Combining Task-level and System-level Scheduling Modes for Mixed Criticality Systems

DS-RT 2019, Cosenza Italy, 2019-10-08

Jalil Boudjadar[1]
Saravanan Ramanathan[2] and Arvind Easwaran[2]
Ulrik Nyman[3] ulrik@cs.aau.dk

[1]Aarhus University, Denmark

[2]Nanyang Technological University, Singapore

[3]Distributed, Embedded and Intelligent Systems, DEIS
Department of Computer Science
Aalborg University, Denmark

AALBORG UNIVERSITY
DENMARK

Combining Task-level and System-level Scheduling Modes for Mixed Criticality Systems

AALBORG UNIVERSITY
DENMARK

Setting and Purpose

Mixed-critical systems

- High criticality tasks (**HC**)
- Low criticality tasks (**LC**)

Setting and Purpose

Mixed-critical systems

- High criticality tasks (**HC**)
- Low criticality tasks (**LC**)
- **HC** tasks should never miss a deadline
- Dropping as few **LC** tasks as possible

Related Work
Two approaches

System-level mode

- Two system modes
  - *Normal*
  - *Critical*

Task-level mode

Related Work
Two approaches

System-level mode

- Two system modes
  - *Normal*
  - *Critical*
- Drop all **LC** tasks
- Run **LC** tasks in degraded mode

Task-level mode

Related Work

Two approaches

System-level mode

- Two system modes
  - *Normal*
  - *Critical*
- Drop all **LC** tasks
- Run **LC** tasks in degraded mode

Task-level mode

- Each **HC** task switches individually to *Critical*

Related Work
Two approaches

System-level mode

- Two system modes
  - *Normal*
  - *Critical*
- Drop all **LC** tasks
- Run **LC** tasks in degraded mode

Task-level mode

- Each **HC** task switches individually to *Critical*
- Other **HC** tasks can miss their deadline

Assumptions

- Tasks are preemptible.
- All tasks are assigned a static criticality level (LC or HC) by design, called default criticality.
- The execution of a HC task must not be discarded under any runtime circumstances.
- The runtime criticality of a LC task can never be upgraded to HC.
- LC tasks stick always to their low confidence WCET.
- There is no dependency between LC and HC tasks.

Task

A task π_i is given by $\langle T_i, C_i^l, C_i^h, \chi_i, \rho \rangle$ where:

- T_i is the task period.
- $C_i^l \in \mathbb{R}_{\geq 0}$ and $C_i^h \in \mathbb{R}_{\geq 0}$ are the worst case execution time for low and high confidence levels respectively. We assume that $C_i^h \geq C_i^l$ for HC tasks, and $C_i^h = C_i^l$ for LC tasks.
- $\chi_i \in \{\mathbf{LC}, \mathbf{HC}\}$ is the default (constant) criticality of the task.
- ρ is the task priority.

The task runtime mode $\Omega()$ will be updated on the fly according to the actual task execution budget.

Task types

- Types of tasks

	HC	**LC**
Type	High criticality	Low criticality
Modes	**HI** and **LO**	no modes
WCET	Varies $C_i^h \geq C_i^l$	Static $C_i^h = C_i^l$

Task types

- Types of tasks

	HC	**LC**
Type	High criticality	Low criticality
Modes	**HI** and **LO**	no modes
WCET	Varies $C_i^h \geq C_i^l$	Static $C_i^h = C_i^l$

- Modes for **HC** tasks

	HI	**LO**
WCET	C_i^h	C_i^l

Three scheduling functions

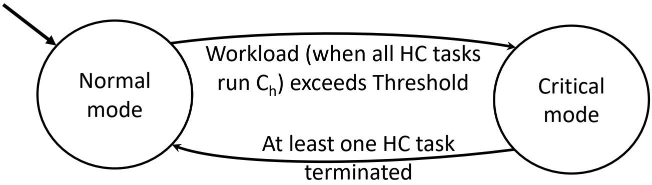
- Fixed priority scheduling $\quad Sched : 2^{\Pi} \times \mathbb{R}_{\geq 0} \rightarrow \Pi$
- Intermediate scheduling

$$Sched_I(\Pi, t) = \pi_i \mid Ready(\pi_i, t) \wedge \forall \pi_j \in \Pi \; Ready(\pi_j, t) \Rightarrow \begin{cases} \Omega(\pi_j, t) < \Omega(\pi_i, t) \\ \vee \\ \Omega(\pi_j, t) = \Omega(\pi_i, t) \wedge Sched(\{\pi_i, \pi_j\}, t) = \pi_i \end{cases}$$

- Critical scheduling

$$Sched_C(\Pi, t) = \pi_i \mid Ready(\pi_i, t) \wedge \forall \pi_j \in \Pi \; Ready(\pi_j, t) \Rightarrow \begin{cases} \chi_j < \chi_i \\ \vee \\ (\chi_j = \chi_i) \wedge \Omega(\pi_j, t) < \Omega(\pi_i, t) \\ \vee \\ (\chi_j = \chi_i) \wedge (\Omega(\pi_j, t) = \Omega(\pi_i, t)) \\ \qquad \wedge \; Sched(\{\pi_i, \pi_j\}, t) = \pi_i \end{cases}$$

System scheduling mode behavior

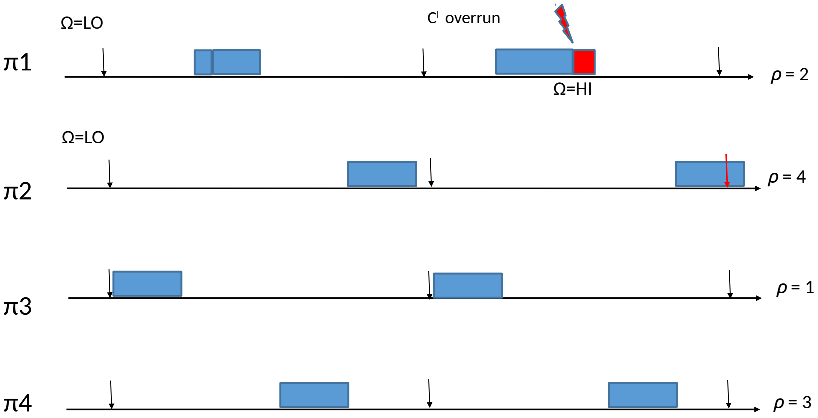


Example
Simple mixed task set

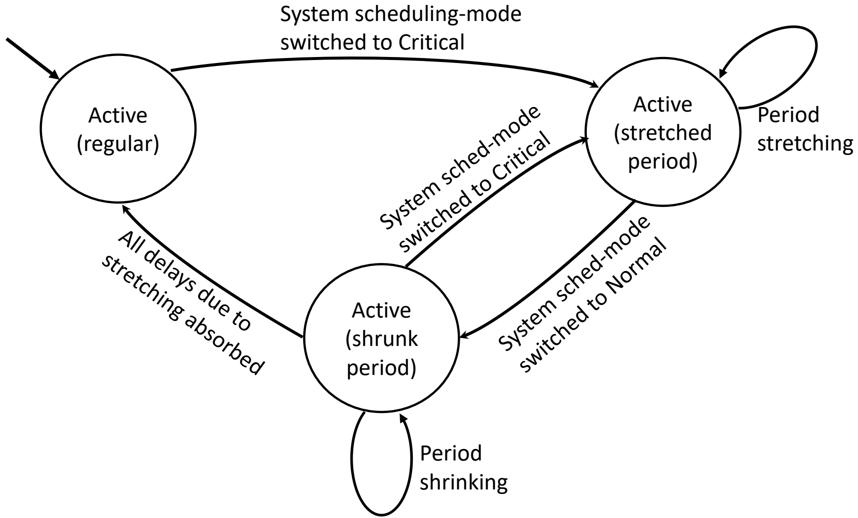
Task	T	C^l	C^h	χ	ρ
π_1	20	5	7	HC	2
π_2	20	5	6	HC	4
π_3	20	5	-	LC	1
π_4	20	4	-	LC	3

Example

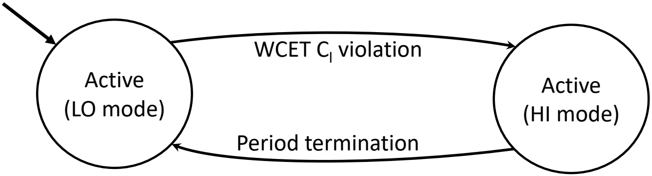
Runtime example for the system in Table



Low criticality task behavior

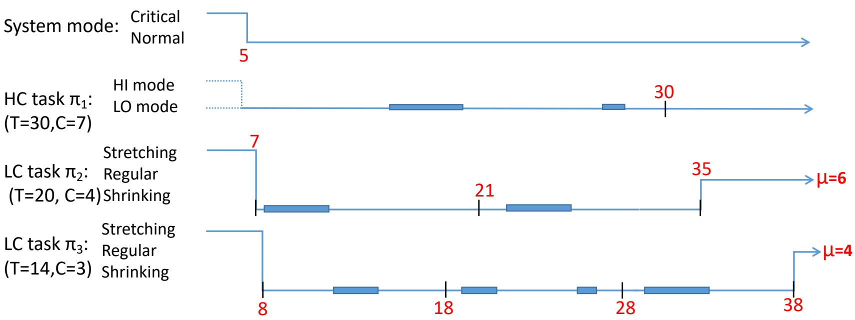


High criticality task behavior



Example of LC task periods shrinking

Shrinking with ***Δ=12*** over interval[5,30]



Algorithm

Algorithm 1: Elastic multimode scheduling

$Init()$;
while *True* **do**
  if $\exists \pi_i \mid Status(\pi_i, t) = Done \wedge t\%T_i = 0)$ **then**
    $Refresh(\pi_i)$;
  end
  if $\exists \pi_i \mid \chi_i = HC \wedge \Lambda(\pi_i, t) \geq C_i^l \wedge Status(\pi_i, t) \neq Done$ **then**
    $\Omega(\pi_i, t) = HI$;
    $Use(Sched^I())$;
  end
  if $Mode(t) = Normal \wedge \mathtt{DEM}(lp_l(t), t) < lp_l(t).T - t\%lp_l(t).T)$ **then**
    $\mathcal{T} = lp_l(t)$;
    $\mathcal{S} = t$;
    $Mode(t) = Critical$;
    $\mathcal{P} = Stretching$;
    $Use(Sched^c())$;
    foreach $\pi_j \mid \chi_j = LC$ **do**
      $T_j \mapsto T_j + (lp_l(t).T - t\%lp_l(t).T)$;
      $\delta = \delta + (\mathcal{T}.T - \mathcal{S})$;
    end
  end
  if $Mode(t) = Critical \wedge \exists \pi_i \mid \mathcal{T}(\pi_i, \mathcal{S}) \wedge t\%T_i = 0$ **then**
    $Mode(t) = Normal$;
    $\mathcal{P} = Regular$;
    $\eta = t$;
    if $\exists \pi_j \mid \Omega(\pi_j, t) = HI$ **then**
      $Use(Sched_I())$;
    end
    else
      $Use(Sched())$;
    end
  end
  if $Mode(t) = Normal \ \wedge \ \delta > 0$ **then**
    if $DEM^{\delta}(lp_l(t), t) \leq lp_l(t).T - t$ **then**
      foreach $\pi_j \mid \chi_j = LC$ **do**
        $T_j = T_j - \mu_j$;
      end
      $\mathcal{P} = Shrinking$;
      $\delta = 0$;
    end
  end
end

Case study
Origin

- Task set from

  [14] R. Dodd. Coloured petri net modelling of a generic avionics missions computer. Technical report, Department of Defence, Australia, Air Operations Division, 2006.

- WCET (C^l) given in original case.
- WCET (C^h) calculated from data fetching times.
  - $20\mu s$ for data words
  - $40\mu s$ for a command
  - $40\mu s$ for a status

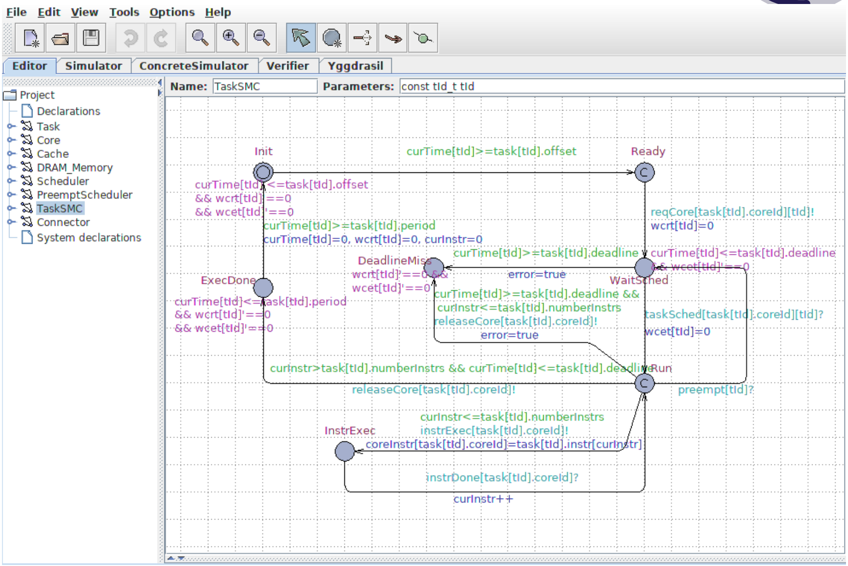
Case study

Tasks

Task	χ	T	C^l	C^h	ρ
Aircraft flight data(π_1)	HC	55	8	8.9	6
Steering(π_2)	HC	80	6	6.3	9
Target tracking(π_3)	HC	40	4	4.2	3
Target sweetening(π_4)	HC	40	2	2	4
AUTO/CCIP toggle(π_5)	HC	200	1	1	12
Weapon trajectory(π_6)	HC	100	7	7.5	10
Reinitiate trajectory(π_7)	LC	400	6.5	-	14
Weapon release(π_8)	HC	10	1	1.2	1
HUD display(π_9)	LC	52	6	-	7
MPD tactical display(π_{10})	LC	52	8	-	8
Radar tracking(π_{11})	HC	40	2	2.2	2
HOTAS bomb button (π_{12})	LC	40	1	-	5
Threat response display(π_{13})	LC	100	3	-	11
Poll RWR(π_{14})	LC	200	2	-	13
Perodic BIT(π_{15})	LC	1000	5	-	15

Case Study

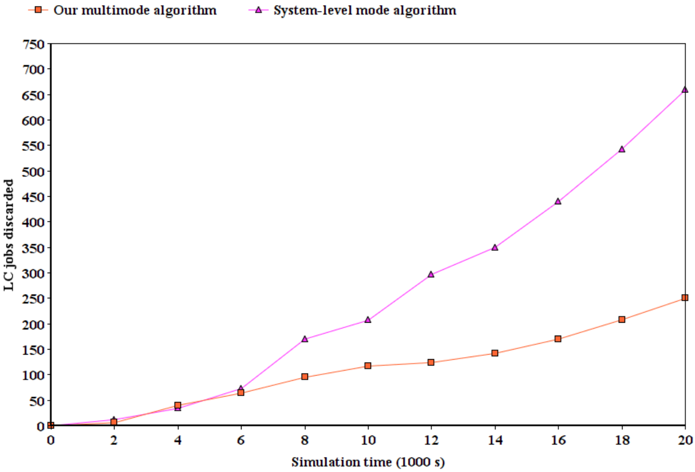
Uppaal model



Case Study
Results

- Not schedulable with classical fixed priority scheduling
  - tasks π_{10} and π_{11} miss their deadlines
- Not schedulable with task-level mode scheduling
  - task π_{10} misses its deadline (response time 106)
- System-level scheduling vs. our algorithm

Case study
Experimental results



Case study
Experimental results

- Discard rate of the LC task jobs achieved by our algorithm is 1.0% to 4.58%
- Discard rate achieved by the state of the art system-level bi-mode scheduling [13], [33] is 2.1% to 11.5%.

[13] D. de Niz, K. Lakshmanan, and R. Rajkumar. On the scheduling of mixed-criticality real-time task sets. In RTSS'09, pages 291–300, 2009.

[33] B. Madzar, J. Boudjadar, J. Dingel, T. E. Fuhrman, and S. Ramesh. Formal analysis of predictable data flow in fault-tolerant multicore systems. In FACS '16, pages 153–171, 2016

Conclusion

Conclusion

- Flexible multi-mode scheduling for mixed-criticality systems
  - accurate and non-aggressive system mode switches
- Stretching of periods
- Much less dropping of **LC** tasks
- Too computation heavy at the moment

Conclusion
And future work

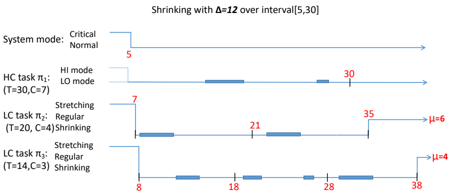
Conclusion

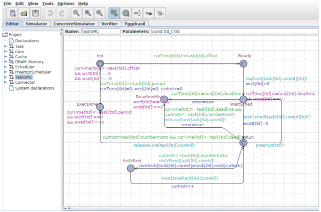
- Flexible multi-mode scheduling for mixed-criticality systems
  - accurate and non-aggressive system mode switches
- Stretching of periods
- Much less dropping of **LC** tasks
- Too computation heavy at the moment

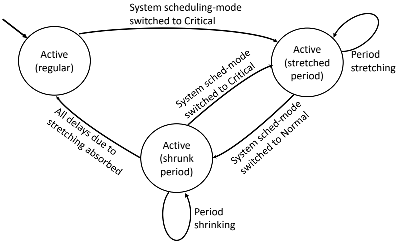
Future Work

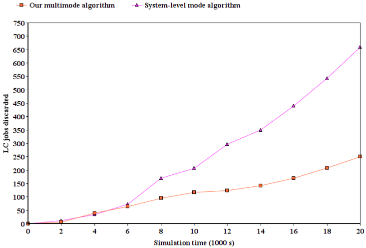
- Real implementation
- Optimization of algorithm overhead

Questions?


Shrinking with **Δ=12** over interval[5,30]
System mode:
Critical
Normal
5
HC task π_1:
(T=30,C=7)
HI mode
LO mode
30
LC task π_2:
(T=20, C=4)
Stretching
Regular
Shrinking
7
21
35
μ=6
LC task π_3:
(T=14,C=3)
Stretching
Regular
Shrinking
8
18
28
38
μ=4


File Edit View Tools Options Help
Editor Simulator ConcreteSimulator Verifier Yggdrasil
Project
Declarations
Task
Core
Cache
DRAM_Memory
Scheduler
PreemptScheduler
TaskSMC
Connector
System declarations
Name: TaskSMC
Parameters: const tId_t tId
Init
Ready
ExecDone
DeadlineMiss
WaitSched
Run
InstrExec


System scheduling-mode
switched to Critical
Active
(regular)
Active
(stretched
period)
Period
stretching
System sched-mode
switched to Critical
All delays due to
stretching absorbed
System sched-mode
switched to Normal
Active
(shrunk
period)
Period
shrinking


Our multimode algorithm
System-level mode algorithm
LC jobs discarded
Simulation time (1000 s)
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
0
2
4
6
8
10
12
14
16
18
20