Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation by Completion Rate

Zhishan Guo[1], Kecheng Yang[2], **Sudharsan Vaidhun**[1],
Samsil Arefin[3], Sajal K. Das[3], Haoyi Xiong[4]

[1]Department of Electrical & Computer Engineering, University of Central Florida
[2]Department of Computer Science, Texas State University
[3]Department of Computer Science, Missouri University of Science & Technology
[4]Baidu Inc., Beijing, China

December 14, 2018

Mixed-Criticality (MC) Systems

Safety certification requirement

All safety critical tasks must **complete execution** within its **deadline**.

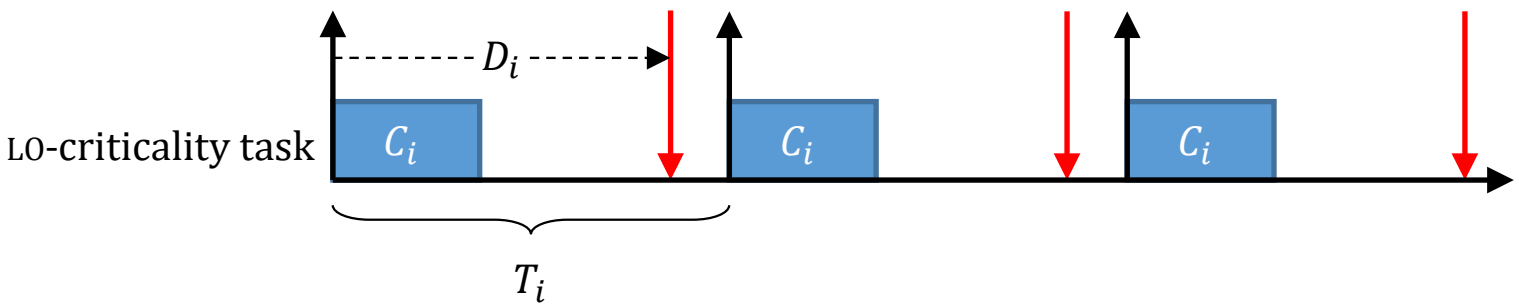
Non-safety critical tasks need not be guaranteed for safety certification, but rather by mission certification.

In general, different certification levels certify different levels of tasks, with safety critical being the highest.

Mixed-Critical (MC) Task Model

- System model of lo-criticality tasks:

$$_i = fC_i; T_i; D_i; \ _i g; \ 8 \ _i \ 2 \ _{\text{lo}};$$



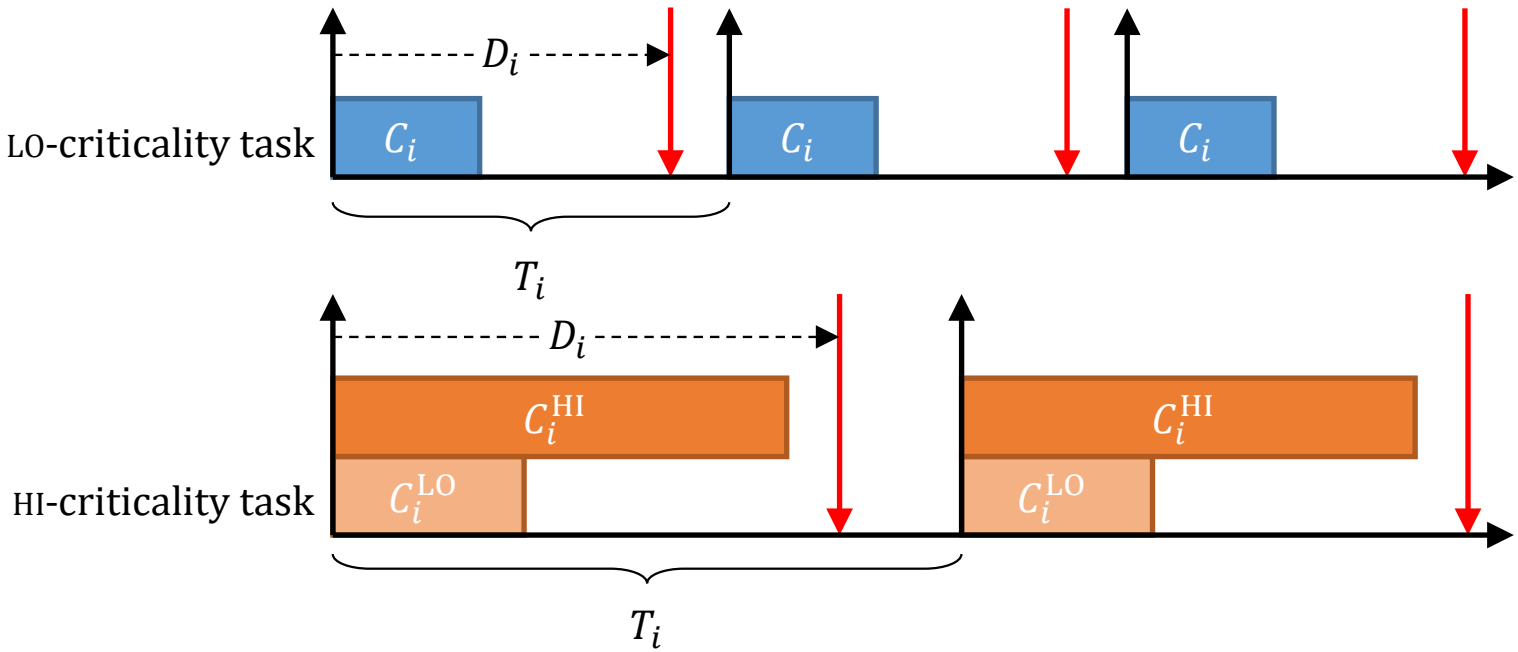
Mixed-Critical (MC) Task Model

- System model of lo-criticality tasks:

$$_i = fC_i; T_i; D_i; \ _i g; \ 8 \ _i \ 2 \ _{\text{lo}};$$

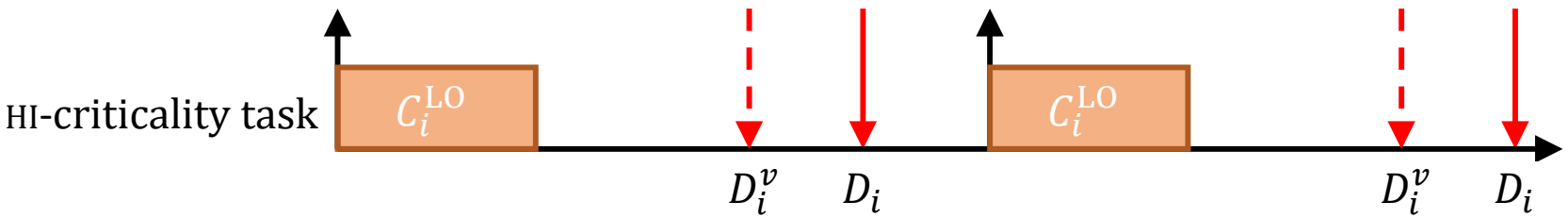
- System model of hi-criticality tasks:

$$_i = fC_i^{\text{lo}}; C_i^{\text{hi}}; T_i; D_i; \ _i g; \ 8 \ _i \ 2 \ _{\text{hi}};$$



EDF-VD Algorithm

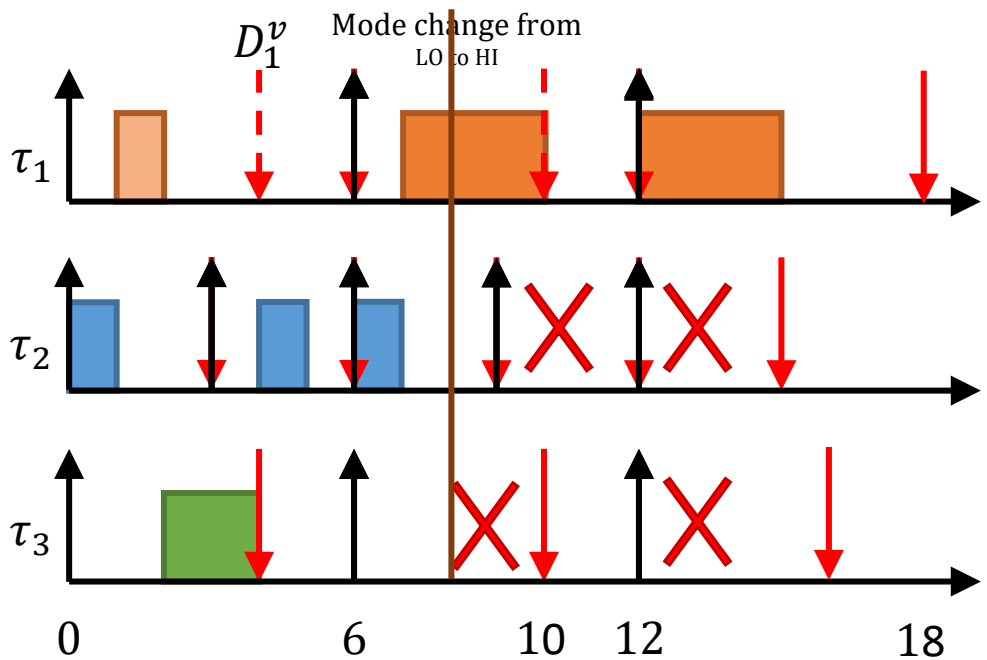
Earliest Deadline First with Virtual Deadline (EDF-VD) algorithm[1]



[1] S. Baruah et al. "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems". In: *Proceedings of the 24th Euromicro Conference on Real-Time Systems*. 2012.

Example Schedule

Task ID	i	C_i (C_i^{lo})	C_i^{hi}	T_i	D_i	D_i^v
1	hi	1	3	6	6	4
2	lo	1	-	3	3	-
3	lo	2	-	6	4	-



Graceful Degradation

LO-criticality tasks are not non-critical.

Graceful Degradation

LO-criticality tasks are not non-critical.

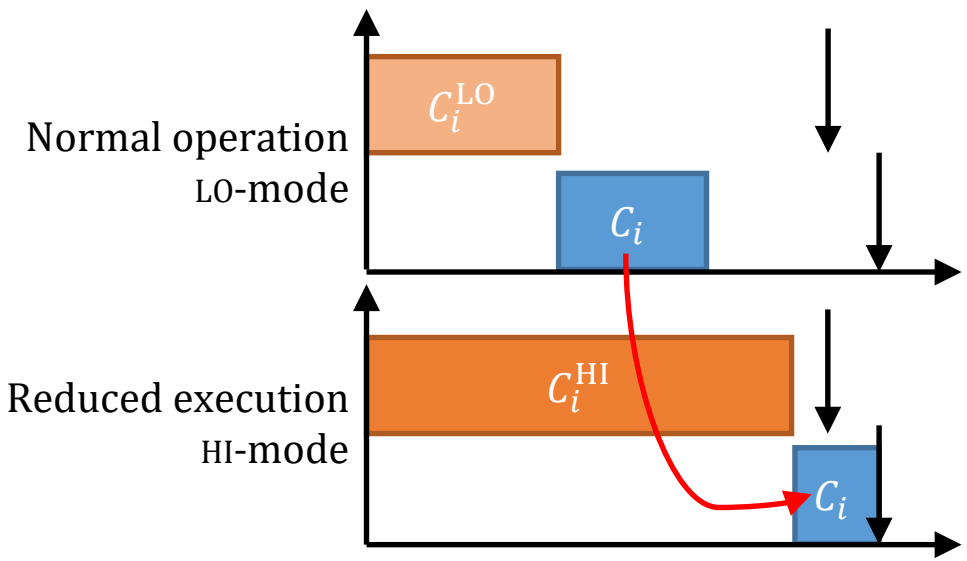
Scenario 1 : In control systems, a reduced rate of sensing can still produce stable control.
Scenario 2 : In signal processing, losing few data packets can still produce decent sound/video.

Providing degraded yet guaranteed service is better than providing no guarantees.

Limitations of Existing Works on Graceful Degradation for Mixed Criticality

Utilization based degradation:[2]

- Reduced execution time - imprecise computation model

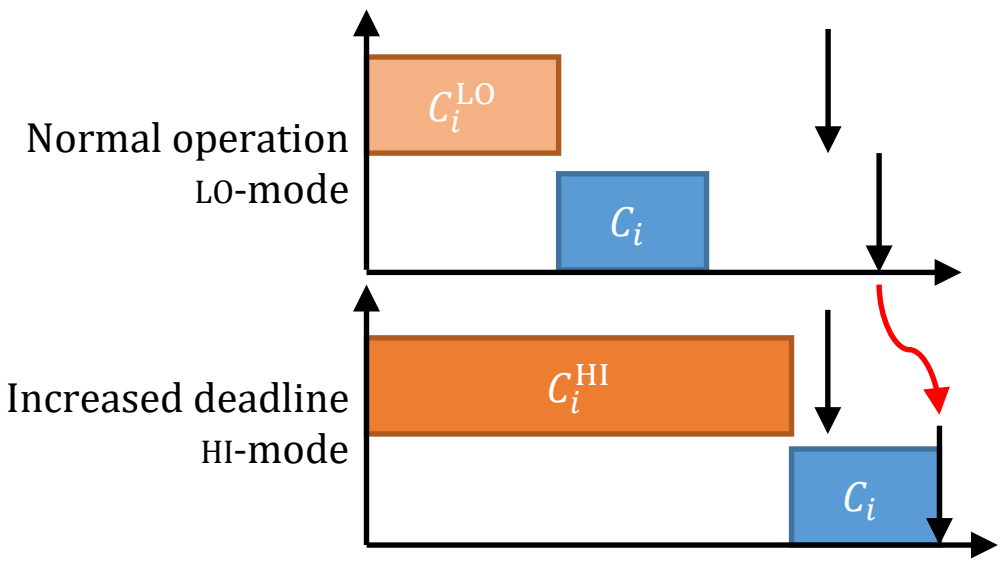


[2] Di Liu et al. "EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees". In: *Proceedings of the 37th IEEE Real-Time Systems Symposium*. 2016.

Limitations of Existing Works on Graceful Degradation for Mixed Criticality

Utilization based degradation:[2]

- Reduced execution time - imprecise computation model
- Increased deadline - delayed responses based on longer period



[2] Liu et al., "EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees".

Limitations of Existing Works on Graceful Degradation for Mixed Criticality

Asymptotic rate guarantees:[3;4]

- No guarantee for short interval of time
- Example:
  - 1-out-of-4 instances scheduled and 1000-out-of-4000 instances scheduled the same percent of tasks (25%) asymptotically.
  - However, in the second case 3000 consecutive instances can be dropped.

[3] M. S. Branicky, S. M. Phillips, and Wei Zhang. "Scheduling and feedback co-design for networked control systems". In: *Proceedings of the 41st IEEE Conference on Decision and Control*. 2002.

[4] Rupak Majumdar, Indranil Saha, and Majid Zamani. "Performance-aware Scheduler Synthesis for Control Systems". In: *Proceedings of the 9th ACM International Conference on Embedded Software*. ACM, 2011.

Graceful Degradation

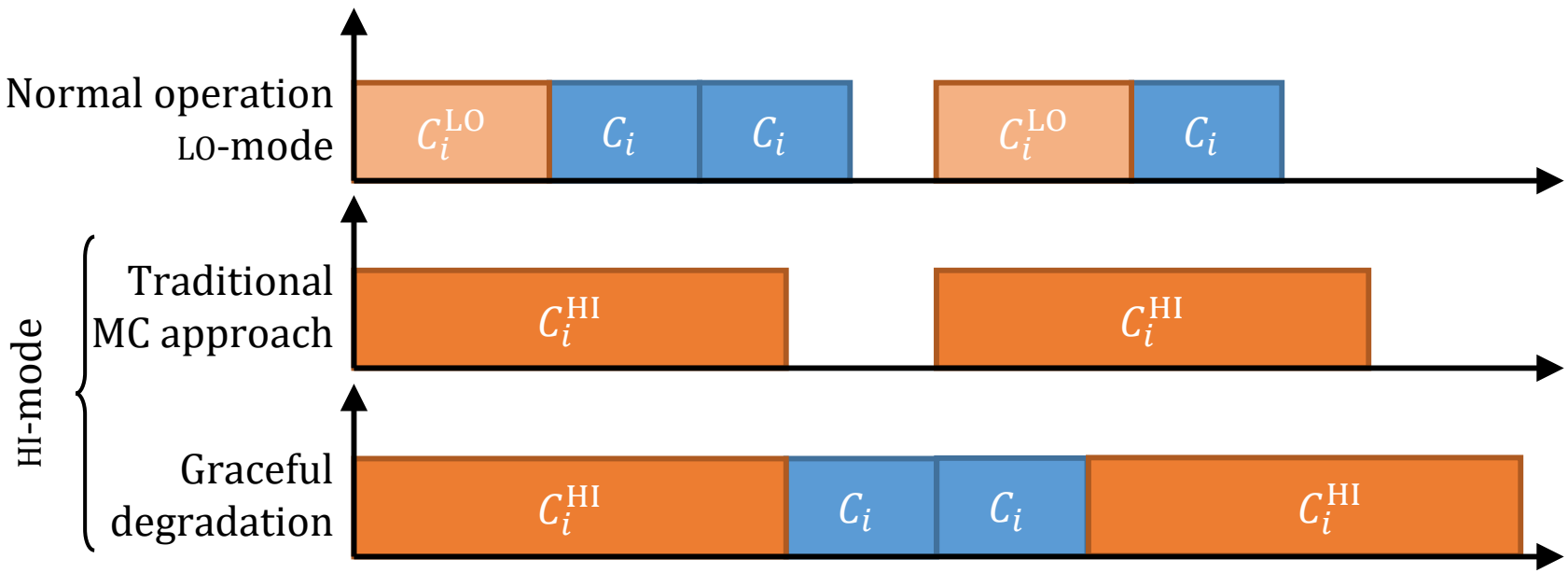
Goals:

- Maintain limited functionality even under failure
- Provide some guarantees to LO tasks even in HI mode

Graceful Degradation

Goals:

- Maintain limited functionality even under failure
- Provide some guarantees to LO tasks even in HI mode



Guaranteed Graceful Degradation to lo Criticality Tasks

System model:

$$8\ _i\ 2\ \ _{\text{lo}};\quad \ _i = fC_i; T_i; D_i; r_i;\quad \ _i g;$$
$$8\ _i\ 2\ \ _{\text{hi}};\quad \ _i = fC_i^{\text{lo}}; C_i^{\text{hi}}; T_i; D_i;\quad \ _i g:$$

r_i is the cumulative completion rate

Guaranteed Graceful Degradation to lo Criticality Tasks

System model:

$$\forall\ i \in \tau_{\text{lo}};\quad \tau_i = \{C_i; T_i; D_i; r_i;\ \ \}_i;$$
$$\forall\ i \in \tau_{\text{hi}};\quad \tau_i = \{C_i^{\text{lo}}; C_i^{\text{hi}}; T_i; D_i;\ \ \}_i:$$

r_i is the cumulative completion rate

Correctness criteria:

- In lo mode, all tasks will receive an execution time of C_i^{lo}

Guaranteed Graceful Degradation to lo Criticality Tasks

System model:

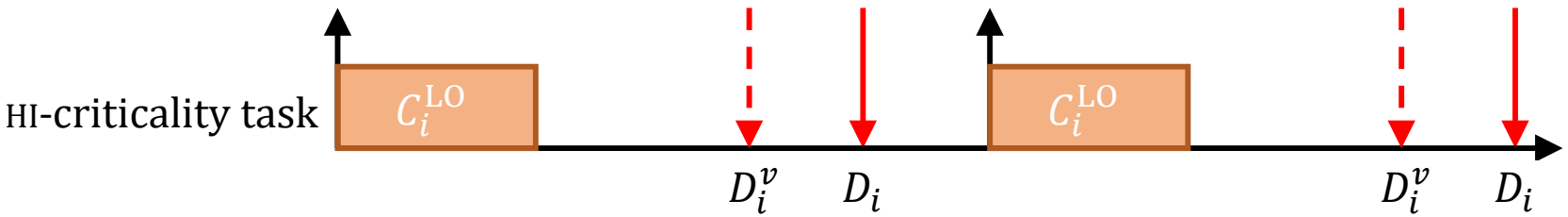
$$8\ _i\ 2\ \ _{\text{lo}};\quad \ _i = fC_i; T_i; D_i; r_i;\quad \ _i g;$$
$$8\ _i\ 2\ \ _{\text{hi}};\quad \ _i = fC_i^{\text{lo}}; C_i^{\text{hi}}; T_i; D_i;\quad \ _i g:$$

r_i is the cumulative completion rate

Correctness criteria:

- In lo mode, all tasks will receive an execution time of C_i^{lo}
- In hi mode, for any $N\ 2\ \mathbb{Z}^+$, at least $dr_i\quad Ne$-out-of-N lo jobs will be guaranteed to receive a full execution time of C_i; hi jobs will receive an execution time of C_i^{hi}

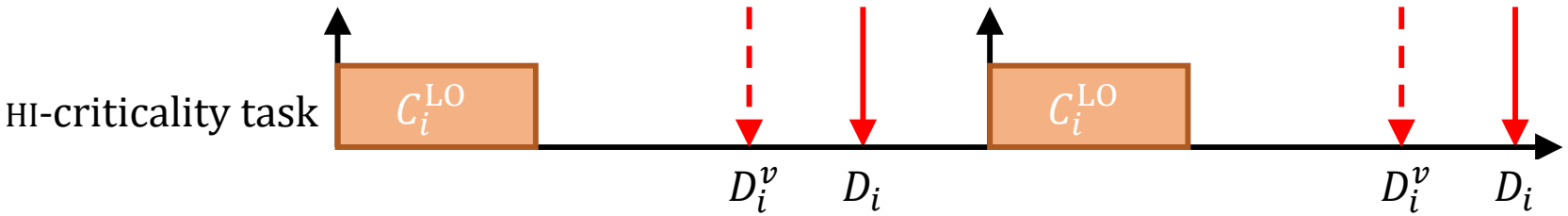
EDF-GVD Algorithm - lo-mode



Adapted from Earliest Deadline First with Virtual Deadline (EDF-VD) algorithm[5].

[5] Baruah et al., "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems".

EDF-GVD Algorithm - lo-mode



Adapted from Earliest Deadline First with Virtual Deadline (EDF-VD) algorithm[5].

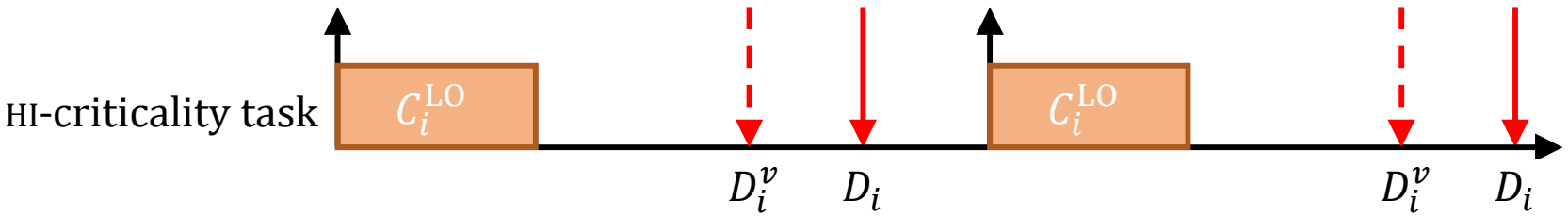
Virtual deadline setting

A simple heuristics is used to test the schedulability

$$D_i^v = \frac{C_i^{lo}}{C_i^{hi}} D_i; \ 8 \ _i \ 2 \ \ _{hi}:$$

[5] Baruah et al., "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems".

EDF-GVD Algorithm - lo-mode



Adapted from Earliest Deadline First with Virtual Deadline (EDF-VD) algorithm[5].

Virtual deadline setting

If simple heuristic does not work, q 2 [0; 1] is determined using binary search up to an error margin

$$D_i^v = q \quad D_i; \ 8 \ _i \ 2 \quad _{\text{hi}}:$$

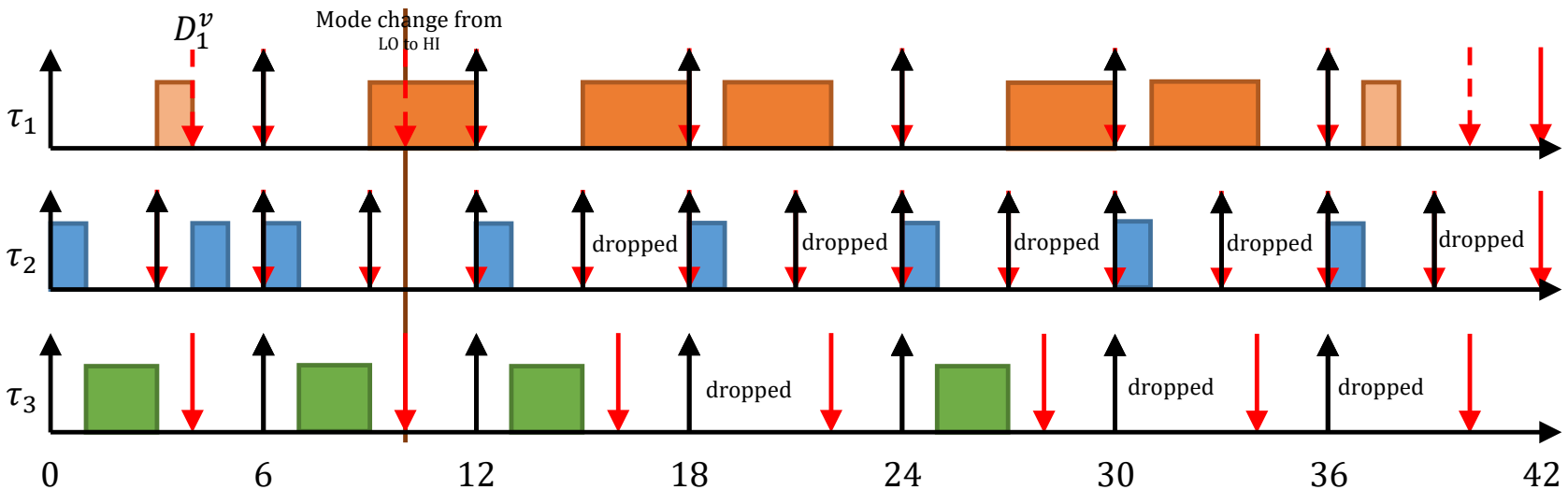
[5] Baruah et al., "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems".

EDF-GVD Algorithm - hi mode

If any hi-criticality task does not signal completion by its C_i^{lo}, there is a system wide mode switch.

Example Schedule

Task ID	i	C_i (C_i^{lo})	C_i^{hi}	T_i	D_i	D_i^v	r_i	i
1	hi	1	3	6	6	4	-	-
2	lo	1	-	3	3	-	0:5	$f10g^1$
3	lo	2	-	6	4	-	0:4	$f10100g^1$



EDF-GVD Algorithm - hi mode

If any hi-criticality task does not signal completion by its C_i^{lo}, there is a system wide mode switch.

- hi-criticality tasks are scheduled for C_i^{hi} execution time.

EDF-GVD Algorithm - hi mode

If any hi-criticality task does not signal completion by its C_i^{lo}, there is a system wide mode switch.

- hi-criticality tasks are scheduled for C_i^{hi} execution time.
- lo-criticality tasks are scheduled based on an admission control procedure.
  1. Static admission control for rational values of r_i
  2. Dynamic admission control for irrational values of r_i

Static Admission Control

$$r_i = \frac{p}{q} \qquad \text{pattern} \quad \text{of } q \text{ length, with } p \text{ no. of '1's}$$

$r_i = 0{:}4 = \frac{2}{5}$, $_i = f10100g^{1}$

actual r_i — $1, \frac{1}{2}, \frac{2}{3}, \frac{1}{2}, \frac{2}{5}, \frac{1}{2}, \frac{3}{7}, \frac{1}{2}, \frac{4}{9}, \frac{2}{5},$

Static Admission Control

$$r_i = \frac{p}{q} \qquad \text{pattern} \quad \text{of } q \text{ length, with } p \text{ no. of '1's}$$

$r_i = 0{:}4 = \frac{2}{5}$, $_i = f10100g^{1}$

actual r_i — $1, \frac{1}{2}, \frac{2}{3}, \frac{1}{2}, \frac{2}{5}, \frac{1}{2}, \frac{3}{7}, \frac{1}{2}, \frac{4}{9}, \frac{2}{5},$

Use cases:

1. Rational r_i value
2. Offline calculation

Dynamic Admission Control

Control the lo-tasks admission in hi mode dynamically

$r_i = 1/\sqrt{2} = 0.7071 \quad , \quad _i = 11101101110$

actual r_i — $1, 1, 1, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{5}{7}, \frac{3}{4}, \frac{7}{9}, \frac{4}{5}, \frac{8}{11},$

Dynamic Admission Control

Control the lo -tasks admission in hi mode dynamically

$r_i = 1 = \sqrt{2} = 0.7071 \quad , \quad _i = 11101101110$

actual r_i | $1; 1; 1; \frac{3}{4}; \frac{4}{5}; \frac{5}{6}; \frac{5}{7}; \frac{3}{4}; \frac{7}{9}; \frac{4}{5}; \frac{8}{11};$

Use cases:

1. Irrational r_i value
2. Low memory - pattern need not be stored

Admission Control Patterns

S = Maximum number of consecutive drops

$$S(r_i) = \frac{1}{r_i} \quad 1:$$

Task ID	r_i	Pattern $_i$	S_i
1	0.4	f 10100g^1	2
2	0.5	f 10g^1	1
3	0.625	f 11011010g^1	1
4	1= p 2	11101101110	1

Provides the system designer control over maximum number of consecutive drops.

Backward Mode Switch

Task ID	χ_i	C_i (C_i^{lo})	C_i^{hi}	T_i	D_i	D_i^v	r_i	ψ_i
1	hi	1	3	6	6	4	-	-
2	lo	1	-	3	3	-	0:5	$\{10\}^1$
3	lo	2	-	6	4	-	0:4	$\{10100\}^1$

Backward Mode Switch

Task ID	χ_i	C_i (C_i^{lo})	C_i^{hi}	T_i	D_i	D_i^v	r_i	σ_i
1	hi	1	3	6	6	4	-	-
2	lo	1	-	3	3	-	0:5	$f\,10 g^{1}$
3	lo	2	-	6	4	-	0:4	$f\,10100 g^{1}$

Backward Mode Switch

System switches back to lo mode when:

1. the processor idles in hi-criticality mode, and
2. the last instance of each hi-criticality task is finished within its lo-WCET

Backward Mode Switch

System switches back to lo mode when:

1. the processor idles in hi-criticality mode, and
2. the last instance of each hi-criticality task is finished within its lo-WCET

The maximum time after the last hi critical task instance to return to lo-mode is

$$L_1 = \frac{\sum_{i \in hi} (2C_i^{lo}) + \sum_{i \in lo} (C_i + 2r_i - C_i)}{1 - \sum_{i \in hi} (C_i^{lo}/T_i)} \sum_{i \in lo} (r_i - C_i/T_i)$$

Schedulability Analysis

Demand bound function

$$8` \quad 0; \sum_{¡\,2} \quad dbf( \ ¡ \ ; `) \quad `:$$

Schedulability Analysis

Demand bound function

$$\forall \ell \geq 0, \quad \sum_{\tau_i \in \tau} \mathrm{dbf}(\tau_i, \ell) \leq \ell :$$

$\mathrm{dbf}^{B}_{A}(\tau_i, \ell)$: dbf of A-criticality tasks in B-criticality mode

$\mathrm{dbf}^{lo}_{lo}(\tau_i, \ell); \mathrm{dbf}^{lo}_{hi}(\tau_i, \ell); \mathrm{dbf}^{hi}_{lo}(\tau_i, \ell); \mathrm{dbf}^{hi}_{hi}(\tau_i, \ell)$

Schedulability Test A - for lo -mode

Condition A:

$$\forall \ell \quad 0 < \ell : \sum_{\tau_i \in \tau_{lo}} dbf^{lo}_{lo}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{hi}} dbf^{lo}_{hi}(\tau_i, \ell) \leq \ell$$

[6] Pontus Ekberg and Wang Yi. "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems". In: Real-Time Systems 50 (2014), pp. 48–86.

Schedulability Test A - for lo-mode

Condition A:

$$\forall \ell \quad 0; \quad \sum_{i \in lo} dbf^{lo}_{lo}(\tau_i; \ell) + \sum_{i \in hi} dbf^{lo}_{hi}(\tau_i; \ell) \quad \ell :$$

Lemma 3: If $\sum_{i \in lo}(C_i/T_i) + \sum_{i \in hi}(C^{lo}_i/T_i)$ c. and $c < 1$ Then, Condition A[6] holds, if it is true for all ℓ such that

$$\ell < \frac{c}{1 - c} \max\{\max_{i \in lo}\{T_i - D_i\}, \max_{i \in hi}\{T_i - D^v_i\}\}$$

[6] Ekberg and Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems".

Schedulability Test B - for hi-mode

Condition B:

$$\forall \ell > 0;\quad \sum_{i \in \tau_{lo}} dbf^{hi}_{lo}(\tau_i;\ell) + \sum_{i \in \tau_{hi}} dbf^{hi}_{hi}(\tau_i;\ell) \le \ell.$$

Lemma 4: If $\sum_{i \in \tau_{lo}} (r_i \cdot C_i/T_i) \le c_1$, $\sum_{i \in \tau_{hi}} (C^{hi}_i/T_i) \le c_2$, $c_1 < 1$, $c_2 < 1$, and $c_1 + c_2 < 1$.
Then, Condition B holds, if it is true for all ℓ such that

$$\ell < \frac{c_1 \cdot \max_{i \in \tau_{lo} \wedge r_i > 0} \{T_i - D_i + \frac{T_i}{r_i}\} + c_2 \cdot \max_{i \in \tau_{hi}} \{T_i - D_i + D^v_i\}}{1 - c_1 - c_2}.$$

Experiment via Simulation

Task set generation:

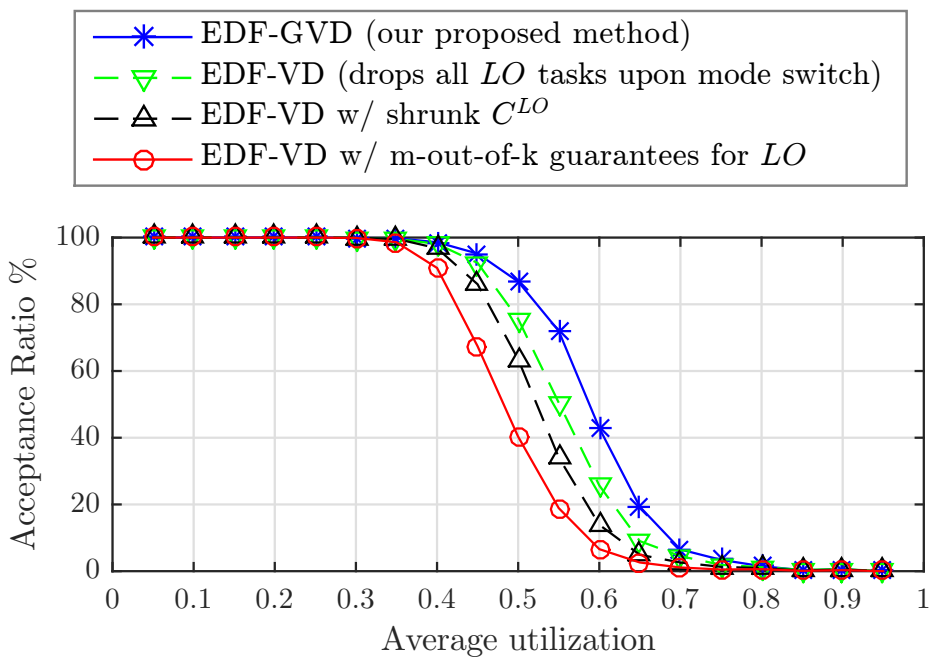
- $P_{\text{hi}} = 0.5$: probability of hi-tasks.
- C_i^{lo} $U[1; 10]$
- $R_{\text{hi}} = 4$: ratio of C_i^{hi} to C_i^{lo}
- $T_{max} = 200$: T_i $U[C_i(\ _i); T_{max}]$.
- $minDR$ $U[0.1; 0.9]$: $D_i =$ T_i; where is uniformly generated from the range $[minDR; 1]$.

Tasks are generated in a way so that each task set have a targeted average utilization U_{avg}.

Simulation results

Variation of acceptance ratio with average utilization

- Constrained deadline tasks - deadline randomly chosen in the range $[0.5; 1]$ period.
- 1000 tasksets each.

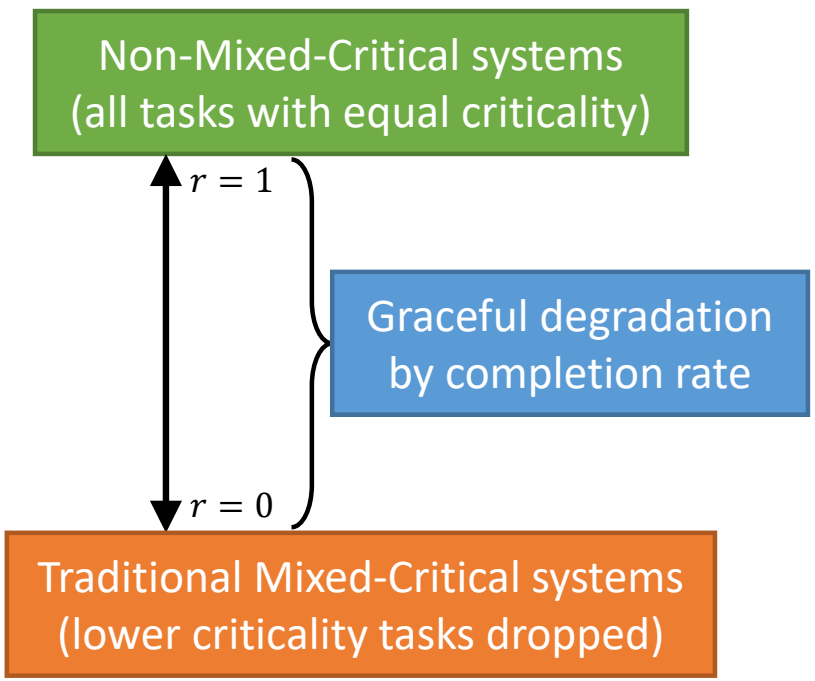


Conclusions

- Stronger correctness criteria - guaranteed service rate to non-safety-critical tasks in hi mode.

Conclusions

- Stronger correctness criteria - guaranteed service rate to non-safety-critical tasks in hi mode.
- Bridges mixed-critical scheduling and non-mixed-critical scheduling



Conclusions

- Stronger correctness criteria - guaranteed service rate to non-safety-critical tasks in hi mode.
- Bridges mixed-critical scheduling and non-mixed-critical scheduling
- The variable S_i is provided to aid the system designers in choosing the r_i value based on the tolerance on consecutive dropped jobs.

Thank You

Thank you for your time!