University of Hertfordshire UH

LBP-CA: A Short-term Scheduler with Criticality Arithmetic

Sajid Fadlelseed[1*], Raimund Kirner[2], and Catherine Menon[3]

[1]*Department of Computer Science, University of Hertfordshire, Hatfield, UK*

*Corresponding author: {q.fadlelseed, r.kirner, c.menon}@herts.ac.uk

In safety-critical systems a smooth degradation of services is a way to deal with resource shortages. Criticality arithmetic is a technique to implement services of higher criticality by several tasks of lower criticality. In this paper, we present LBP-CA, a mixed-criticality scheduling protocol with smooth degradation based on criticality arithmetic. In the experiments we show that LPB-CA can schedule more tasks than related scheduling protocols (BP and LBP) in case of resource shortage, minimising the negative effect on low-criticality services. This is achieved by considering information about criticality arithmetic of services.

Keywords: real-time systems; safety integrity level; scheduling; mixed-criticality

Introduction

Criticality Arithmetic (CA) or SIL-arithmetic as termed in [1], is a Mixed-criticality (MC) model that assembles a number of replicated tasks with low criticality levels, to implement a service of higher criticality level. The Adaptive Tolerance-based Mixed-criticality Protocol - Criticality Arithmetic (ATMP-CA) [4] is CA-aware mid-term scheduler that optimises the utility of individual tasks when permanent fault occurs e.g., core-failure, to maximise the overall system utility, here we present a novel CA-aware short-term scheduler (LBP-CA) which assures return to Normal-mode from Critical-mode, much earlier than reference schedulers that do not take the use of criticality arithmetic into account. MC systems enter the Critical-mode whenever a transient fault e.g., task overrun occurs, which results in abandoning release of Low-criticality tasks to avoid their interference on High-criticality tasks during the Critical-mode [5].

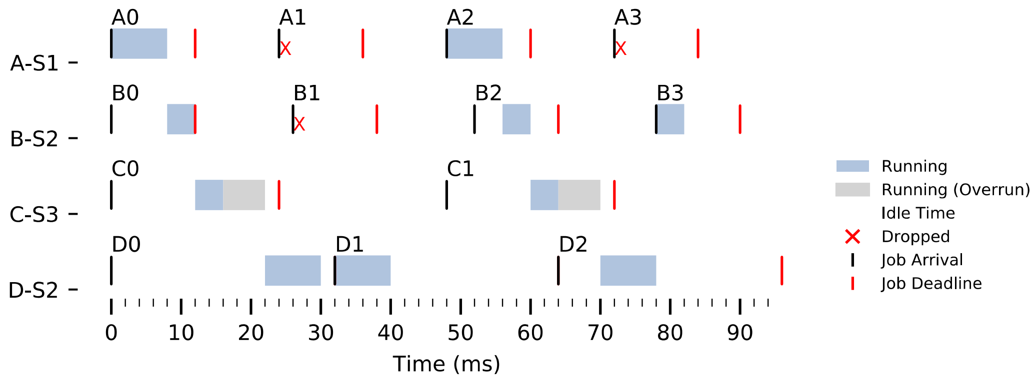
Reference schedulers are Bailout Protocol (BP) [3] and Lazy Bailout Protocol (LBP) [2]. BP and LBP define three criticality modes to schedule the execution of tasks with different criticality levels: Normal-mode, Bailout-mode and Recovery-mode. Bailout-mode represents the Critical-mode explained above, and Recovery-mode is used to ensure that the last High-criticality task with Low-priority is executed before returning to Normal-mode. LBP differs from BP in that instead of dropping Low-criticality tasks during Bailout and Recovery modes, they are added to a Low-priority queue for possible execution when the system returns to Normal-mode. Though LBP may drop less tasks than classic BP, it doesn't improve the BP functionality that operates the process of returning to Normal mode.

System Model: We assume a single processor mixed-criticality system, which consists of multiple services that could have different levels of criticality. A service can be implemented by one task or multiple tasks using criticality arithmetic [1]. Each service is identified by the tuple: $\boldsymbol{s} = \langle \boldsymbol{id}, \boldsymbol{l}, \boldsymbol{T} \rangle$, where $\boldsymbol{id}$ is the service identifier, $\boldsymbol{l}$ is the service criticality and $\boldsymbol{T}$ is the set of tasks implementing the service. Each individual task () is defined by the tuple $\boldsymbol{\tau} = \langle \boldsymbol{id}, \boldsymbol{p}, \boldsymbol{d}, \boldsymbol{c1}, \boldsymbol{c2}, \boldsymbol{L}, \boldsymbol{s} \rangle$, where $\boldsymbol{id}$ is the task identifier, $\boldsymbol{p}$ is the task period, $\boldsymbol{d}$ the task deadline, $\boldsymbol{c1}$ *optimistic* worst-case execution time estimate (WCET1), $\boldsymbol{c2}$ *pessimistic* worst-case execution time estimate (WCET2), task criticality is defined by $\boldsymbol{L}$ and $\boldsymbol{s}$ is the service that is implemented by the task.

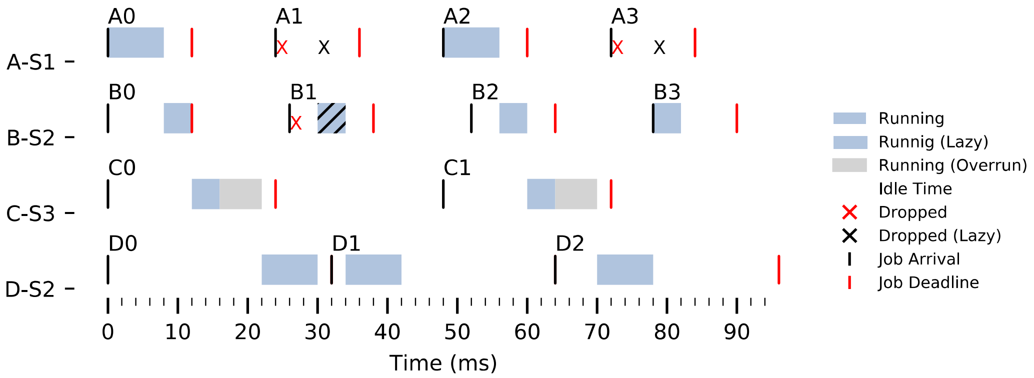
Experimental Setup

We have implemented a short-term scheduling simulator which is configured to simulate mixed-criticality services on a single processor system. The simulator has also implemented the underlying scheduling algorithm (deadline-monotonic) and the references mixed-criticality protocols (BP and LBP) and the novel mixed-criticality protocol (LBP-CA).

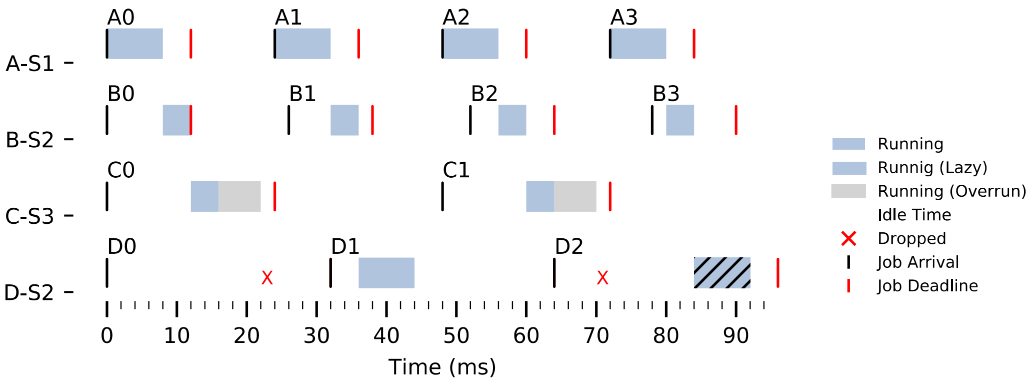
We have generated a task-set with random parameters for task periods and worst-case execution time and mix of implicit and constrained deadlines. The criticality of a task or service is either High or Low, which corresponds to the criticality level of the task. We have constrained the task generation such that it includes a single High criticality CA-aware service (S2). The complete structure of tasks and services is implemented in the following task-set in Table 1.



(a) BP protocol (3 *dropped jobs, 10 completed jobs)*



(a) LBP protocol (2 *dropped jobs, 11 completed jobs)*



(a) LBP-CA protocol (1 *dropped job, 12 completed jobs)*

Figure 1. Comparison of scheduling mixed-criticality tasks between BP, LBP and LBP-CA

t.id	t.p	t.d	t.c1	t.c2	t.L	s.id	s.l
A	24	12	8	8	Low	S1	Low
B	26	12	4	4	Low	S2	High
C	48	24	4	10	High	S3	High
D	32	32	8	8	High	S2	High

Table 1. Set of Services and Tasks (only S2 use Criticality Arithmetic)

Results and discussion (Section heading, Calibri 12 pt)

The purpose of our experiment was to show that the LBP-CA returns to Normal-mode with the least number of abandoned Low-critical tasks compared to reference protocols. Figure 3 shows the schedule for the task-set presented in Table 1. In Figures 3 (a), (b) and (c), we can observe that job $\boldsymbol{C0}$ overruns its $\boldsymbol{c1}$ estimates, which results in entering Critical-mode. As per Figures 3 (a), (b) and (c), in BP and LBP protocols we can observe that the overrun caused the system to enter the Bailout-mode and abandon the Low-Criticality jobs ($\boldsymbol{A1}$ and $\boldsymbol{B1}$), to avoid possible interference with High-Criticality jobs. However, the LBP protocol (Figure 3 (b)) shows the successful allocation for the job $\boldsymbol{B1}$ using its lazy execution mechanism. In contrast, our LBP-CA protocol (as shown in Figure 3 (c)) scheduled all jobs successfully except job $\boldsymbol{D1}$. This is because the first instance of its replica B has been executed successfully. Hence entering Recovery-Mode has been mitigated. Overall, the collected simulation results indicate that the LBP-CA drops a smaller number of Low-Criticality jobs and efficient management for the system run-time modes in comparison to reference schedulers (BP and LBP protocols).

Conclusion

Integrating CA to Mixed-criticality schedulers as in (LBP-CA), allows efficient mode-change management between Normal-mode and Critical-mode/s due to transient faults. LBP-CA can access information about criticality arithmetic (CA) via task redundancy compared to referenced scheduling protocols (BP and LBP) which are CA-agnostic. Our simulation data shows that even at/after resource shortage, LBP-CA returned to the normal state prior to BP and LBP, providing a smoother service degradation.

References

[1] Menon, C., Iacovelli, S. and Kirner, R., 2020, May. ODRE Workshop: Using SIL Arithmetic to Design Safe and Secure Systems. In 2020 IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC) (pp. 213-218). IEEE
[2] Iacovelli, S. and Kirner, R., 2019. A lazy bailout approach for dual-criticality systems on uniprocessor platforms. Designs, 3(1), p.10.
[3] Bate, I., Burns, A. and Davis, R.I., 2015, July. A bailout protocol for mixed criticality systems. In 2015 27th Euromicro Conference on Real-Time Systems (pp. 259-268). IEEE.
[4] Fadlelseed, S., Kirner, R. and Menon, C., 2021. ATMP-CA: Optimising Mixed-Criticality Systems Considering Criticality Arithmetic. Electronics, 10(11), p.1352.
[5] Vestal, S., 2007, December. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In 28th IEEE international real-time systems symposium (RTSS 2007) (pp. 239-243). IEEE.