criticality and *HI*-criticality behaviors are separately schedulable
- $\hat{T}_i = 0$ for each τ_i s.t. $\chi_i = HI$
i.e. *HI*-criticality tasks always have earliest deadline

Speed-up bounds

The Speed-up factor of an algorithm $\mathcal{A}$ is the smallest real number f such that any task system that is clairvoyantly schedulable on a unit speed processor is schedulable by $\mathcal{A}$ on a f-speed processor

Theorem

The speed-up factor of EDF-VD is $\frac{4}{3}$

Theorem

No non-clairvoyant algorithm for scheduling dual-criticality systems can have a speed-up factor smaller than $\frac{4}{3}$

Utilization bounds

The two given sufficient condition are:

$$x \geq \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}}$$

$$x \leq \frac{1 - U_{HI}^{HI}}{U_{LO}^{LO}}$$

that are satisfied if and only if

$$\frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}} \leq \frac{1 - U_{HI}^{HI}}{U_{LO}^{LO}}$$

that is

$$U_{HI}^{HI} \leq 1 - U_{HI}^{LO} \left(\frac{U_{LO}^{LO}}{1 - U_{LO}^{LO}} \right)$$

Utilization bounds

U_{HI}^{LO} \ U_{LO}^{LO}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.98	0.97	0.9	0.93	0.90	0.85	0.76	0.60	0.10
0.2	0.97	0.95	0.914	0.86	0.80	0.70	0.53	0.20	
0.3	0.96	0.92	0.87	0.80	0.70	0.55	0.30		
0.4	0.95	0.90	0.82	0.70	0.60	0.40			
0.5	0.94	0.87	0.78	0.60	0.50				
0.6	0.93	0.85	0.74	0.50					
0.7	0.92	0.82	0.70						
0.8	0.91	0.80							
0.9	0.90								

Outline

Input instances

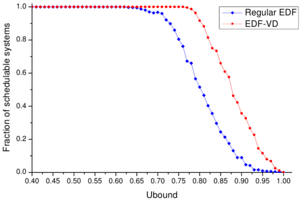
Randomly generated instances with the following parameters:

- $U_{\text{bound}} = \max\{U_{LO}^{LO} + U_{HI}^{LO}, U_{HI}^{HI}\}$: larger utilization of the task system
- $[U_L, U_U]$: utilization of the a task
- $[Z_L, Z_U]$: ratio between *HI*-criticality utilization and *LO*-criticality utilization of a task
- P: probability that a task is an *HI*-criticality task

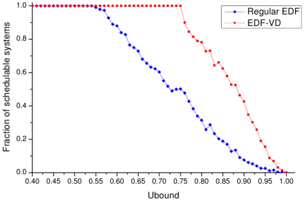
We generate the tasks one by one until the required utilization parameters are met

We generated 1000 instances and measured the fraction of instances which are schedulable by algorithms EDF-VD and Regular-EDF (WCR approach)

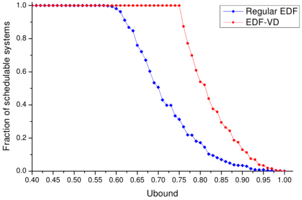
Results



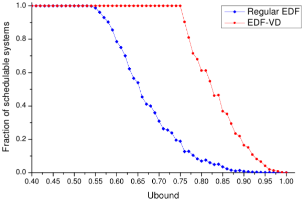
$U_L = 0.02,\ U_U = 0.2,\ Z_L = 1,\ Z_U = 2,\ P = 0.5$



$U_L = 0.02,\ U_U = 0.2,\ Z_L = 1,\ Z_U = 8,\ P = 0.5$



$U_L = 0.02,\ U_U = 0.2,\ Z_L = 1,\ Z_U = 4,\ P = 0.5$

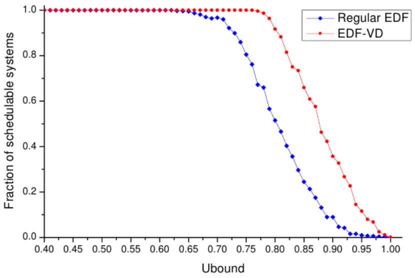


$U_L = 0.02,\ U_U = 0.2,\ Z_L = 1,\ Z_U = 8,\ P = 0.3$

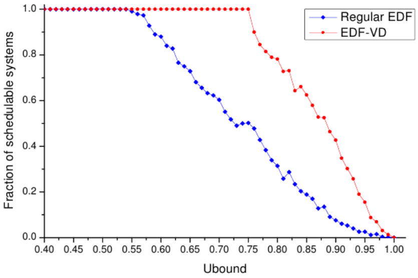
In any case, the system is schedulable by Regular-EDF if $U_{\text{bound}} \leq 0.5$ and by EDF-VD if $U_{\text{bound}} \leq 0.75$

Results

The improvement is more significant if the ratio between *HI*-criticality utilization and *LO*-criticality utilization increases

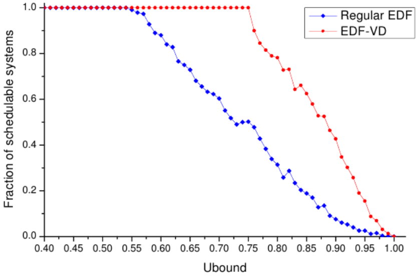


$Z_U = 2$

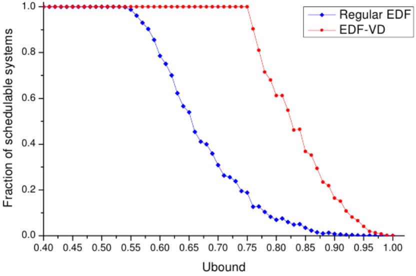


$Z_U = 8$

It is even more significant if the ratio between the *HI*-criticality utilization and *LO*-criticality utilization of a the system approaches 1



1.6



1.06

Outline

Conclusion

We analyzed algorithm EDF-VD:

- we have shown that its speed-up factor is $4/3$
- we have shown that no non-clairvoyant algorithm can can have a speed-up factor smaller than $4/3$
- we have shown how to implement it in logarithmic computational time
- we have derived utilization bounds and simulate its behavior

Thank you for your attention