Mixed-Criticality Scheduling of Messages in Time-Triggered Protocols

Zdeněk Hanzálek
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
and Porsche Engineering Services, s.r.o.
hanzalek@fel.cvut.cz

Tomáš Tunys
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
tunystom@fel.cvut.cz

Abstract

In different communication protocols, a message retransmission is often used to increase reliability. Messages with higher criticality are allowed for more retransmissions and therefore different execution time values should be specified for each criticality level of the given message. The paper deals with a problem of the static scheduling of mixed-criticality messages on a communication resource (collision domain) without preemption. On the basis of the current state of the art in the mixed-criticality scheduling, our problem, denoted as $1|r_i, d_i, mc = \mathcal{L}|\beta C_{max} + \alpha \sum T$*, is formulated. We use two metaheuristic methods used to solve the mixed-criticality scheduling problem.*

1 Introduction

The problem we address in this paper is motivated by increasing trend to study and research *mixed-criticality* scheduling. It follows right from the attempts of embedded systems to provide multiple functionalities upon a single shared platform. Therefore, the systems become mixtures of critical functionalities that need to pass a certification process; and non-critical functionalities, which do not.

We can look at the making of an embedded system as a struggle between its designers that wish to make the system as efficient as possible in order to provide maximum performance, and the certification authority requiring guarantees of the worst-case behavior. The research of the mixed-criticality scheduling is therefore aimed towards the construction of scheduling policies such that on the one hand they will help these mixed-criticality systems to facilitate the certification process (see [3]), but on the other hand they will also make good use of the shared resources.
[1]

1.1. Mixed-criticality and communication protocols.

We demonstrate our motivation to study the non-preemptive mixed-criticality scheduling on a communication protocol that will use the concept of criticality to improve its reliability and to make better use of the bandwidth. We may look at a communication bus as a resource that is being shared by plenty of different messages – jobs. It is not so hard to imagine a situation in which some of the messages need to carry more important information than the others. These important messages would correspond to high-critical jobs and, therefore, they may be transmitted at the expense of other less important messages that, on the contrary, would correspond to low-critical jobs. With this we acquire a mixed-criticality system represented by the messages and their criticality (importance). The non-preemptiveness arises from the nature of the protocol – the messages on a MAC layer must not be chunked, i.e. the corresponding jobs cannot be preempted.

Based on the observation that "the more confidence one needs in a task execution time bound, the larger and more conservative that bound tends to be in practice" Vestal [7] proposed that different execution time values should be specified for each criticality level of a given job. Our idea is to use the concept of criticalities, when the message retransmissions are allowed up to the criticality of the message and they lead to multiple execution times.

Consider an important message is not delivered, then the following questions arise: How should the dispatching system respond to this particular situation? Is it possible to resend the message and to deliver all messages on time? These questions relate to a problem of MC scheduling. We study this problem in the context of the time-triggered protocols where the static part/segment of the period has a fixed schedule handling periodic messages with zero jitter and the dynamic part/segment is used for aperiodic messages. We are trying to tackle the optimization version of the mixed-criticality scheduling problem, while minimizing the length of the static part in order to leave room for the dynamic part.

[1] **Acknowledgments**: This work relates to Department of the Navy Grant N62909-12-1-7009 issued by Office Naval Research Global. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

1.2. Related work and paper outline

In this paper we work with the model of mixed-criticality (MC) systems formulated by Baruah, Li, and Stougie in [1]. Baruah, in [4], provides a proof of intractability of a problem to decide whether the given MC instance is schedulable or not (*MC Schedulability problem*). In [2], the former results are elaborated further and a new scheduling algorithm is proposed - OCBP (*Own Criticality-Based Priority*) - that is capable of scheduling a subset of MC instances according to recursively created priority list in polynomial time. The OCBP algorithm also creates a sufficient MC-schedulability test, i.e. an MC instance that is so-called OCBP-schedulable is also MC-schedulable. It is worth pointing out that according to the same source, the first person who identified and fomalized the scheduling problem that arises from multiple certification requirements was Vestal in [7].

In [1], the need for a new scheduling theory by demonstrating the failure of the existing approaches (*reservations-based* and *priority-based*) in solving the MC schedulability problem is also justified.

In Section 2, we present the problem statement and the formal model for the representation of mixed-criticality systems based on Baruah, Li, and Stougie in [1]. In Section 3, we refer to two metaheuristic algorithms [2] for solving the proposed problem.

2 Problem Statement

Before we embark onto the problem statement we restate here the formal model of the mixed-criticality systems, which appeared in [1], and has been further elaborated on in [2]. We follow the terminology and the formal model described in the latter source and we adopt it as follows: A mixed-criticality (MC) system with $\mathcal{L} \in \mathbb{N}$ criticality levels is defined as a finite collection of MC jobs. A mixed-criticality job is characterized by a 4-tuple of parameters: $J_i = (r_i, d_i, \chi_i, P_i)$, where $r_i \in \mathbb{R}^+$ denotes the release time (often called *offset* in the real-time scheduling community), $d_i \in \mathbb{R}^+$ denotes the due date, $\chi_i \in \{1, 2, \ldots, \mathcal{L}\}$ denotes the criticality of the job (the larger the value, the higher the criticality), and $P_i : \{1, 2, \ldots, \mathcal{L}\} \rightarrow \mathbb{R}^+$ specifies the Worst Case Execution Time (WCET) estimate of the job for each level of criticality. We assume that P_i is a non-decreasing partial function, i.e.

$$\forall \ell_1, \ell_2 \in \{1, 2, \ldots, \chi_i\}, \ell_1 \leq \ell_2 : P_i(\ell_1) \leq P_i(\ell_2)$$

MC instance. An MC instance is specified as a finite set of MC jobs: $I_{MC} = \{J_1, \ldots, J_n\}$.

Behavior. The semantics of the specified model is as follows: each job J_i of an MC instance I_{MC} needs to start its execution after a time instant r_i and should be completed till d_i. Within this time interval, the job is continuously (without interruption) executed for some amount of time γ_i, which is not known *a priori*. The time γ_i becomes revealed only by actually executing the job until it *signals* that it has been completed. In different execution runs each of the γ_i may take different value, so the collection of specific γ_i values of a single run is referred to the *behavior* of the I_{MC}.

Scheduling strategy. A *scheduling strategy* for instance I_{MC} specifies, for all possible behaviors of I_{MC}, which job (if any) to execute at each instant in time. A *scheduling strategy* does not have a priori knowledge of the behavior of I_{MC}, hence at each instant, scheduling decisions are made based only on the γ_i values of the already scheduled jobs.

Correctness. A scheduling strategy is *correct* if it satisfies the following criterion for each $\ell \geq 1$: when scheduling any behavior of criticality level ℓ, it ensures that every job J_i with $\chi_i \geq \ell$ receives sufficient execution time to signal its completion.

2.1 Optimization problem

Given an MC instance I_{MC} comprised of n jobs with maximum criticality level $\mathcal{L}$ (i.e., $\chi_i \leq \mathcal{L} \forall i \in \{1, \ldots, n\}$), we are concerned with a problem of designing a correct non-preemptive scheduling strategy that minimizes the makespan of the scheduled instance. We extend the Graham's notation as $1|r_i, d_i, mc = \mathcal{L}|\beta C_{max} + \alpha \sum T$, where 1 denotes a single resource, r_i denotes the release date (often called an offset), d_i stands for the due-date, the middle parameter mc denotes the maximum criticality level of the scheduled instance, and the objective is a minimization of the maximum completion time C_{max}.

Example. Consider the following MC instance I_{MC} comprised of 4 messages corresponding to 4 jobs, i.e. $I = \{J_1, J_2, J_3, J_4\}$, where J_2 and J_4 have a criticality of 2 and the others have a criticality of 1. This setting creates a mixed-criticality system with 2 criticality levels, where the execution times of a job J_i are denoted by 2-tuple $[P_i(1), P_i(2)]$, where $P_i(1)$ corresponds to the transmission time of the message on the resource and $P_i(2)$ corresponds to the transmission and retransmission time. The specification of the system is as follows:

- $J_1 = (0, 4, 1, [2, 2])$
- $J_2 = (0, 6, 2, [2, 5])$
- $J_3 = (5, 8, 1, [1, 1])$
- $J_4 = (4, 9, 2, [1, 3])$

The behavior of this instance has a criticality of 1 only if the execution times of J_2 and J_4, i.e. γ_2 and γ_4, do not exceed 2 or 1 units, respectively, otherwise it has a criticality of 2. On the other hand, any behavior for which the execution time γ_2 or γ_4 exceeds 5 or 3 units, or any behavior of criticality 1 for which the execution times γ_1

[2] http://support.dce.felk.cvut.cz/pub/hanzalek/MCScheduling/index.htm

and γ_3 exceeds 2 or 1 units, respectively, is by definition errorneous.

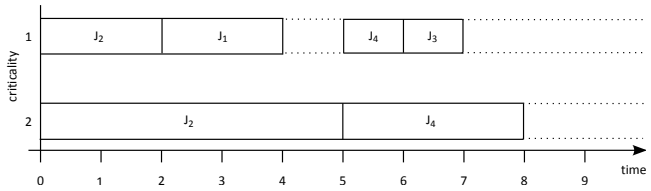

Figure 1. An optimal schedule for the I_{MC} instance. The certificate of optimality is the length of the schedule at criticality level 2 that is 8 units long and it cannot be lower since the criticality–2 jobs are executed without idle times when they both exceed their criticality 1 WCETs.

Figure 1 shows a solution of the $1|r_i, d_i, mc-2|\beta C_{max} + \alpha \sum T$ problem given the described MC instance I_{MC} in form of a schedule that determines for each behavior which job is being executed at any time.

Non-preemptiveness. Due to the non-preemptive nature of our problem it does not necessary mean that when the behavior of an MC instance is revealed to be of criticality $\ell > 1$ then all jobs J_i with $\chi_i < \ell$ do not get executed at all. In fact once the job that raises the criticality level of the behavior finishes its execution some of the jobs with lower criticality actually get executed. Consider the behavior $(2, 4, 1, 1)$ of the example MC instance scheduled as depicted in Figure 1, clearly the job J_3 will be executed despite its criticality ($\chi_3 = 1$) is less than the criticality level of behavior ($\ell = 2$).

3 Metaheuristic Scheduling Algorithms

This section contains a brief description of 2 metaheuristic algorithms to solve the $1|r_i, d_i, mc = \mathcal{L}|\beta C_{max} + \alpha \sum T$ problem. The two described algorithms have been implemented in native C# code without use of any supporting libraries.

3.1 Multicriteria Evaluation Function

In both of these algorithms a multi-criteria evaluation function is used to decide between two (or more) *valid* solutions, which one of them is better.

Valid solutions. A solution of an MC instance $I_{MC} = \{J_1, J_2, \ldots, J_n\}$ is a mapping $S : \{1, 2, \ldots, n\} \to \mathbb{R}_0^+$ which assigns to each job J_i a starting time $S(i)$. We call a solution *valid*, if it obeys the release date constraint and it obeys the correctness constraint (defined in Section 2). The overall tardiness, dependent on the due dates, is incorporated in the criterion function.

Encoding. As has been noted a *valid* solution of the $1|r_i, d_i, mc = \mathcal{L}|\beta C_{max} + \alpha \sum T$ optimization problem for an MC instance having n jobs is represented by the mapping S. Consider a permutation π on $\{1, 2, \ldots, n\}$ for which applies $\forall i, j \in \{1, 2, \ldots, n\} : i \leq j \Rightarrow S(\pi(i)) \leq S(\pi(j))$, i.e. π represents the ordering of the jobs in a non-decreasing order according to their start times. This permutation π *encodes* the associated solution S. To avoid misunderstanding, instead of $S(\pi(i))$ we use $S_{\pi(i)}$ since this way is referred to the start time as well as an associated solution.

Evaluation. The multi-criteria evaluation function of an encoded solution π has the following form

$$E(\pi) = \alpha \sum_{i=1}^{n} \max\{0, S_{\pi(i)} + P_{\pi(i)}\left(\chi_{\pi(i)}\right) - d_{\pi(i)}\} + \beta C_{max} \quad (1)$$

where the start times S are defined as

$$S_{\pi(1)} = r_{\pi(1)} \quad (2)$$

$$S_{\pi(i)} = \max_{j<i}\{S_{\pi(j)} + P_{\pi(j)}(\min\{\chi_{\pi(i)}, \chi_{\pi(j)}\}), r_{\pi(i)}\} \quad i = 2, \ldots, n \quad (3)$$

and the length of the schedule C_{max} is

$$C_{max} = \max_{i=1,\ldots,n}\{S_{\pi(i)} + P_{\pi(i)}(\chi_{\pi(i)})\} \quad (4)$$

3.2 Simulated Annealing Algorithm

The simulated annealing is a broadly used optimization method for solving different kinds of problems. We have implemented it to solve the specified problem; the pseudo-code is depicted on [3] partly insipred by [6].

3.3 Genetic algorithm

We have also used a genetic algorithm with elitism to solve the problem, whose pseudo-code is on the same web. We have implemented for it various selection methods, crossover, and mutation (permutation-based) operators, but the following list of selection methods and operators contains only those which proved themselves better than the others, and those are: *Roulette Wheel Selection*, *Randomized Tournament Selection*, *Order Crossover Operator*, *Position-Based Crossover Operator*, *Insertion Mutation Operator*, and *Exchange Mutation Operator*.

4 Experimental Results

For experimental purposes, we used our own test instances that were generated by methods described on the web.

[3] http://support.dce.felk.cvut.cz/pub/hanzalek/MCScheduling/index.htm

The performance and the results of the algorithms are presented in the tables at the end of this section.

4.1 Test Results

The following 5 tables show the results of the algorithms for the described test sets.

Table 1. Results for test set T1. The numbers in the parentheses is the total sum of the jobs' tardiness. All genetic algorithms ran for 5 minutes.

	SAA		GA1	GA2
n	C_{max}	CPU [s]	C_{max}	C_{max}
50	925 (18)	0.81	802 (32)	801 (20)
100	1657 (201)	3.28	1269 (25)	1232 (29)
200	2737 (667)	13.55	2608 (125)	2630 (100)
300	5107 (224)	32.30	4575 (334)	4552 (253)

Table 2. Results for test set T2.

	SAA		GA1	GA2
n	C_{max}	CPU [s]	C_{max}	C_{max}
50	2647 (0)	0.74	2647 (0)	2652 (0)
100	5882 (0)	3.23	5882 (0)	5882 (0)
200	10589 (0)	13.46	10589 (0)	10589 (0)
300	17006 (0)	32.42	17006 (21)	17006 (0)
400	23606 (0)	58.46	23661 (354)	23606 (0)
500	28184 (0)	112.01	28184 (340)	28184 (424)

Table 3. Results for test set T3.

	SAA		GA1	GA2
n	C_{max}	CPU [s]	C_{max}	C_{max}
50	2431 (0)	0.73	2431 (4)	2431 (0)
100	4882 (0)	3.15	4882 (0)	4882 (0)
200	10696 (0)	13.76	10729 (0)	10705 (73)
300	16466 (0)	31.51	16446 (185)	16466 (98)
400	22464 (0)	55.08	22464 (19)	22464 (0)
500	27660 (0)	88.47	27660 (0)	27660 (0)

Table 4. Results for test set T4.

	SAA		GA1	GA2
n	C_{max}	CPU [s]	C_{max}	C_{max}
100	6450 (0)	3.35	6450 (0)	6450 (0)
200	13024 (0)	13.81	13024 (0)	13024 (17)
300	20544 (0)	33.80	20544 (0)	20544 (207)
400	26434 (0)	59.34	26434 (606)	26434 (423)
500	27881 (0)	93.90	27881 (545)	27881 (263)

Table 5. Results for test set T5.

	SAA		GA1	GA2
n	C_{max}	CPU [s]	C_{max}	C_{max}
100	5585 (0)	3.29	5585 (0)	5585 (0)
200	11617 (0)	13.52	11617 (0)	11617 (56)
300	16652 (0)	33.26	16652 (204)	16652 (168)
400	25357 (0)	60.70	25357 (0)	25357 (119)
500	28140 (0)	94.04	28140 (0)	28140 (455)

5 Summary

In this paper we have formalized and examined a non-preemptive mixed-criticality scheduling problem referred to as $1|r_i, d_i, mc = \mathcal{L}|\beta C_{max} + \alpha \sum T$. Moreover, we have brought a new motivation, in the context of communication protocols, to study the non-preemptive scheduling. Two metaheuristic algorithms have been implemented and experimentally evaluated on the test sets as a way to tackle this problem

References

[1] Sanjoy Baruah, Haohan Li, and Leen Stougie. Towards the design of certifiable mixed-criticality systems. In *Proceedings of the Real-Time Technology and Applications Symposium (RTAS)*, Stockholm, Sweden, April 2010. IEEE Computer Society Press.

[2] Sanjoy Baruah, Haohan Li, and Leen Stougie. Mixed-criticality scheduling: improved resource-augmentation results. In *Proceedings of the ICSA International Conference on Computers and their Applications (CATA)*, Stockholm, Sweden, April 2010. IEEE Computer Society Press.

[3] S. K. Baruah and G. Fohler. Certification-cognizant time-triggered scheduling of mixed-criticality systems, In *Proceedings of IEEE Real-time Systems Symposium 2011*, December 2011.

[4] Sanjoy Baruah. Mixed-criticality scheduability analysis is highly intractable. Available at `http://www.cs.unc.edu/~baruah/Pubs.shtml`, 2009.

[5] S.Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N.Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. *Mathematical Foundations of Computer Science 2010: 35th International Symposium*, volume 6281 of *LNCS*, October 2010. Springer.

[6] Sean Luke. Essentials of Metaheuristics. Available for free at `http://cs.gmu.edu/$\sim$sean/book/metaheuristics/`, 2009.

[7] Vestal, S. Preemptive scheduling of multicriticality systems with varying degrees of execution time assurance. In *Proceeding of the Real-Time Systems Symposium*, IEEE CSP, pages 239–243. December 2007.