Non-Preemptive Scheduling of Periodic Mixed-Criticality Real-Time Systems

Jasdeep Singh
Luca Santinelli
Federico Reghenzani
Konstantinos Bletsas
Zhishan Guo

CISTER-TR-200303

Non-Preemptive Scheduling of Periodic Mixed-Criticality Real-Time Systems

Jasdeep Singh, Luca Santinelli, Federico Reghenzani, Konstantinos Bletsas, Zhishan Guo

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

https://www.cister-labs.pt

Abstract

In this work we develop an offline analysis of periodic mixed-criticality real-time systems. We develop a graph-basedexploratory method to non-preemptively schedule multiple criticality tasks. The exploration process obtains a schedule for eachperiodic instance of the tasks. The schedule adjusts for criticality mode changes to maximize the resource usage by allowing lowercriticality executions. At the same time, it ensures that the schedulability of other higher criticality jobs is never compromised.We also quantify the probabilities associated to a criticality mode change by using task probabilistic Worst Case Execution Times.A method to reduce the offline complexity is also proposed.

Non-Preemptive Scheduling of Periodic Mixed-Criticality Real-Time Systems

Jasdeep Singh*, Luca Santinelli†, Federico Reghenzani‡, Konstantinos Bletsas§ and Zhishan Guo¶
*†ONERA - DTIS Toulouse, Toulouse, †Airbus Defence and Space, Munich, ‡DEIB - Politecnico di Milano, Milan,
§CISTER Research Centre and ISEP/IPP, Porto, ¶University of Central Florida, Orlando
Email: *jasdeep.singh@onera.fr, †luca.santinelli@airbus.com, ‡federico.reghenzani@polimi.it,
§ksbs@isep.ipp.pt, ¶zhishan.guo@ucf.edu

Abstract

In this work we develop an offline analysis of periodic mixed-criticality real-time systems. We develop a graph-based exploratory method to non-preemptively schedule multiple criticality tasks. The exploration process obtains a schedule for each periodic instance of the tasks. The schedule adjusts for criticality mode changes to maximize the resource usage by allowing lower criticality executions. At the same time, it ensures that the schedulability of other higher criticality jobs is never compromised. We also quantify the probabilities associated to a criticality mode change by using task probabilistic Worst Case Execution Times. A method to reduce the offline complexity is also proposed.

Index Terms

Probabilistic Real-Time System, Mixed Criticality, Graph, Probabilistic Worst Case Execution Time

I. Introduction

Real-time systems must be reliable in functioning within real world timing constraints. Safe upper bounds are determined to the execution time of the tasks in the real-time system as the Worst Case Execution Time (WCET). With the increase in complexity of systems, tight task WCET estimates become difficult to obtain [7], [8]. Mixed-Criticality (MC) systems attempt to cope with the conflict between timing safety and resource efficiency through mode-based approach [4]. Such systems operate by switching between various criticality modes depending on the resource requirements by the executing tasks [6]. MC real-time systems must function at various levels of assurances or criticalities [17].

The term *criticality* is a task property which refers to the importance of the tasks. The term *mode* refers to the run-time property of the task which reflects the higher demand of resource time. A *criticality mode switch* is performed from a lower to a higher criticality mode to provision larger execution time. Classically, the mode switch is a system-wide property and triggers an action to provide more resources to the high criticality tasks, like dropping the execution of lower criticality tasks. A system must be provably schedulable in all the modes, i.e. all the tasks present in a given criticality mode must meet their timing constraints. Therefore the MC problem necessitates special analysis at design time which accounts for mode switch.

A real-time system where at least one of its parameters is described with a probability distribution is called a probabilistic Real-Time System (pRTS). The probabilistic approaches in the real-time systems aim at obtaining probabilistic WCET (pWCET) of the tasks. The pWCET is a worst case probability distribution which upper bounds the execution time of any possible task execution behaviour [7]. It generalizes the notion of worst-case task model with multiple WCET thresholds, each with a probability associated of being exceeded. Since a task executing for longer than a certain threshold triggers a switch to a higher criticality mode, pWCETs allows expressing the probability of mode switch at a certain time.

MC and probabilistic approaches, both tend towards a common objective of finding a method to quantify and reduce resource over-provisioning in the system. Probabilistic models perform this quantification in terms of probability of occurrence of various scenarios (eg. probability of deadline miss). MC model allows to clearly distinguish extreme execution scenarios and performs necessary actions to safely execute most important tasks. Mixed-criticality scheduling is a problem not typically approached with probabilistic models, very few such works (e.g., [1], [2], [13], [8]) can be found among the citations in the MC survey [6]. However, according to the certification guidelines (e.g., branching from the IEC6150, ISO26262 or DO-178C standards), one of the criteria to be considered when designing such a system is the probability/frequency of the system failing at run-time. Since MC reflects system usage, probability of the rare event of system switching to higher criticality (not a failure) must be quantified. Therefore, it sensibly motivates us to approach the MC scheduling problem through probabilities. On one hand it is possible to quantify the pessimism in the system, and on the other hand it is possible to ensure safe execution of all the tasks, especially those in the higher criticalities.

Novel job-level mode switching: For greater efficiency (less over-provisioning of processor resources), in this work we introduce a more-fine grained (i.e., job-level) concept of modes. Whenever a job (i.e., an instance of a periodic task) exceeds the WCET assumed for a given mode, *only that job* switches mode (changes run-time criticality), and more a pessimistic WCET (appropriate for a higher criticality) is assumed. This is in contrast to most classic works, where even one task exceeding its WCET threshold for a mode causes the assumption of more pessimistic WCETs for all tasks through a system-wide mode switch. Job-level mode switch makes the scheduling problem harder, meaning that some lower-criticality tasks might need to be sacrificed. However, our model allows us to limit the amount of dropped tasks, (by comparison, most works assume that all lower-criticality tasks are dropped).

Probability quantification: Using the pWCETs, we obtain the probability of system entering higher criticalities. We can only quantify this probability because we aim to obtain a schedule and the pWCETs do no depend on this schedule.

Therefore, the main problem and contribution of our work can be summarised as follows:

Problem and contribution: The problem approached is to quantify the probability of criticality mode changes in a multi-criticality probabilistic real-time system and find a schedule in offline for each criticality while keeping the maximum possible number of lower criticality jobs. Our work introduces a graph-traversal-based analytic framework for mixed-criticality scheduling on a uniprocessor. Jobs are scheduled non-preemptively with fixed priorities.

- Via the proposed graph-based task execution model, we explore and analyze all the job and mode combinations in order to obtain a resource-efficient schedule of jobs for each task depending on the job criticality level.
- We use probabilities from the task pWCET to quantify the probability of a job entering a higher criticality mode.
- For each job in each of its run-time criticality level modes, we provide a schedule from that point onward which maximizes the system utilization.

The probabilities can only be quantified and not used for decision making process. This is because, while we obtain the schedule, the probabilities come from the pWCET and pWCET is not affected by the schedule. We also present a method to reduce the complexity of our offline process.

State of the art. Work in [15] presents a Mixed-Criticality probabilistic Real-Time System (MC pRTS) model, with quantification of the probability of mode change. It concerns with obtaining probability of system entering higher criticality. It does not obtain a schedule and lacks in obtaining actions to take when the system does enter higher criticality. Our work allows to reason about mode change happening in different specific points in time and being triggered by a specific job. This affords us a deeper understanding of the complex system behaviour and offline elaboration of optimal schedule adaptations.

Scheduling algorithms in [12], [9] and [5] focus on assigning feasible and safe priorities to the tasks. Our approach is a step further as it aims at obtaining and optimizing a schedule in addition to assuring the safety of the highest criticality tasks. This implies maximizing resource usage and allowing lower criticality jobs to execute.

This work fits very well with the Simplex Architecture in real-time system [3] where there is a complex system model which can potentially result in erroneous results, and an assuredly safe system model. The decision logic switches to the safe system model when the complex model results in errors or false values. In our context, the system execution in low criticality is the complex model and the system execution in high criticality is the safe model. The decision logic switches the system to safe model when a task executes for a longer duration. The safe model, that is system in high criticality, is safely schedulable. [11] shows that MC problem lies beyond NP and PSPACE complexity and it is NP in uniprocessor case. Dealing with MC problems will lead to complexity issues.

This paper is structured as follows. In Section 2 we present the notations used in this paper regarding probabilities as well as MC system. Section 3 details the graph model used to obtain the schedule. Section 4 extracts certain metrics from the graph model which are used to decide for an optimal schedule. Test cases are shown in Section 5 to exhibit the advantage of our approach. Section 6 concludes this paper.

II. Notations and Assumptions

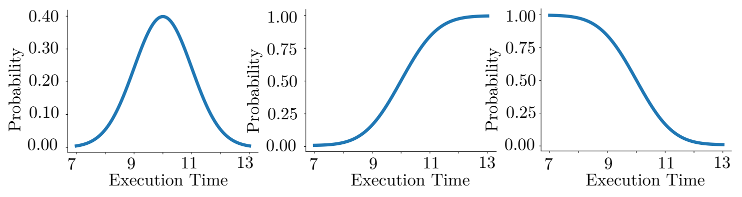
This section defines the notations, conventions, and assumptions used in this work.

A. Probabilistic background

A random variable $\mathcal{C}$ that represents the task worst-case execution time has multiple worst-case values, each with a probability associated. For continuous distributions, the *Probability Density Function (PDF)* $f(x)$ of $\mathcal{C}$ gives the probability that a value of $\mathcal{C}$ lies between a and b: $P(a \leq \mathcal{C} \leq b) \overset{def}{=} \int_a^b f(x)dx$; with $\int_0^\infty f(x)dx = 1$. For discrete distributions, this function is called *Probability Mass Function (PMF)*: $f(x) \overset{def}{=} P(\mathcal{C} = x)$ with $\sum_0^\infty f(x) = 1$. The *Cumulative Distribution Function (CDF)* $F(x)$ representation gives the cumulative probability that a value of $\mathcal{C}$ is less than x as: $F(x) \overset{def}{=} P(\mathcal{C} \leq x) \overset{def}{=} \int_0^x f(x)dx$, for continuous distributions, and $F(x) \overset{def}{=} P(\mathcal{C} \leq x) \overset{def}{=} \sum_0^x f(x)$ for discrete distributions. The complementary of the CDF is called *Complementary Cumulative Distribution Function (CCDF)* $\bar{F}(x)$: $\bar{F}(x) \overset{def}{=} P(\mathcal{C} > x) = 1 - F(x)$, for both continuous and discrete distributions. An example of the continuous version of PDF, CDF and CCDF is depicted in Figure 1a, 1b and 1c.

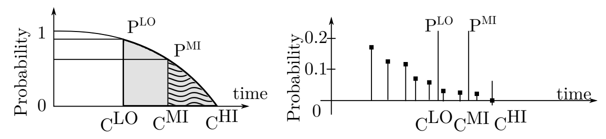
B. Computational model

A set Γ of periodic tasks executing in a system is given. A periodic instance of a task is called a *job* and is represented by J. The random variable $\mathcal{C}$ represents the execution of a job. Its pWCET is the function $f(x)$. C is a worst-case execution value which has probability $\mathcal{P}$ of not exceeding at run-time. We assume that the input pWCET distributions can be discrete or



(a) PDF (b) CDF (c) CCDF

Fig. 1: PDF, CDF and CCDF distributions.



(a) Continuous. (b) Discrete.

Fig. 2: The pWCET distribution and the probability of LO and MI criticality.

continuous. $\mathcal{P}$ is the cumulative probability is extracted from $\mathcal{C}$ given as $\mathcal{P} \overset{def}{=} P(\mathcal{C} > C) = \bar{F}_{\mathcal{C}}(x)$. Job J has an arrival time a and a deadline d. The values of d and a are deterministic single-value parameters. L is the criticality level of the job. The job inherits the criticality level of the task to which it belongs. We assume that the arrival of the first job of each task is always at time zero. We do not consider a system criticality level or mode in our approach. We assign criticality levels to a single job and the criticality mode change of each.

The *hyperperiod* H of the system is the minimum amount of time necessary for the periodic schedule to repeat itself. It is given as the least common multiple of the periods of the tasks. The *hyperperiod job set* is the list of jobs with arrival times in the time-span $(0,H)$: $\Lambda \overset{def}{=} \{J \text{ s.t. } 0 \leq a < H\}$.

C. Criticality model

In this framework, we consider three criticality levels for a job: $\text{LO}, \text{MI}, \text{HI}$; the generalization to more-than-three criticality levels is left as future work. $\text{LO}, \text{MI}, \text{HI}$ respectively represent lowest, middle and highest level of importance/assurance to a job; the order relation is $\text{LO} < \text{MI} < \text{HI}$.

The WCET thresholds for criticalities LO, MI and HI are C^{LO}, C^{MI} and C^{HI}. The probability of exceeding these execution thresholds are extracted from the pWCET. The probability that a job J execution exceeds C^{LO} is $\mathcal{P}^{\text{LO}} = 1 - \sum_{0}^{x=C^{\text{LO}}} f(x)$. The probability that the J executes for more than C^{MI} is $\mathcal{P}^{\text{MI}} = 1 - \sum_{0}^{x=C^{\text{MI}}} f(x)$ The probability of executing for more than C^{HI} is 0 since it is the maximum execution threshold that cannot be exceeded. The probabilistic version of the MC job worst-case execution model gets inspired by [17]. See Figure 2a and Figure 2b for these probabilities and criticality thresholds.

Definition 1 (Job criticality modes)**.** *An L-criticality job J is said to be in L-criticality mode when its execution time is between $\mathcal{C}^{L-1}$ and $\mathcal{C}^{L}$, where L can be* LO, MI *or* HI.

The criticality of a job is its property which corresponds to its importance in the system. Criticality of a job represents the degree of consequence on the system functional safety if its deadline is missed. The criticality mode change of a job is a run-time property.Therefore, a HI-criticality job is: $J \overset{def}{=} (\{(C^{\text{LO}}, C^{\text{MI}}, C^{\text{HI}}), (\mathcal{P}^{\text{LO}}, \mathcal{P}^{\text{MI}})\}, a, d, \text{HI})$; a MI-criticality job is: $J \overset{def}{=} (\{(C^{\text{LO}}, C^{\text{MI}}), (\mathcal{P}^{\text{LO}})\}, a, d, \text{MI})$; a LO-criticality job is: $J \overset{def}{=} (\{(C^{\text{LO}})\}, a, d, \text{LO})$.

The set of all the jobs Λ can be split into three disjointed sets: Λ^{LO}, containing all the LO-criticality jobs, Λ^{MI}, containing all the MI-criticality jobs, and Λ^{HI}, containing all the HI-criticality jobs. It follows that $\Lambda = \Lambda^{\text{LO}} \cup \Lambda^{\text{MI}} \cup \Lambda^{\text{HI}}$.

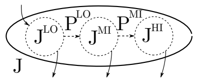
We consider finite pWCET distributions to be always limited with a fixed upper-bound on the WCET. As a side note, for HI-criticality jobs the value C^{HI} can be also computed deterministically with traditional static analysis tools [10] [14]. The WCET thresholds can be chosen arbitrarily, however the property $L > L' \implies C^{L} > C^{L'}$ must be guaranteed. This is because a job has to enter in a lower criticality mode before entering in a higher criticality mode, thus $C^{\text{HI}} > C^{\text{MI}}$.

For our analysis, we consider the following problem: *Given a Mixed Criticality probabilistic Real-Time System with three criticalities, job executions described using pWCET and constrained deadlines non-preemptively scheduled on a uniprocessor machine, what is the optimal schedule when a job enters high criticality mode?*

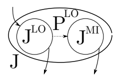
III. Mixed-Criticality Probabilistic Real-Time Model

As the jobs execute, any job can take more time to finish execution and thus enter higher criticality mode. This is especially true for MI and HI criticality jobs. This calls for an analysis which is precise in determining when this can happen. Moreover, the problem demands an optimality, at least in the resource usage. The exact meaning of optimality will be presented in the next section. In any case, global optimality can only be proven when all the possibilities are explored.

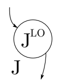
In this section, we propose a model of mixed-criticality system based on a graph structure. With the graphs it is possible to represent a network of possible events that can occur the system at run-time. Then, through graphs we can perform explorations in a space of all possible schedules and aim at optimality. We do so through trees which are derived from the graph model which form a forest to be explored. This tree model will be exploited in Section IV to obtain a schedule.



(a) HI-criticality job.



(b) MI-criticality job.



(c) LO-criticality job.

Fig. 3: The node representations with subgraph for job criticality levels.

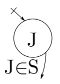


Fig. 4: Representation of early node.

A. Graph structure and construction

We describe the jobs of mixed-criticality probabilistic real-time system and their relations through a *directed graph*. The graph represents the possible job schedules of the system in each hyperperiod. A *directed graph* is defined as a tuple $G = \{V, E\}$, where V is a finite set of elements called nodes and E is the finite set of ordered pairs of elements of V called arcs.

1) Nodes: Each node $J \in V$ of the graph represents the execution of the job J. In the graph, each node represents a job. We directly use the symbol J to represent the node.

Further within each node, there is a subgraph which represents the criticality mode changes of the job during execution.

Definition 2 (Subgraph)**.** *For each node $J \in V$ a subgraph of J is defined as follows:*

- *if J is* HI*-criticality, i.e. $L =$ HI, the node J contains a subgraph with three nodes representing the job criticality modes: J^{LO}, J^{MI}, J^{HI};*
- *if J is* MI*-criticality, i.e. $L =$ MI, the node J contains a subgraph with two nodes representing the job criticality modes: J^{LO}, J^{MI};*
- *if J is* LO*-criticality, i.e. $L =$ LO, the node J does not contain a subgraph because this job does not enter higher modes.*

The subgraphs for HI and MI criticality jobs are depicted in Figure 3a and 3b, respectively. The node for LO criticality job is depicted in Figure 3c. It should be noted that the probabilities labelled are for representation only. That is, the probability to enter HI criticality node is cumulative and not conditional that the job was in MI node. This does not affect the schedule as it only concerns with system probabilities. Since the nodes with subgraphs represent job criticality modes, there are associated WCETs in each of the modes. We do not label them in the graph but we refer to them later while computing the scheduling metrics.

The schedule of the jobs can begin with any job as the first job of a task. The jobs arriving at time zero are defined in the following set.

Definition 3 (Early nodes set)**.** *The* early nodes set *S is a subset of the node set $S \subseteq V$ such that $a = 0$ $\forall J \in S$, i.e. the corresponding job is the first job of a task in the hyperperiod.*

Graphically, we identify the early nodes in the set S with an extra arc entering the node which is without the source node, as shown in Figure 4. These arcs are not considered part of the E set.

2) Arcs: The arcs correspond to the possible sequence relations of the jobs. Each arc $\{J, J'\} \in E$ represents a possible ordering of jobs, in particular, the job J' executes after the execution of the job J: $\{J, J'\} \in E$ if $a < d'$; where d' is the deadline of the job J'. It should be noted that, since the graph is directed, $\{J, J'\}$ is not the same as $\{J', J\}$. Also, no self loop exist, i.e. arcs do not connect to themselves $\nexists \{J, J\} \in E$.

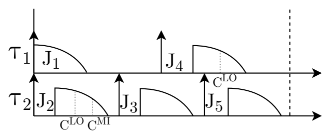
As shown in Figure 3, the nodes of the subgraph are connected by labelled arcs representing the possible transitions of the job criticality modes with the probability label $\mathcal{P}^L$. These transitions are represented as the dotted lines. These are taken because a job demands more execution time and thus enters higher criticality. From scheduling point of view, the dotted transitions represent an uncertainty in the system and do not represent a scheduling decision.

The arc $\{J^L, J^{L'}\}$ exists if $L < L'$ and $\nexists L'' : L < L'' < L'$. It follows that two arcs exist in a HI-criticality job subgraph, one arc exists in a MI-criticality job subgraph.

Definition 4 (Exiting arcs from a subgraph)**.** *A node of a subgraph is connected to another subgraph of the nodes according to the following specification:*

- *An arc $\{J^L, J'^{\text{LO}}\}$ exists if $\{J, J'\}$ exists.*
- *No arc $\{J^L, J'^{L'}\}$ exists if $L' =$ MI or $L' =$ HI.*

These two specifications are put because in our model as a result of bringing notion of criticality mode change to job level. A job always begins execution in LO-criticality mode and then moves to higher criticalities. Therefore, the arcs from other jobs can only enter LO-criticality node because the job always begins execution in LO-criticality. However, within the subgraph the arcs to higher criticality nodes can only come from the lower criticality nodes of the same job. The arcs can exit from any node implying the job can finish execution in any criticality mode. We illustrate this construction with the following example.


(a) Probabilistic job executions of jobs of task set Γ_1 in a hyperperiod.

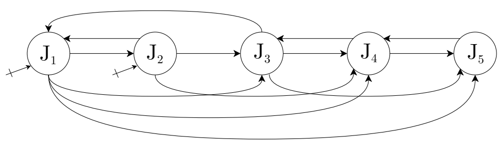
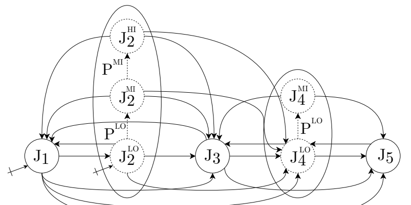

(b) Graph for task set Γ_1.

Fig. 5: Jobs and corresponding graph for task set Γ_1.


(a) Graph with subgraphs for task set Γ_1.

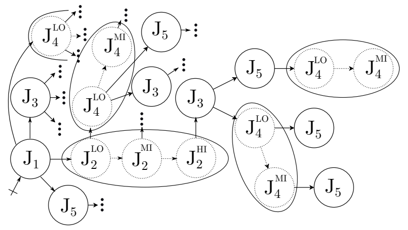

(b) Exploration tree of task set Γ_1.

Fig. 6: Graph and tree for task set Γ_1.

Example 1. *Let Γ_1 be a task set of two tasks with five jobs with probabilistic executions and implicit deadlines in the hyperperiod, as represented in Figure 5a. Job J_2 is a* HI*-criticality job and Job J_4 is a* MI*-criticality job. Thus, $\Lambda^{\mathrm{HI}} = \{J_2\}$, $\Lambda^{\mathrm{MI}} = \{J_4\}$ and $\Lambda^{\mathrm{LO}} = \{J_1,J_3,J_5\}$. Their WCETs for entering* MI *and* HI *criticality are $\mathcal{C}^{\mathrm{LO}}$, $\mathcal{C}^{\mathrm{MI}}$ with $\mathcal{C}^{\mathrm{HI}}$ as the maximum WCET. The graph for the jobs is built as shown in Figure 5b. Each arc represents a possible ordering between two jobs. For example, the arc $\{J_1,J_2\} \in E$ represents the possibility to execute J_2 after J_1. Another example is the missing arc $\{J_3,J_2\} \notin E$: we can not execute J_3 before executing J_2, otherwise J_2 would for sure miss the deadline. The early node set $S = \{J_1,J_2\}$ has been depicted with arrows without source node. The arcs represent possible scheduler choices. In particular, the arcs originating from a node shows the possible choices for the next job to execute. For example, a schedule J_1,J_2,J_3,J_4,J_5 is plausible, while J_2,J_3,J_4,J_1,J_5 is not, because no arc from J_4 to J_1 exists. Considering the criticality levels of the jobs as $L_1 = L_3 = L_5 =$* LO*, $L_2 =$* HI*, $L_4 =$* MI*, we build the subgraph with the showed in Figure 6a. Inside each subgraph, the mode changes are depicted with dotted arrows labelled with the probability. These mode changes represent an event happening at run-time, that can not be forecast during the offline scheduling analysis.*

B. Scheduling tree

We need to explore all the possible combinations of sequences of the jobs in order to obtain a schedule which is optimal in resource usage. To do so, the graph defined above is unfolded into trees which are directed graphs without cycles. The trees represent all possible sequences of job execution; we call them *exploration trees*.

Definition 5 (Exploration Tree)**.** *The* exploration tree *T of a graph G with an early node $\bar{J} \in S$ is defined as a* tree *$T = \{\bar{V},\bar{E}\}$, where $\bar{V}$ is the set of nodes of T and $\bar{E}$ is the set of arcs of T.*

An example of the tree is shown in Figure 6b. Each node of the tree is labelled with a job J. In particular, since the tree is a *rooted tree*, its *root* is defined as the unique node with the label $\bar{J}$. The solid lines represent the scheduling decision in the tree. The dotted lines represent a job entering higher criticality and they are not a part of scheduling as they represent uncertain events in the system.

The set of all exploration trees for different roots is called the *exploration forest F*. A *leaf node J* is a node without successors, i.e. $\mathrm{succ}(J) = \{\emptyset\}$. In order to navigate the tree to obtain a schedule, we define a path.

Definition 6 (Path)**.** *A path in the tree $path(J,J''')$ is defined as a unique sequence of connected arcs starting from the node J to a leaf node J'''; $path(J,J''') = \{\{J,J'\},\{J',J''\},...,\{J'',J'''\}\}$ with $\{J,J'\} \in \bar{E})$ for any J, J', J'' with J''' a leaf node.*

It should be noted that when we refer to the notation $path(J,J''')$, we always refer to a unique path with job J and leaf node J''' unless otherwise specified. Two paths are same if their elements are same. Notation wise, there are many possible paths from a node to a leaf node notated by the same jobs. Also, a path can begin at any node but must end at a leaf node.

Remark 1 (Repeated Node)**.** *A node J' does not exist in tree if it already exists between the root and the desired point of addition.*

$$J' \neq succ(J) \text{ if } \exists\{J,J'\} \in path(\bar{J},J'''), J''' \text{ is a leaf node}, \bar{J} \in S$$

The above remark is to prevent addition of same node in again in the tree. This is to prevent an unrealistic scenario when same job is scheduled more than once.

Example 2. *Using the example test case shown in Figure 5a and its graph in Figure 6a, a portion of the exploration tree is shown in Figure 6b. The tree starts at a root node J_1 because J_1 arrives at time 0. The jobs that can be executed after J_1 are J_2, J_3, J_4, J_5, and hence there are such arcs to those jobs. This way the tree is built by adding possible jobs or jobs in higher criticalities from each node without repetition.*

C. Comments

Each path represents a schedule in the hyperperiod which is composed of nodes with unique labels. The paths which pass through the different criticality nodes of the same job are considered different. For example, a path which passes through LO node of a job J is different from a path which pass through LO followed by MI node of the same job J. This is done to distinguish the paths interpreted as schedules in various criticality modes.

Jobs uniqueness in a path: In any path of a tree $path(J, J''') = \{\{J, J'\}, \{J', J''\}, ..., \{J'', J'''\}\}$, the set of connected node $\{J, J', ..., J'''\}$ has unique job nodes. The nodes do not repeat in the path from root node to a leaf node. Thus, the same job is not schedule twice. At the same time, the graph represents all possible schedules from any job. Exploration tree is the unfolding of this graph to obtain a schedule. Since the graph representation is complete and there are no node repetitions in the tree, all paths are unique.

All schedules represented: The graph is a complete representation of all the precedence among the tasks. The forest of trees is built using the graph which explores all the possible permutations of jobs. Consequently, all the possible permutations of jobs $\{J, J', ..., J'''\}$ are represented with at least one path.

Complexity: There are n the number of jobs in one hyperperiod. In the worst case of the graph, all the job nodes are connected to all the other job nodes, the complexity is $O(n^2)$. The size of the F set, i.e. the number of exploration trees, is the equal to that of S, i.e. the number of jobs arriving at time zero. Regarding the tree complexity, each path represents a possible schedule. Since all the possible schedules are represented in the tree, the worst-case complexity scenario is when all jobs have the same arrival time. All the possible schedules have to be represented and the number of complete path is $O(n!)$.

As we see, the graphs and trees are a representation of all the possible combination of schedules in the hyperperiod. We navigate the tree using the paths. In the next section, we use some metrics to quantify those paths in order to obtain schedules in the hyperperiod. A path can begin from any job node and because we know all the job combinations, we can foresee a criticality mode change of a job and choose a best schedule from that point onward.

IV. SCHEDULABLITY ANALYSIS

In this section, we will use the trees constructed in the previous section to obtain job scheduling. The selection of schedule is based on certain metrics which are presented first; the trees are explored by using those metrics on paths. We also present a method to reduce the complexity of the process of building the offline tree.

We quantify the probability of the system entering MI or HI criticality from the probabilities obtained from the job pWCETs. An important point to note is that this probability does not change depending on the schedule because it comes from the pWCET which in turn does not depend on the scheduling. However, because the probability is obtained by choosing an execution threshold on the pWCET, the probability of any job entering the high criticality depends on this choice. If the probability is higher than desired, changes in the given task parameters are requested.

A. Tree metrics

Here we elaborate the information that is extracted from the graph representation of the system in the previous section.

Response Time: The *response time* of a job is the difference between its finishing time and the arrival time. The value of finishing time depends on both, the execution time, and on the time instant at which the job actually begins execution. Since arrival time is known, the worst-case value of finishing time depends on both the scheduling decision and on the execution of the previous jobs. The Worst Case Response Time (WCRT) is given as:

Definition 7 (Worst-Case Response Time)**.** *Given a node J''^L in the subgraph of the job J'' in a path $path(J, J''')$ for some $J'' : \exists\{J', J''\} \in path(J, J''')$ then the Worst-Case Response time for node J''^L in L criticality, $L = \{\text{LO}, \text{MI}, \text{HI}\}$ is defined as:*

$$WCRT(J''^L) = \max(0, WCRT(J'^L) - a'') + C''^L$$

where a'' is the arrival time and C''^L is the WCET at L criticality mode of the job J''.

In the proposed model, the WCRT is consequently a function of the path, because it is different among different schedules and tasks criticality levels.

Using the above definition, we define the deadline miss of a job as follows. A job represented by its node J'' in a path $path(J, J''')$ is said to have missed a deadline if $WCRT(J''^L) > d'', J'' \in path(J, J''')$ for any L in $\{\text{LO}, \text{MI}, \text{HI}\}$.

Path utilization: Utilization is the processor demand by the jobs scheduled in the path. The utilization as determined by the job characteristics is as follows,

Definition 8 (Utilization)**.** *The utilization for a path $path(J,J''')$ is given as:*

$$U(path(J,J''')) = \frac{\sum C'}{H-a} \forall J' \in path(J,J''')$$

Where a is the arrival time of job J and H is the hyperperiod.

Probability of System Criticality mode: The probabilities come from the pWCETs of the jobs. There exists a probability that a job will take a certain time to execute and in turn, execute in certain criticality mode. The pWCET of the jobs does not depend on the schedule, thus it does not change if the job execution sequence is changed. Since the probabilities are extracted from the pWCETs, these probabilities also remain unaffected by the schedule. Therefore, the only information that can be inferred is the probability that the system will enter higher criticality at a certain point of time.

We do not have to notion of system criticality in this paper. However, in order to relate to the approaches where system criticality is used, we can use the pWCETs to obtain the probability that the system enters a higher criticality. The system enters MI criticality if at least one of the jobs in its path enters MI criticality, i.e. the first jobs enters MI OR the second job enters MI OR..., and so on. We use the law $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ for any two events A and B with $P()$ giving their probability of occurrence [16].

Definition 9 (System Criticality Probability)**.** *For a certain path $path(J,J''')$ for some J, the probability that the system enters* HI *criticality is given as*

$$\mathcal{P}_{sys}^{\mathrm{MI}}(path(J,J''')) = 1 - \prod(1 - \mathcal{P}'^{\mathrm{MI}}) \forall J' \in path(J,J''')\ \ \&: J' \in \Lambda^{\mathrm{MI}} OR\ J' \in \Lambda^{\mathrm{HI}}, J''' \textit{ is a leaf node}$$

Similarly, the probability that the system enters HI criticality is $\mathcal{P}_{sys}^{\mathrm{HI}}(path(J,J''')) = 1 - \prod(1 - \mathcal{P}'^{\mathrm{HI}}) \forall J' \in path(J,J''')\ \ \&\ \ J' \in \Lambda^{\mathrm{HI}}, J'''$ is a leaf node.

It should be noted that this probability is not a path property but a system property. It does not change with the schedule because the total number of jobs remain the same in the hyperperiod. If this probability is higher than a certain allowed maximum probability, it cannot be reduced through scheduling and it must be improved at the level of design before obtaining the pWCETs.

B. System Scheduling

In this subsection, we use the information extracted from the tree in the above definitions to obtain an optimal schedule. In order to obtain a schedule, the exploration tree is pruned by removing all except one optimal path for each job in each criticality. We begin with defining a valid path in the tree.

Definition 10 (Valid Path)**.** *A path $path(J,J''')$ for a node J and a leaf node J''' in a tree is said to be a valid path iff:*

$$WCRT(J'^{\mathrm{HI}}) \leq d' \forall J' \in path(J,J''') \textit{ and } \exists J' \in path(J,J''') \forall J' \in \Lambda^{\mathrm{HI}} \tag{1}$$

According to the above definition, a valid path is not required to pass through all the LO-criticality jobs. Thus, there are paths which may contain only the HI criticality jobs and no MI or LO criticality jobs.

A non-valid path is represented as $\tilde{path}(J,J''')$. A non-valid path can exist by having all the jobs meeting their deadlines but not containing all the HI-criticality jobs and vice versa.

Definition 11 (Dangerous Valid Path)**.** *A valid path $\overline{path}(J,J''')$ is said to be a dangerous path if:*

$$\exists J^L \in \overline{path}(J,J''') : J^{L'} \in \tilde{path}(J,J'''), L > L'$$

A dangerous path is not necessarily a non-valid path. A job might meet its deadline in MI criticality and thus, might seem schedulable. However, the same job might miss its deadline if it enters HI criticality. Such a node belongs to a dangerous path. We extract an optimal valid path defined as follows.

Definition 12 (Optimal Valid Path)**.** *Amongst all the finite possible paths for a node $J \in T$, a valid path until a leaf node J''', $path(J,J''')$ is optimal if the following conditions are met in the order of priority:*

- *the number of* MI*-criticality jobs is maximum in the path $path(J,J''')$;*
- *the number of* LO*-criticality jobs is maximum in the path $path(J,J''')$;*
- *the path $path(J,J''')$ has maximum utilization.*

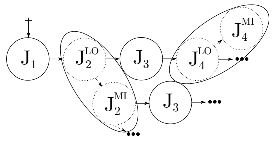


Fig. 7: A portion of the optimized scheduling tree T^{schd} for task set Γ_1.

We look for such optimal paths for each job in each of its criticality. The output of this strategy is a tree T^{schd} with the information listed above. T^{schd} is thus given as,

$$T^{schd} = \text{Optimize}(T - \overline{path}_T(J,J''') - \tilde{path}(J,J'''))\forall J^L \in J, \forall J \in \Lambda$$

where J''' is a leaf node, for all possible $\overline{path}_T(J,J''')$ and $\tilde{path}(J,J''')$ and function Optimize() removes all but one valid and optimal path for each job J according to the Definition 12. The tree $T \in F$ with certain root node $\overline{J} \in S$ is chosen in which there at least one path $path(J,J''')$ for any J with maximum utilization $U(path(J,J'''))$.

To recall, the objective is to find a valid schedule for the jobs. At the end of the analysis we obtain the following.

- A valid path is a schedule with all the jobs in the LO-criticality mode. If there is no such combination possible, that is there is no such path, the schedulability test fails and changes in task parameters are requested.
- For each MI-criticality and HI-criticality job, an optimal valid path is obtained which is an optimal schedule of the remaining scheduled jobs from that point in time onward. Such a schedule necessarily contains all the MI and HI criticality jobs and includes maximum number of LO criticality jobs.

C. *Complexity reduction*

We check the condition $WCRT(J') \leq d'$ whenever a new node J' is added to the tree during its construction. As soon as this condition becomes invalid, the tree is not built in that direction. This is because $WCRT(J') > d'$ implies J' is in a non-valid or dangerous path. That is, the node being added already missed its deadline, then the system should not be scheduled along this path. This way, the tree is pruned during construction. This reduces the complexity of the process of building the trees.

D. *Online schedule:*

From the offline analysis, we obtain a scheduling tree T^{schd} for the task set Γ. The schedule is optimal in resource usage and ensures scheduling of all the high criticality jobs. The system is scheduled from the root node $\overline{J} \in S, T^{schd}$ as the first job. Then the sequence of jobs in T^{schd} is followed.

The online schedule is fixed in the sense that the schedules for each job in each of its criticality is fixed. On the other hand, the schedule is adaptive in the sense that the job criticality mode change selects the fixed schedule from that point onward. At run-time, a job can take more time to execute and enter a higher criticality. This is represented as the dotted transition in Figure 7. This changes the path taken by the system by moving to the higher criticality subgraph of the job. The optimized path from that node onward is already available in the tree T^{schd}. The system continues in this manner for all jobs generated by the task set. This online process goes on until the leaf node job at the end of the hyperperiod. This way, we have optimized the schedule offline and online scheduler simply has to follow the easy to adapt sequence of jobs.

Using our approach, the notion of system criticality does not exist as we only study job criticality mode changes. All the jobs begin execution in their LO-criticality mode and then may move to higher criticality. The scheduling tree is safely and optimally prepared for such criticality mode changes.

Example 3. *For the task set Γ_1, the scheduling tree T^{schd} is shown in Figure 7. The schedule begins with job J_1 followed by J_2, J_3 and so on. If J_2 enters* MI-*criticality, the schedule can safely continue. However, we see that if J_4 enters* MI-*criticality, there is no node for J_5. That is because J_5 can only be schedule if J_4 remains in* LO-*criticality and does not enter* MI-*criticality.*

V. Experiments

In this section we propose a non exhaustive but a realistic test case to explain all the contributions made. We also discuss complexity in this section. The task sets and the scheduling which are obtained from their graph trees are presented. We show the advantage of our approach by maximizing the number of LO-criticality jobs which are executed when certain jobs enter higher criticality modes. This is done as a result of finding an optimal schedule. Our approach is opposed to the classical one where all the LO-criticality jobs are dropped when the system enters high criticality.

The task set Γ_2 is shown in Table I. There are 5 tasks, task τ_1 being of HI-criticality, tasks τ_2 and τ_4 of MI-criticality and tasks τ_3 and τ_5 of LO-criticality. The corresponding periods and execution times C_i^{LO}, C_i^{MI} and C_i^{HI} are also shown. All times are in units. The task set has 10 jobs in the hyperperiod equal to 30 time units. These jobs are shown in Figure 9. As an

	$T_i = D_i$	Criticality	C_i^{LO}	C_i^{MI}	C_i^{HI}	P_i^{LO}	P_i^{MI}
τ_1	10	HI	1	1.5	2.5	0.3	0.1
τ_2	30	MI	2	6	-	0.2	-
τ_3	15	LO	2	-	-	-	-
τ_4	30	MI	3	7	-	0.1	-
τ_5	10	LO	2.5	-	-	-	-

TABLE I: Task set Γ_2.

Job	Criticality	Schedule	Utilization
J_{11}	MI	$J_{11}, J_{51}, J_{41}, J_{31}, J_{52}, J_{12}, J_{32}, J_{13}, J_{21}$	0.883
J_{11}	HI	$J_{11}, J_{51}, J_{41}, J_{31}, J_{52}, J_{12}, J_{32}, J_{13}, J_{21}$	0.983
J_{12}	MI	J_{32}, J_{21}, J_{13}	0.55
J_{12}	HI	J_{32}, J_{21}, J_{13}	0.65
J_{13}	MI	J_{21}	0.75
J_{13}	HI	None	0.15
J_{21}	MI	None	0.25
J_{41}	MI	$J_{31}, J_{52}, J_{12}, J_{32}, J_{21}, J_{13}$	0.75

TABLE II: Schedules of Γ_2 when jobs enter MI or HI criticality

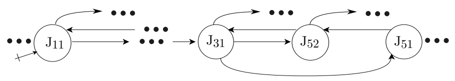


Fig. 8: Graph for the jobs in Γ_2.

example, if first job of τ_1 executes, it can be followed by first jobs of the rest of the jobs or the second job of itself. We explore such possibilities as follows.

A portion of the graph for this task set is shown in Figure 8. There are 134 nodes in the graph. It is developed into trees of the forest. A portion of the one of the trees in the forest the with J_{11} as root is shown in Figure 10. There are 502,063 nodes in this tree. From the extracted valid paths in LO-criticality mode of the jobs, the following scheduling is proposed: $J_{11}, J_{51}, J_{41}, J_{31}, J_{52}, J_{12}, J_{32}, J_{13}, J_{21}, J_{53}$ with utilization of 0.65. The probability that system enters MI criticality is 0.22. The probability that system enters HI criticality is 0.099. For each job in higher criticality mode, the optimized schedule to be taken by the jobs is shown in Table II. It also shows the corresponding utilization of the remaining schedule. The table also shows that all the jobs in MI and HI-criticality are safely schedulable. We see that the job J_{13} enters HI-criticality, the job J_{21} cannot be executed anymore. The job J_{21} is the last job in the hyperperiod. Therefore, the schedule for these two cases are 'None'. This shows the advantage of our approach. If J_{13} enters only MI-criticality, there is room for execution of J_{21}. It is only when J_{13} enters HI-criticality is when J_{21} is needed to be dropped from execution.

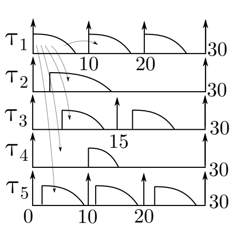


Fig. 9: Jobs of task set Γ_2

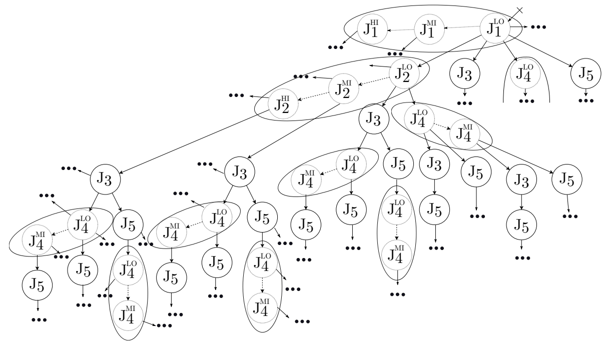


Fig. 10: A portion of an exploration tree for task set Γ_2 with the root node J_1.

Apart from the results obtained above, in order to magnify the benefit of our approach, we examine another task set Γ_3 shown in Table III. In this case, there are four tasks, task τ_1 being of HI-criticality, tasks τ_2 of MI-criticality and tasks τ_3 and τ_4 of LO-criticality. The corresponding periods and execution times C_i^{LO}, C_i^{MI} and C_i^{HI} are also shown. The task set has 7 jobs in the hyperperiod equal to 30 time units. In this scenario, the proposed LO-criticality schedule is:
$J_{11}, J_{41}, J_{31}, J_{21}, J_{12}, J_{13}, J_{32}$. the utilization of this schedule is 0.966.

	$T_i = D_i$	Criticality	C_i^{LO}	C_i^{MI}	C_i^{HI}
τ_1	10	HI	5	7	8
τ_2	30	MI	3	4	-
τ_3	15	LO	4	-	-
τ_4	30	LO	3	-	-

TABLE III: Task set Γ_3.

No. of jobs	No. of tree nodes	No. of jobs	No. of tree nodes
3	25	7	125941
4	157	8	1.5 million
5	1261	10	2.5 million +
6	11965	-	-

TABLE IV: Number of jobs vs number of tree nodes.

We see that task τ_1 is barely schedulable in the HI-criticality mode where it has an execution time of 8 with a period of 10. The task τ_1 begins in LO-mode. However, if the job J_{11} enters HI-criticality, the valid optimal path from the node of J_{11} in HI-criticality is the one which contains all the HI-criticality jobs meeting their deadlines and then maximum number of MI-criticality jobs, followed by maximum number of LO-criticality jobs. From the tree, there exists a path: J_{11} from LO to MI to HI, J_{21} LO to MI, J_{12} from LO to MI to HI and J_{13} from LO to MI to HI. Thus the schedule from J_{11} in HI-criticality is: J_{11},J_{21},J_{12},J_{13}. The utilization of this schedule when all the HI-criticality and MI-criticality jobs are in highest mode and is 0.933. In the case where all LO-criticality jobs are dropped, i.e. if the schedule from J_{11} in HI-criticality would not include MI-criticality job J_{21}, the utilization would have been 0.8. However, as we see, there is room for including J_{21} to execute.

Moreover, the schedule after J_{11} in HI-criticality does not include J_{41}. This is because the path from the J_{11} HI-criticality node going through J_{41} does not reach a node for J_{13} in HI-criticality, causing it to be dangerous path. In this case the maximum utilization is 1.033 which is clearly undesirable.

Complexity: Firstly, the complexity of building a tree depends on the total number of jobs. It increase with the number of MI and HI jobs as it causes more number of nodes in the graph. Then, it depends on the given pWCET and the WCET values of C_i^{LO}, C_i^{MI} and C_i^{HI} extracted as well as the deadlines of the jobs. The more the WCET values are closer to the deadline, the less exploration branches are there to explore. This also means, there are less possible schedules. This is because there are less number of valid paths. A possible complexity of number of jobs vs the number of nodes is shown in Table IV. The complexity is shown in the worst case when all the jobs arrive at the same time. Actual complexity will be less than that is shown.

VI. Conclusion

In this paper we have used a graph based exploratory method to obtain a schedule for Mixed Criticality probabilistic Real-Time System. The schedule is obtained for each job in each of its criticality. Each of those schedules are optimized in resource usage and ensure safely schedulable for safety criticality applications. At the same time, the complexity of the exploration process is reduced. In addition, the probabilities are quantified relating to the system and the schedule. As a next step, we intend to introduce mixed criticality defined using the response time which is application oriented, where probabilities come into decision making process. We intend to further optimize such a construct to obtain schedules. We will also study the affect of task dependence in mixed criticality probabilistic real-time system.

References

[1] B. Alahmad and S. Gopalakrishnan. A risk-constrained markov decision process approachto scheduling mixed-criticality job sets. In *Workshop on Mixed-Criticality Systems*, 2016.

[2] B. Alahmad and S. Gopalakrishnan. Risk-aware scheduling of dual criticality job systems using demand distributions. *LITES*, 5(1):01:1–01:30, 2018.

[3] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha. The system-level simplex architecture for improved real-time embedded system safety. In *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 99–107, April 2009.

[4] S. Baruah and A. Burns. Implementing mixed criticality systems in Ada. In *16th Ada-Europe Conference*, pages 174–188, 2011.

[5] S. Baruah, A. Easwaran, and Z. Guo. Mc-fluid: Simplified and optimally quantified. In *2015 IEEE Real-Time Systems Symposium*, pages 327–337, Dec 2015.

[6] A. Burns and R. Davis. Mixed criticality systems – a review (12th ed.). Technical report, Dept of CS, U. of York, UK, Mar. 2019.

[7] R. I. Davis, A. Burns, and D. Griffin. On the meaning of pwcet distributions and their use in schedulability analysis. In *In Proceedings Real-Time Scheduling Open Problems Seminar at ECRTS*, 2017.

[8] Z. Guo, L. Santinelli, and K. Yang. EDF schedulability analysis on mixed-criticality systems with permitted failure probability. In *Proc. RTCSA*, 2015.

[9] S. Huang, Y. Zhu, and J. Duan. A new scheduling approach for mix-criticality real-time system. In *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 43–46, June 2013.

[10] S. Jiménez Gil, I. Bate, G. Lima, L. Santinelli, A. Gogonel, and L. Cucu-Grosjean. Open challenges for probabilistic measurement-based worst-case execution time. *IEEE Embedded Systems Letters*, 9(3):69–72, Sep. 2017.

[11] R. Kahil, D. Socci, P. Poplavko, and S. Bensalem. Algorithmic complexity of correctness testing in mc-scheduling. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems, RTNS 2018, Chasseneuil-du-Poitou, France, October 10-12, 2018*, pages 180–190. ACM, 2018.

[12] H. Li and S. Baruah. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In *Proceedings of the 2010 31st IEEE Real-Time Systems Symposium*, RTSS '10, pages 183–192, Washington, DC, USA, 2010. IEEE Computer Society.

[13] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran. Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, RTNS '17, pages 237–246, New York, NY, USA, 2017. ACM.

[14] F. Reghenzani, G. Massari, and W. Fornaciari. The misconception of exponential tail upper-bounding in probabilistic real-time. *IEEE Embedded Systems Letters*, pages 1–1, 2018.

[15] J. Singh, L. Santinelli, G. Infantes, D. Doose, and J. Brunel. Mixed criticality probabilistic real-time systems analysis using discrete time markov chain. In *6th International Workshop on Mixed Criticality Systems (WMC)*, 2018.

[16] T. T Soong. *Fundamentals of probability and statistics for engineers*. 01 2004.

[17] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS)*, pages 239–243. IEEE Computer Society, 2007.