

Corrections to and Discussion of “Implementation and Evaluation of Mixed-criticality Scheduling Approaches for Sporadic Tasks”

TOM FLEMING, University of York

HUANG-MING HUANG, Washington University in St. Louis

ALAN BURNS, University of York

CHRIS GILL, Washington University in St. Louis

SANJOY BARUAH, University of North Carolina - Chapel Hill

CHENYANG LU, Washington University in St. Louis

The AMC-IA mixed-criticality scheduling analysis was proposed as an improvement to the AMC-MAX adaptive mixed-criticality scheduling analysis. However, we have identified several necessary corrections to the AMC-IA analysis. In this article, we motivate and describe those corrections, and discuss and illustrate why the corrected AMC-IA analysis cannot be shown to outperform AMC-MAX.

CCS Concepts: • **Computer systems organization** → **Real-time systems**

Additional Key Words and Phrases: Adaptive mixed-criticality scheduling, real-time systems

ACM Reference Format:

Tom Fleming, Huang-Ming Huang, Alan Burns, Chris Gill, Sanjoy Baruah, and Chenyang Lu. 2017. Corrections to and discussion of “implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks”. *ACM Trans. Embed. Comput. Syst.* 16, 3, Article 77 (May 2017), 4 pages.

DOI: <http://dx.doi.org/10.1145/2974020>

1. INTRODUCTION

The AMC-IA mixed-criticality scheduling analysis [Huang et al. 2014] was proposed as an improvement to the AMC-MAX adaptive mixed-criticality scheduling analysis [Baruah et al. 2011]. AMC-IA uses two definitions of a function $n(s)$ to represent the number of releases of a task by time s which is defined as *the last deadline before a criticality change*. For low-criticality tasks, n is defined by:

$$n_j(s) = \left\lceil \frac{s}{T_j} \right\rceil. \quad (1)$$

Work described in this note was supported in part by an EPSRC(UK) MCC grant; by US National Science Foundation grants CNS-0834755, CNS-0834270, CNS-0834132, CNS-1016954, CCF-1136073, and CNS-1329861; by US ARO grant W911NF-09-1-0535; by US AFOSR FA9550-09-1-0549; and by AFRL grant FA8750-11-1-0033.

Authors' addresses: T. Fleming and A. Burns, Computer Science Department, University of York; emails: {tdf506, alan.burns}@york.ac.uk; H.-M. Huang, C. Gill, and C. Lu, Department of Computer Science and Engineering, Washington University in St. Louis; emails: huangming.huang@gmail.com, {cdgill, lu}@cse.wustl.edu; S. Baruah, Computer Science Department, University of North Carolina - Chapel Hill; email: baruah@cs.unc.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2017 ACM 1539-9087/2017/05-ART77 \$15.00

DOI: <http://dx.doi.org/10.1145/2974020>

For high-criticality tasks, a different definition for n is given:

$$n_k(s) = \max\left(\left\lfloor \frac{s - D_k}{T_k} \right\rfloor + 1, 0\right). \quad (2)$$

Note that subscript j is used in the first definition and k in the second.

The response-time of a two-criticality system, during the change from low-criticality mode LO to high-criticality mode HI (0 to 1), is given by the following equation (from Huang et al. [2014])¹:

$$R_i^s = C_i(0) + \sum_{\tau_j \in H_i} n_j(s)C_j(0) + \sum_{\tau_k \in HHC_i} \left(\left\lfloor \frac{R_i^s}{T_k} \right\rfloor - n_k(s)\right) C_k(1), \quad (3)$$

where H_i is the set of task with higher priority than τ_i (of either criticality) and HHC_i is the set of task with higher priority and higher (or equal) criticality than τ_i .

2. CORRECTIONS TO AMC-IA

The first correction to the AMC-IA analysis is that the initial computation time (for τ_i) itself should be $C_i(1)$ as it executes for its maximum value. The second correction to AMC-IA, to remove confusion as to which of the n functions should be used, is to rewrite the preceding equations in an equivalent but more obvious form. We still define n by Equation (1) and introduce a new function m to encode Equation (2):

$$m_k(s) = \max\left(\left\lfloor \frac{s - D_k}{T_k} \right\rfloor + 1, 0\right). \quad (4)$$

We then rewrite Equation (3):

$$\begin{aligned} R_i^s = & C_i(1) + \sum_{\tau_j \in HLC_i} n_j(s)C_j(0) + \sum_{\tau_k \in HHC_i} m_k(s)C_k(0) \\ & + \sum_{\tau_k \in HHC_i} \left(\left\lfloor \frac{R_i^s}{T_k} \right\rfloor - m_k(s)\right) C_k(1), \end{aligned} \quad (5)$$

where HLC_i is the set of low-criticality tasks with higher priority than τ_i , and HHC_i is the set of high-criticality tasks with higher priority than τ_i .

Following this notational clarification, we then modify the definition of AMC-IA itself. As the third correction, the “+1” is removed from Equation (4):

$$m_k(s) = \max\left(\left\lfloor \frac{s - D_k}{T_k} \right\rfloor, 0\right). \quad (6)$$

The fourth correction to the AMC-IA analysis is that each “s” point is now defined to be the *first deadline after a criticality mode change*.

3. DISCUSSION

The two significant corrections described earlier are to the definitions of “s” and the equation for $m_k(s)$. The change to “s” is necessary because the initial definition could underestimate the impact of high-priority low-criticality tasks on other tasks. This is easy to see with a simple system that has one high-priority low-criticality task τ_1 with $D_1 = T_1 = 10$, and a set of high-criticality tasks with $D > 12$. If all tasks are released at time 0, and a mode change occurs at time 12 (in some high-criticality task), then the

¹In Huang et al. [2014], LO and HI criticalities are denoted as 0 and 1, respectively; thus, for example, the LO-criticality WCET of τ_i is denoted $C_i(0)$ (rather than $C_i(LO)$ – the notation used in Baruah et al. [2011]).

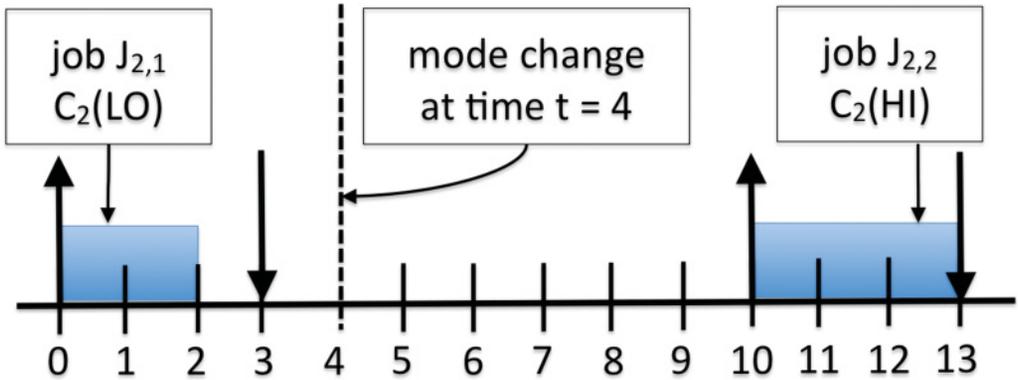


Fig. 1. High-criticality task example for discussion.

“last deadline before 12” is at time 10; Equation (1) gives $n(10) = 1$, but it should be 2, as releases at times 0 and 10 will impact tasks at time 12. To avoid underestimating the interference given by Equation (1), it is necessary to define “ s ” as the *first deadline after a criticality mode change*. Unfortunately, this can now underestimate the interference of high-criticality tasks, as illustrated in Figure 1.

In this example, consider a high-criticality task τ_2 with $C_2(LO) = 2$, $C_2(HI) = 3$, $D_2 = 3$ and $T_2 = 10$. If a criticality mode change occurred at time 4 (with all tasks released at time 0, and all other tasks having deadline later than time 13), then the new definition of “ s ” would give a point at time 13 (as 13 is the next deadline after time 4). At this point, Equation (4) gives a value of 2. This implies that in any interval after 13 there will be two executions of the task with $C(LO)$. It is easy to see that this is wrong, as at time 13 there may be only one execution with $C(LO)$ and hence one with $C(HI)$. As $C(HI) \geq C(LO)$ this could lead to an underestimation of this task’s interference. To correct for this error, the “+1” in Equation (4) is removed to give Equation (6). Now $m(13) = 1$, which is a sufficient correction.

If these two corrections are applied, then it is not clear how Equation (5) can lead to tighter analysis than AMC-MAX. With the removal of “+1,” a task with $D = T$ will assume no $C(LO)$ interference unless “ s ” is $2T$ or greater. AMC-MAX will assume (for $s < 2T$) either 0 or 1 $C(LO)$ hits - depending on the value of R . Hence, they will often give the same result, but AMC-MAX can in some situations deliver (correctly) less interference.

It remains an open question whether there is a definition of AMC-IA that lies between the incorrect published one and the one given earlier that is both sufficient and “better” than AMC-MAX, where “better” means that it will deem more task sets to be schedulable. Certainly AMC-MAX is not exact, so such a scheme is possible. However, with our current investigation, it appears so far that in order for AMC-IA to be made sufficient it may not be able to outperform AMC-MAX.

4. CONCLUSIONS

In this article, we have described two necessary corrections to the AMC-IA analysis and have shown that with those corrections AMC-IA cannot be shown to outperform AMC-MAX. It would be helpful to examine further whether the approach taken by AMC-IA offers potential insights into how AMC-MAX could be improved, though as we have demonstrated in this article, such improvement remains future work.

REFERENCES

- Sanjoy K. Baruah, Alan Burns, and Robert I. Davis. 2011. Response-time analysis for mixed criticality systems. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*. 34–43.
- Huang-Ming Huang, Christopher Gill, and Chenyang Lu. 2014. Implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks. *ACM Trans. Embed. Comput. Syst.* 13, 4s, Article 126 (April 2014), 25 pages. DOI : <http://dx.doi.org/10.1145/2584612>

Received July 2016; accepted July 2016