Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Jasdeep SINGH

le vendredi 6 mars 2020

Titre :

Schedulability Analysis of Probabilistic Real-Time Systems

École doctorale et discipline ou spécialité :
ED MITT : Informatique et Télécommunications

Unité de recherche :
Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :
M. Luca SANTINELLI (directeur de thèse)
M. Jean-Loup FARGES (co-directeur de thèse)

Jury :
Mme Christine ROCHANGE Professeure Université Paul Sabatier - Présidente
Mme Laura CARNEVALI Associate Professor University of Florence
M. Laurent GEORGE Professeur ESIEE Paris
M. Zhishan GUO Associate Professor University of Central Florida
M. Didier LIME Maître de conférences École Centrale de Nantes - Rapporteur
M. Luca SANTINELLI Ingénieur de recherche Airbus - Directeur de thèse
M. Pierre SIRON Professeur ISAE-SUPAERO
M. Enrico VICARIO Professor University of Florence - Rapporteur

Acknowledgements

First and foremost, I would like to pass my heartfelt thanks to my supervisors Luca Santinelli, Guillaume Infantes whose place was later taken by Jean-Loup Farges. With them, I would like to sincerely thank David Doose and Julien Brunel, all of them stood by me through the learning process of the thesis. I wish to express my sincere gratitude to Zhishan Guo, Konstantinos Bletsas and Federico Reghenzani who helped me shape my thesis with their immense support. I am indebted to all of them.

My heartfelt thanks to the thesis reviewers Enrico Vicario and Didier Lime for their help in making the thesis as refined as possible. I must not forget to thank the unknown reviewers of the accepted as well as unaccepted articles as they helped to improve the thesis.

My sincere gratitude to all the colleagues at ONERA and ISAE-SUPAERO who contributed to my work environment in invaluable ways. A million thanks to my friends and family for their continuous support through this journey.

Contents

List of Figures

List of Tables

Table of acronyms

BCET *Best Case Execution Time*

CCDF *Complementary Cumulative Density Function*

CDF *Cumulative Density Function*

CTMCh *Continuous Time Markov Chain*

DTMCh *Discrete Time Markov Chain*

EDF *Earliest Deadline First*

FoS *Factor of Safety*

FP *Fixed Priority*

ICDF *Inverse Cumulative Density Function*

MBPTA *Measurement Based Probabilistic Timing Analysis*

MC *Mixed Criticality*

MCh *Markov Chain*

PDF *Probability Density Function*

PMF *Probability Mass Function*

pRTS *Probabilistic Real-Time System*

pWCET *Probabilistic Worst Case Execution Time*

pWCRT *Probabilistic Worst Case Response Tim*

SPTA *Static Probabilistic Timing Analysis*

WCET *Worst Case Execution Time*

WCRT *Worst Case Response Time*

CHAPTER 1

Introduction

> If you know the enemy and know yourself, you need not fear the results of a hundred battles.
>
> *Sun Tzu*

The objective of any act beyond the basic needs of food, shelter and clothing by the humans tends to have the direction which eases the human life for the same basic needs. Now that curiosity to look for these acts is born and ambitions grew, it has made us develop great machines, from the primitive machine inventions by Leonardo da Vinci to the self landing rockets boosters by SpaceX. With this progress, the complexity of understanding the systems rose, and the functional reliability of the machines, computer systems in our case, became a question. So did the complexity of the computer systems which must perform in real world timing constraints, which are called real-time systems. When computer systems are applied for scenarios which are safety critical, like those in satellites or aircraft controls, they need to be reliable. If they are not reliable, they end up becoming an enemy which threaten human life, like an aircraft losing its control. The above quote by Sun Tzu from The Art of War says exactly that, to know your enemy. In this thesis we explore such systems which are applied in real world scenarios, the need for them, the challenges that exist today and the research undertaken for the solutions proposed to those challenges.

L'objectif de tout acte au-delà des besoins fondamentaux de nourriture, d'abri et de vêtements par les humains a la direction qui facilite la vie humaine pour les mêmes besoins de base. Maintenant que la curiosité de recherche est né et les ambitions ont grandi, il nous a fait développer de grandes machines, comme des inventions de machines primitives de Léonard de Vinci et les propulseurs de fusées à atterrissage automatique de SpaceX. Avec ces progrès, la complexité de la compréhension des systèmes a augmenté et la fiabilité fonctionnelle des machines, des systèmes informatiques dans notre cas, sont devenues une question. Même pour la complexité des systèmes d'ordinateur qui doivent fonctionner dans les contraintes temps réelles, appelés systèmes en temps réel. Lorsque des systèmes informatiques sont utilisés pour des scénarios critiques pour la sécurité, comme ceux des satellites ou les commandes des avions, elles doivent être fiables. S'ils ne sont pas fiables, ils finissent par devenir un ennemi qui menacent la vie humaine, comme un avion qui perd son contrôle. La citation ci-dessus par Sun Tzu de «The Art of War» dit exactement cela, pour connaître votre ennemi. Dans cette thèse, nous explorons de tels systèmes qui sont appliqués dans des scénarios du monde réel, leur besoin, les défis qui existent aujourd'hui et les recherches entreprises pour les solutions proposées à ces défis.

1.1 Real-Time Systems

Computer systems which operate under real world timing requirements are called real-time systems. That is, real-time systems must produce the result of their functionality within a timed deadline. These are practical applications of digital computers intended to serve a dedicated function. At the same time, the functions they perform must be completed by a certain timed limit. There are several examples of such systems ranging from highly important avionics systems in the aeroplanes or satellites to household items like washing machines and microwaves. A washing machine has a dedicated function which is a cycle of various digital and physical executions to be completed by a certain time in order to wash clothes in the real world. Similarly, satellites have dedicated function of collecting, processing and transmitting data to the earth stations, also done by a certain timed limit. Real-time systems are classified as hard and soft; *Hard real-time systems* are those which can never afford to defy a timing constraint, like satellite systems; *Soft real-time systems* can afford to miss some deadlines.

Real-time systems are composed of various *tasks*. These tasks are the unit of execution which is essentially a code which takes certain inputs and produces outputs. Task has an execution time which is the time it takes to finish execution. Task has a deadline which is the maximum allowed time to end its execution after it is released. Task can also be periodic in which case it repeats execution in a certain period. An example of periodic task is one which takes data from a sensor at a certain frequency. In application, there are more than one such tasks which may be dependent on one another, e.g. output of one task is the input for another. These tasks require *resources* to executed, most importantly a processor for execution, in conjunction with memories and buses for data transport. Figure 1.1 shows such a depiction of a real-time system.

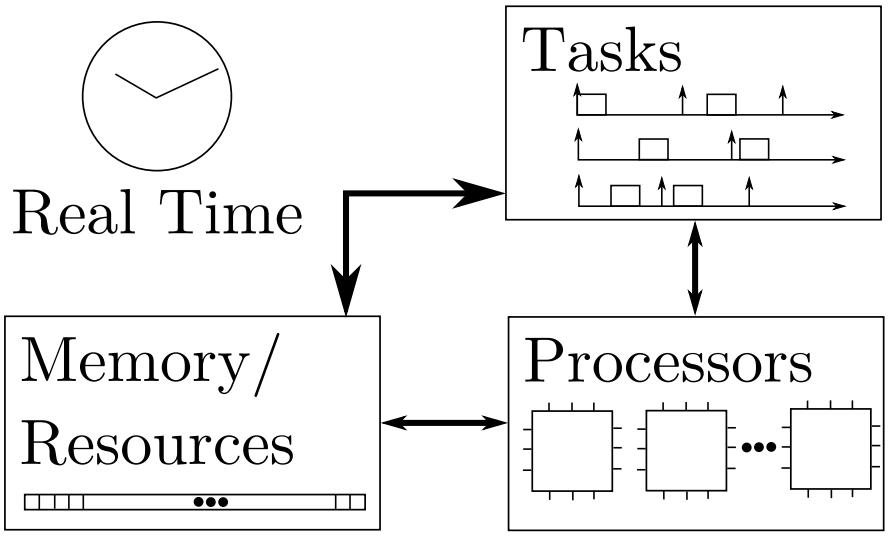


Figure 1.1: A depiction of a real-time system.

For example, a washing machine has a task to wash the clothes followed by a task responsible for drying operation, all requiring processor and inputs like water level and total weight.

Real-time systems must be *predictable* for numerous reasons. The reasons include highly important ones like cost of implementing a satellite to the those of lower importance like satisfactory usability of

video streaming. That is, there must be confidence before its application that the system will function as designed. Predictability is this confidence associated with the application.

For real-time systems, predictability is essentially related to timing characteristics of the tasks. Each task takes a certain time to execute and produce its output. The requirement of timed limit of the tasks in the real-time systems demands timing analysis of the system. Timing characteristics depend on the system hardware and on the real-time demands of the application. E.g., an avionic system is required to calculate airspeed every 10 milliseconds. The hardware frequency must support that under all circumstances. Timing analysis is a study on a given real-time system in order to predict the time the tasks might take to execute.

The emphasis on the previous sentence is on the word '*might*'. The exact duration of the task execution is extremely difficult to obtain. This is because the conditions in which the task executes may change during application. For example, a satellite heats up in the sun and cools down in the eclipse portion of its orbit (behind earth away from the sun). This change in temperature affects the material of the buses and processor which changes the rate of flow of electrons, in turn affecting the rate of flow of data. This affects the task execution duration because it uses that data to produce output. So, even if the timing characteristics are determined by hardware, the execution time may change due to environment of the application. Because of this uncertainty in determining the exact execution duration, a worst case upper bound is obtained called *Worst Case Execution Time (WCET)*. WCET is the maximum amount of time the task takes to finish execution. For example, if we only consider temperature change in the case of the satellites, the WCET of the tasks will consider the fact that the resistance increases of the conducting material with increase in temperature. As a result of the timing analysis, a task will never execute for more than its WCET. The literature also defines a minimum amount of time task requires to execute called *Best Case Execution Time (BCET)*. These are represented in the Figure 1.2. There are various methods at present to determine WCET or BCET like measurement based analysis, static analysis etc. and is wide field of research. Determining the value of WCET or BCET is a subject of timing analysis and is not the focus of this study.

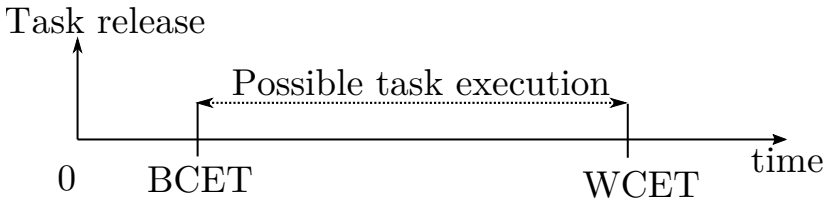


Figure 1.2: Task execution times lies between the BCET and WCET for predictability of task execution.

In order to ensure that the tasks are able to execute within their timing constraints or deadlines, the tasks are scheduled. *Task scheduling* is a problem to obtain an arrangement of task ordering or sequencing which provides them enough room for execution based on their WCET and ensures that each task has sufficient room to execute by its deadline. Scheduling is the result of the requirement of predictability by imposing a specific order of task executions and preventing random and on-the-go executions. Random order of task executions may or may not be completed by the deadlines. This uncertainty is not acceptable. Task scheduling can be online, i.e. task sequence is determined during the application; or offline, i.e. task sequences are known beforehand and fixed. Various algorithms or schemes for scheduling exist like Earliest Deadline First (EDF), Fixed Priority (FP), Round Robin (RR),

etc. In addition, scheduling can be preemptive or non-preemptive. Preemption means that an already executing task can be interrupted by another task which demands immediate execution. Preemption is or is not allowed by the scheduler and therefore is a scheduling property.

A *schedulability analysis* is thus performed which is an extensive, exhaustive and complete formal analysis of the real-time system tasks which guarantees their executions with respect to their deadlines. That is, a given set of tasks is scheduled using an algorithm. The objective of the respective schedulability analysis is to ensure that the scheduling using that algorithm is feasible. It must ensure that no task scheduled using the scheduling algorithm misses its deadline.

The analysis is usually performed offline because it can be extremely complex and infeasible to do so online. An analysis can be an exploration of the possibilities, which can only be feasible offline. An empirical condition like a processor demand criteria can be determined online but is usually not enough to ensure predictability.

In real scenarios, the actual execution time of the task is rarely equal to its WCET and always lies between WCET and BCET. There is always an uncertain gap between actual execution and the WCET of the task. Since the scheduling is done based on the WCET, this gap leads to potential idle time of the processor and induces pessimism in the system. Pessimism is to the over-allowance of resources to the tasks for their execution. A scheduling which is based on WCET does not account for this pessimism and results in an overall pessimistic system. This pessimism is not necessarily undesirable from predictability point of view. Pessimism at least ensures functionality of the system. However, rigorous formal modelling of this problem has the potential of quantifying and controlling this pessimism. In this thesis we approach this very problem. We explore a set of possible solutions which also define the direction the research has taken through the study.

1.2 Probabilistic Real-time Systems

Probabilities are extensively used in modelling science, finance and various other fields. It allows us to see the random choices being made everywhere in a formal way. This way, we are able to dive deeper into the meanings of things and try to make use of them.

Probability is the the measure of an event occurring which can be quantified as the number of times it can occur divided by the total number of possible outcomes for the trial. For example, an unbiased dice with numbers $1,2,3,4,5,6$ is rolled. The probability of obtaining the number 2 is $1/6$ where 6 is the total number of possible outcomes when the dice is rolled. The term 'Probabilistic' is sometimes used interchangebily with 'stochastic'. In this text we only use 'probabilistic' referring to something involving probability. The term 'stochastic' should be used in relation to a process which involves making reasonable predictions of choices. The measure of probability to these predictions leads to the use of the term 'probabilistic'. The term 'probabilistic' should be used in relation to the measure of a trial.

Let us recall the statement that the pessimism in scheduling task using its WCET can be quantified and controlled using formal approaches. Recent efforts have been made to do so using probability

theory, specifically in timing analysis. The task execution time can take any value which is upper bounded by the WCET. Each of those execution times has an associated probability of occurence. This results in a probability distribution of execution times. We can obtain many such distributions from task behaviour under different scenarios. A single probability distribution can be obtained which tightly upper bounds all the execution time distributions. This upper bound is called a *probabilistic Worst Case Execution Time (pWCET)*, shown in Figure 1.3 with the pWCET ICDF. The figure shows various possible execution time distributions with the dotted line which are PDFs. The pWCET is shown as an ICDF which upper bounds all those distributions.

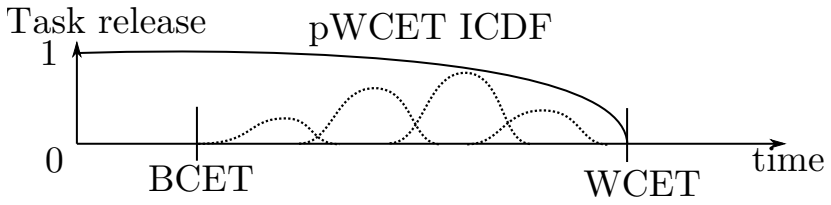


Figure 1.3: Task probabilistic execution through pWCET represented as ICDF with BCET and WCET.

pWCET is a worst case probability distribution which upper bounds all possible execution times of a task. It is a probability distribution which can be continuous or discrete. The method to obtain pWCET is not covered in this thesis and is assumed as given. For this thesis, the task execution is described by pWCET instead of WCET. A real-time system in which at least one property is described using a probability distribution is called *probabilistic real-time system (pRTS)*.

Probability can potentially quantify other task properties too like probabilistic period and deadline which is left for future studies. For example, probabilistic period interprets as probabilistic inter arrival time of the tasks. For our study, the task period and deadline are deterministc (fixed) and not probabilsitic.

Probabilistic representation of task execution allows us to quantify the occurrence of certain execution time through probabilities. Evidently, the pWCET distribution contains more information within itself than a deterministic WCET about task execution duration. Since there is more information, the methods to study and utilize the pWCETs should exploit this quality. With this we can foresee that the methods for schedulability analysis which we intend to study in this thesis will tend to have high complexity. Figure 1.4 shows an example of task set execution where the tasks are defined with WCET or pWCET. Scheduling of tasks using WCET is less complex because the deterministic values of WCET are used. However, the exact instant at which tasks finish execution is unknown. On the other hand, pWCET contains the probabilistic information that task execution finishes or begins after delay from another task. pWCET scheduling analysis can lead to higher complexity but more control over resource usage through scheduling.

Going back to the notion of predictability and the requirement of schedulability analysis, careful observation is needed when we have pWCET. By including a probability distribution in the real-time system analysis, we have a method to quantify the overall pessimism. Naturally, the guarantees which will be provide therafter will also be probabilistic. For hard real-time systems, the resulting probabilities can be upper bound by the worst case value. As we will see, the pessimism in the system analysis can be quantified with probabilities but the analysis must aim at predictability. There are several challenges

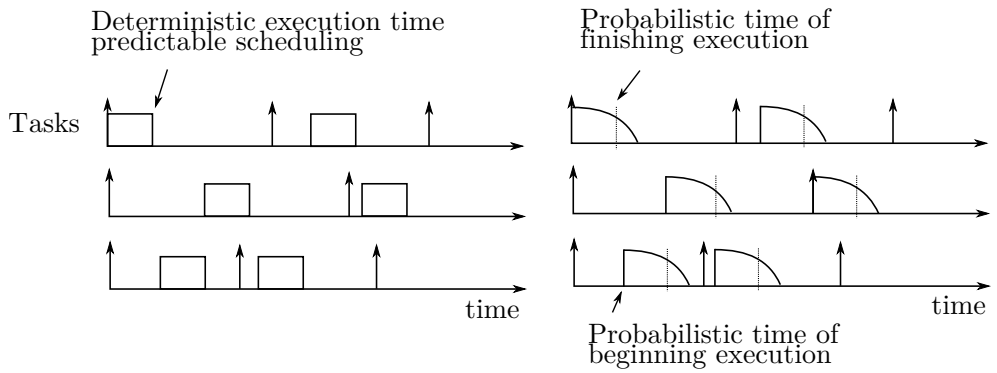


Figure 1.4: Tasks executions with WCET(left) or pWCET(right) .

in this direction.

Probabilistic Interactions: Scheduling a task with pWCETs enables probabilistic interactions between them. The execution of a task can be delayed by other tasks: by a task executing immediately before which pushes the execution of the current task: another task preempting the current task; etc. Since pWCET describes their execution, all these delays to the execution are probabilistically known. This affects the finishing time of the tasks which in turn affects whether the tasks meet their deadline constraints. When designing a schedulability analysis, these interactions have to be formally captured.

Discrete vs Continuous distribution: The pWCET can be mathematically defined as a continuous or discrete distribution. Discrete distribution gives the probability of occurrence of a certain single value of execution time. On the other hand, continuous distribution gives the probability of occurrence in an interval of execution time. Probability obtained from continuous distribution at a single value is zero. From the real-time point of view, this poses a challenge. As we will see, coping with probabilistic interactions involves performing certain operations on the pWCET distributions. It needs to be ensured that the probabilistic information is not lost after an operation on the distributions. At the same time, the nature of the distribution must be taken into account. If it is done, the mathematical representation of a continuous function after the operation becomes challenging. As for the discrete distribution, the system events can occur in between the discrete values represented by the distribution. Therefore, the probabilities at the discrete values must also be an upperbound in the interval. This must be carefully addressed and handled.

Determinism vs Non-determinism: It is required to clarify the difference between non-deterministic and probabilistic system. *Deterministic system* is one in which all of its functionalities are known and the system is assured to function as desgined. Deterministic systems are predictable as they do not deviate from the desired operation. *Non-deterministic* system is one whose output cannot be predicted for the same input because it contains uncertain choices within. It contains randomness at some point in its process which cannot be determined beforehand. For example, hardware random number generator which generates random numbers from physical input is a non-deterministic system.

If the random choices within the non-deterministic system have a probability of occurrence which is known, the non-deterministic system becomes a *probabilistic system*. That is, in a probabilistic

system there is a measure of probability to make a predicition of the future states in a non-deterministic system. In this case there are random choices being made but all the possible random choices are known and each has a probability associated. The associated probability is usually described by a distribution.

If these non-deterministic systems can be bounded, they tend to become deterministic systems. That is, if the maximum and minimum value of its output is determined, then it can be seen as a deterministic system with a worst case or best case output. Task execution is such an example. Task taking a certain execution time can be seen as a random choice taken by the task, making the task itself a non-deterministic system. Task execution bounded with WCET and BCET makes it a deterministic system which is predictible. On the other hand, non-deterministic task associated with the probability distribution pWCET makes it a probabilistic system. The challenge from this perspective is to obtain schedulability analysis model which is inherently probabilistic which models randomness.

Safety: In the context of real-time systems, *safety* refers to the safety of operation of the system. This means, the system is predictable and will function as desired. A deterministic system is always safe. A non-deterministic system is safe if the random choices in the system are bounded. In the context of probabilistic system, the notion of safety dives deeper in the analysis and overall safety is ensured at the bounds. Functionality of the Probabilistic systems contains probabilistic properties within. In probability, safety links to pessimism. Pessimistic choices like worst case probabilities ensure that the actual probability is always lower. For example, actual probability of failure of a system is 0.1. In that case, the probability of failure of 0.2 from the analysis is safe while 0.05 is not safe. Thus, it must be ensured that all the operations performed during the analysis are at least pessimistic, if they are not exact. An exact analysis must be proven so through rigorous model checking.

Scalability: *Scalability* refers to the ability of an analysis to handle larger number of tasks. Scalability refers to the complexity of the analysis. A simple enumerations of all the possible scenarios (like state space exploration in Petri Net), increases in complexity as the number of tasks increases. On the other hand, a simple empirical evaluation which does not depend on the number of tasks has low complexity . Enumeration guarantees completeness of the analysis and confirms schedulability. Empirical evaluation need not guarantee a completeness of the analysis. Scalability of an analysis must be evaluated and ensured feasible. For methods with high complexity, justification must be made as to what is gained with this increase in complexity.

In this thesis we will explore and use such a model for analyzing pRTSs in the Chapter 3. In this direction of exploration, we also encounter another approach towards coping with pessimism as we explain further.

1.3 Mixed Criticality Real-Time Systems

Let us again recall the statement that the pessimism in task execution from WCET can be quantified and controlled using formal approaches. In this direction, another field of extensive research is that of Mixed Criticality (MC) systems. MC real-time systems contain tasks of various importance or criticalities. In two criticality systems, two types of tasks exist which are high criticality and

low criticality. High criticality tasks are those which are of higher importance for correct system functionality. Low criticality tasks are of lesser importance for system functionality. Higher criticality tasks must always be executed and ensured schedulable, even if it requires dropping of low criticality tasks.

This notion of criticality then extends to the notion of system criticality 'mode'. The high criticality tasks are given with two WCETs, one greater than the other. The higher WCET is more pessimistic than the lower WCET. Low criticality tasks have only one WCET. In system high criticality mode, it is imperative that the high criticality tasks execute and are scheduled using the larger WCET. This way, the system high criticality mode allows more room for the execution of high criticality tasks. In this case, the system usually chooses to drop the low criticality tasks. In the system low criticality mode, the high criticality tasks are scheduled using the lower WCET. The low criticality tasks are effectively scheduled in the gap between the two WCETs of the high criticality tasks. At this point, it is natural to raise the question of how can there be two different worst cases of execution time. It is simply something embedded in the literature and we will continue to use this to avoid ambiguity with other references.

Thus, a real-time system which is composed of tasks with more than one criticality level is called a Mixed Criticality real-time system. It is a way to quantify pessimism in the classical models by having multiple WCETs for each level of pessimism. It allows to use a common set of resources for different scenarios of the system application. For example, the aircraft controls are of prime importance in an aircraft while the in-flight entertainment system is not. Whenever required, the system can drop the entertainment systems and allow more execution time for the flight controls to ensure that they execute completely. An example is shown in Figure 1.5 where there are two high criticality tasks and two of low criticality. Both the tasks are scheduled using their low WCET. When the system is triggered to enter high criticality mode, the low criticality tasks are dropped from execution. The high criticality tasks are then allowed more time to execute according to their high WCET.

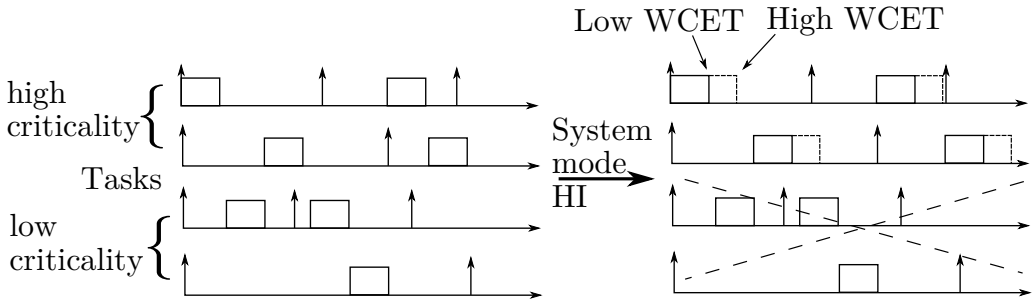


Figure 1.5: Mixed Criticality scheduling.

MC systems pose a scheduling problem in order to be predictable. Current approaches in practice are based on hardware partitioning based on the criticality levels. That is, each criticality is given a dedicated set of resources and the schedulability of the tasks of that criticality is ensured. This readily poses the problem of having large set of hardware, in turn increasing the cost. In this case, system truly becomes mixed criticality if there is some sharing or dependencies between tasks of different criticality. This dependency cause further challenges for timing analysis it is difficult to upper bound the waiting time for the dependent task.

The challenge here is the predictability of execution of a set of mixed criticality tasks which are scheduled on a given set of resources. This includes consideration of interference, not only due to execution, but also due to criticality mode changes.

1.4 Mixed Criticality Probabilistic Real-Time Systems

We see that mixed criticality as well as probabilistic approaches have the potential to quantify and control the pessimism in the system. For MC, different WCETs can be obtained for a task, in the increasing order of pessimism. This will result in as many criticality levels. Doing so affects the scheduling decisions as the scheduling depends on the WCETs of the tasks. This will extend to many system criticality modes and the scheduling options will depend on these modes. For example, an existing approach is switching to high criticality system mode in a two criticality tasks, implying two WCETs, which enables dropping low criticality tasks.

From probabilistic approach, the quantification of pessimism is done using a probability distribution. The distribution assigns probability to each possible execution time. The scheduling decision does not change and remains as defined by a scheduling algorithm. At the present state of the art, scheduling based on probabilities is not accomplished.

In this thesis, we use this direction Chapter 2 onward which combines MC and pRTSs to model a MC pRTS. In this case, the task executions are described by a probability distribution. On the distribution of the high criticality task, we impose a threshold which corresponds to the lower WCET. The meaning of lower WCET for a task does not change by doing so. What we get in addition is the probability of the task exceeding this threshold, i.e. exceeding lower WCET end entering higher criticality mode. This extends to the probability that the system enters high criticality mode.

Decisions based on probability: Until this point in MC systems we see that the task criticality is defined based on its WCET. We have pWCET instead of WCET and there are probabilities associated. The scheduling decisions do not change the pWCET of the tasks, rather they are based on the WCET. Following this line of thought, the probability that the system enters high criticality can only be quantified.

MC theory in itself is a method to quantify and handle pessimism in real-time systems. By switching to the higher criticality mode, the system drops all non-essential low criticality tasks. The resources which were earlier given to low criticality tasks are now used to provide more time for execution for high criticality tasks. The high criticality tasks are then scheduled using a higher WCET. Applying probabilities to the MC theory enables to quantify the probability of this mode change. Since, resources are taken away from low criticality tasks which drops their execution, system entering high criticality mode is treated as a special case. The probability of this occurring should be quantified. Beyond this, by closely observing the MC system, the probabilities get affected by the scheduling decisions. By minimizing the probability of system entering high criticality, the MC scheduling problem can be optimized. Furthermore, the classical idea that all the low criticality tasks are suspended from execution in the system high criticality mode is also dropped in this thesis. Low criticality tasks are allowed to execute as long as they do not sabotage the execution of the high criticality ones.

The probabilities can not be included in the decision making process until they are affected by the decisions themselves. This statement might sound self contradictory because the decision changes the probability and vice versa. However, the decisions can be enumerated and the affect of all those enumerations on the probabilities can be obtained. This way, an optimal decision can be deduced. We will see how we accomplish this in Chapter 5 and 6.

1.5 Formal Methods

Formal methods are ways to prove a certain property of a system model through rigorous underlying mathematical or logical foundation. A formal method has a framework which allows a system to be modelled. This system model can then be checked for various properties. Through formal methods, the system model checking is assured to be reliable. These are extensively used to ensure safety critical systems like affect of failure of a system component on the whole system, ensure continuity (no deadlock) of the system application, etc.

An example of formal methods is Petri Net. Petri Net models the system using states and triggers between those states. The states contain tokens which pass to other states through triggers. Trigger represents an action, which can be timed or physical, and can change the state of the system. This way Petri Net allows system modelling. Then, the Petri Net is exploded into a state space which contains all the possibilities from the system model. This state space is explored to check various possibilities like reachability, deadlock, etc. Suppose, a Petri Net represents an avionics system with a state representing complete system failure. Through model checking, the state of system failure should never be reached. Petri Net comes under the theory of Automata which models states with transitions between them. Automata is a general definition which describes a sequence of events. At the same time, there are formal methods which are based on deductive logic. For example, if property A of a system is true and another property B of the system is true, a statement for system utilizes those results. Frama-C is an example of such a tool.

Markov Chain (MCh) is also such an example. MCh is a set of states and transitions between them. A set of states and unidirectional transition is a MCh if it respects Markov property. Markov property states that to know the future state of any state, any knowledge of the past states is not required. In a Discrete Time Markov Chain (DTMCh), each transition is labelled with the probability of being chosen. In the case of Continuous Time Markov Chain (CTMCh), the transitions are labelled with the exponential probability distribution parameter. Various properties exist of such a system which is mathematically proven. For example, a MCh which is closed (there is no absorbing or end state), there exists a steady state probability of the system staying in a state. This is the probability that the system is in a state as time tends to infinity. Formal methods like this are based on a mathematical formulation and not state space exploration.

We intend to use such formal methods for our study. We recall that the real-time systems must be ensured safe. Using formal methods, we can conclude this with certainty through model checking of the system model. By doing so, there remains no question of loss of probabilistic information.

We will use such methods for next two chapters. We will also explore the limitation of such

approaches and we will be compelled to find a more suitable solution from the safety point of view. We will develop our own analysis in chapters 5 and 6 which does not refer to classical formal methods.

1.6 The Thesis

The objective of the thesis is to utilize probabilistic approaches to model and analyze real-time system behaviour. This is done in order to exploit the advantage of probabilistic task models. The advantage in this case is the probabilistic quantification of the pessimism that exists in task execution time estimation. This pessimism then propagates through the system during its analysis. Therefore, the global objective of this thesis is, and for the rest of the text will remain, reduction of pessimism in the system in order to improve resource usage efficiency.

The thesis is partitioned into four parts which progressively exhibit the problems encountered and their respective solutions. These problems are the objectives of the thesis. These are as follows:

1. Given a probabilistic real-time system with scheduled task executions which are described with pWCET, what is the probability that the tasks miss their deadline.
2. Given a Mixed Criticality probabilistic real-time system with task executions described with a continuous pWCET, what is the probability that the system enters high criticality mode.
3. Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the optimal schedule when the system enters high criticality mode
4. Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the optimal application oriented schedule when the system enters high criticality mode.

The answer to each problem is also the motivation for next problem. We interpret and answer the problems as follows.

1. Given a probabilistic real-time system with scheduled task executions which are described with pWCET, what is the probability that the tasks miss their deadline?

This question is the formal reformulation of the title of the thesis. We are given with a real-time system with a set of periodic tasks on a uniprocessor machine. The task executions are described by a continuous pWCET. We consider continuous pWCET because the output of the timing analysis from Measurement Based Probabilistic Timing Analysis (MBPTA) are continuous distributions. Continuous distributions cannot be converted to discrete distributions because of the difference in the interpretation of the two. Moreover, the continuous and discrete must be safely handled with worst case upper bounds during performing analysis. We perform analysis on Earliest Deadline First as well as Fixed Priority.

In order to analyse the system we observe the following. The system needs to be represented in a discrete form using states. That is because the system occurences are discrete in nature, like preemption,

release of task, etc. If we see them as continuous properties, we will eventually have to discretize them to make them computationally useful. This will lead us to complexity issues affecting the scalability of the solution. The states must have transitions between them because the system acts by switching from one state to another, e.g. execution followed by preemption. Moreover, a task cannot execute again after deadline miss implying the transitions have to be unidirected. Thus we need a set of states and unidirected transitions to model a pRTS. The execution is probabilistic, implying that there might be some transitions relating to task execution which are probabilistic. This means we need a set of states and transitions which can cope with probabilistic transitions. Moreover, we need a formal method to do so because only then the underlying mathematical foundation can be used to ensure correctness of the analysis. This relates to safety of the method used. Through formal methods, there will not be question of losing any probabilistic information because the mathematics is formulated as such.

At this point it is important to observe the difference between probabilistically distributed system and a sequential system. Discrete event systems like Petri Net and Automata are sequential systems. They begin with an initialization, like tokens in a node of Petri Net. Then they evolve in a precise manner described by the transitions the system can take. This precise evolution results in a state space in which various properties are verified, like reachability, liveness, etc. Models of probabilistically distributed systems on the other hand do not evolve sequentially. The probability is distributed throughout its states as soon as it is constructed. This means as long as the system model is complete, the system has a certain probability of being in any state. This probability exists any time the system is executed, like the probability of obtaining number 6 exists each time the dice is rolled. The value of probability is independent of the event occurrence. To illustrate the difference, a state in the sequential system may or may not be reachable but a state in probabilistic system is always reachable with a certain probability. In addition, the state reachability in sequential system is a question of when and whether will it be reached, but in probabilistcally distributed system, state is always reachable at any time.

On the other hand, we need observations at deterministic times from a probabilistically distributed model in order to be conclusive of the scheduling decisions. For example, the time instant of deadline miss is a deterministic value and the probabilistic model has to be checked for task execution against this instant. Similar is the case for preemption because time of preemption for periodic tasks is deterministically known. Thus, we need a formal method which can cope with this requirement.

From this observation, we can clarify that in a probabilistic system there cannot be states representing deadline miss. This is because reaching the state deadline miss will always be possible with a probability. This is not practically possible because deadline miss may occur at a specific time or may not occur at all. Representing this would have been possible in the sequential system. Also, a continuous distribution applied to a sequential system will require a discretization of the distribution. The fine-ness of the discretization will affect the complexity of the model because we would have as many number of states. Using a probabilistically distributed model for a continuous distribution will not have this problem.

To summarize, we need a probabilistically and continuously distributed non-deterministic modelling formal method. We choose Continuous Time Markov Chain (CTMCh). Markov Chain is a set of states and transitions with the Markov Property. Markov Property states that to determine the future state, the system does not need to remember the past states. The system model can be prepared in the manner that this property is respected. The transitions in Markov Chain are labelled with the probability of it being

chosen. In the case of CTMCh, the transitions are labelled with exponential distribution parameter which describes the probabilistic rate of the transition. The CTMCh can be checked using a formal model checker, like PRISM Model Checker in [KNP11].

We will build one CTMCh model for each instance of a task. The task set is scheduled and we perform schedulability analysis on it. It has states representing executions after release and preemptions and a state that it has finished execution. This model can be checked for deadline using a formal property, 'the probability that finish state is reached by the deadline'. Similar properties can be checked for preemption. We also propose a solution in case the input probability distributions are not exponential which is safe by pessimism. This solution is presented in Chapter 3.

2. Given a Mixed Criticality probabilistic real-time system with task executions described with a continuous pWCET, what is the probability that the system enters high criticality mode?

At this point we incorporate mixed criticality into the pRTS. The input is kept the same that the task executions are described by a continuous distribution. We add another specification that some tasks in the input are high criticality tasks and the rest are low criticality. The task enters high criticality if it crosses an execution time threshold which is less than the WCET. This makes the system a MC pRTS on a uniprocessor machine. The objective here to obtain the probability that the system enters high criticality.

Here, the system criticality is defined on the basis of task criticality. That is, if a task enters high criticality at any point of time, the system switches to high criticality mode. The event of a task entering high criticality is a discrete event even though it is extracted from a continuous distribution. Thus we need a discrete system modelling method which can also incorporate the probability. In addition, it should remain a formal method to ensure safety.

In this direction we choose Discrete Time Markov Chain (DTMCh). DTMCh is Markov Chain in which the transition between the states is labelled by the probability of being taken. The number of outgoing transitions from a state are limited and the total probability of the outgoing transitions must be equal to one. The discrete probability of a task entering high criticality from the pWCET can be directly mapped to these transitions. Such states and transitions represent the set of tasks executing as the schedule. On the resulting model, the formal properties can be checked, like 'the probability that at least one task enters high criticality', etc. We define a set of such properties to evaluate the probability that the system enters high criticality in Chapter 4.

3. Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the optimal schedule when the system enters high criticality mode?

Until this point we have been performing schedulability analysis on the task set which are scheduled according to a given algorithm. This method works as long as all the low criticality tasks are dropped as soon as the system enters high criticality. This is because the schedulability is ensured for those high criticality tasks. This method does not take into consideration the possibility that; first, one task entering high criticality does not mean all the tasks will surely enter high criticality; second, if there is room for execution of low criticality tasks even in the system high criticality mode. This imposes a pessimism which is due to scheduling decisions and not due to task execution. With this motivation,

we decide to obtain a schedule for mixed criticality pRTS.

The schedule obtained here aims at maximizing resource usage. We fine grain the classical mixed criticality approach by decomposing the notion itself to the tasks and not to the system. We remove the assumption that if one task enters high criticality, the system enters high criticality mode. By doing so, we have removed the pessimistic action of dropping all the low criticality tasks. The notion of criticality is decomposed to the task execution level and the notion of system wide mode switch is removed.

We need a model that can capture this fine grained approach for mixed criticality system. In order to prove global optimality, we need to explore all the possibilities of schedule. The modelling methodology should be discrete because the events of task entering criticality or missing deadlines are discrete events. We naturally tend towards a graph based exploration. There are states which represent the task execution and unidirected transitions between the states. The exploration across all the possible schedules is performed. This exploration removes all the possibilities which are not feasible (deadline miss) or non-optimal and keeps the optimal and complete schedules. This means, for a task entering high criticality, the schedule keeps the room for execution of all the high criticality tasks in the high criticality mode as well as maximum possible number of low criticality tasks. From this exploration, we obtain a schedule which is optimal in resource usage and is prepared for any task entering high criticality. We are able to precisely see when a task can enter high criticality and plan ahead if it does enter high criticality. This solution is presented in Chapter 5.

4. Given a Mixed Criticality probabilistic real-time system with task executions described with pWCRT, what is the optimal application oriented schedule when the system enters high criticality mode?

We have seen so far that the probabilities obtained from pWCET are fixed. These probabilities do not change because of the scheduling, rather the scheduling is based on these probabilities. Through this, we can only quantify of something occuring, like the system entering high criticality. The probability never tells us when the event will occur. In the case of mixed criticality systems, we cannot know exactly when the task enters high criticality.

We also study and interpret the meaning of criticality by making an observation. The mixed criticality theory is a result of uncertainity in task execution. That is, the uncertainty exists while the task is executing in the applied system. Effectively, the criticality is decided while the task executes, i.e. online. Online task execution is described by probabilistic Worst Case Response Time (pWCRT). Therefore, the criticality should be defined using the pWCRT and not pWCET. This is what we mean by an application oriented mixed criticality schedule because mode change depends on the application When we do so, the probability of a task entering high criticality depends on the schedule and so does probability of system entering high criticality. With this, the probability comes into play to decide the schedule of the system.

Now we explore the case to obtain an optimal schedule when the mixed criticality is defined using pWCRT. That is, a schedule for which the probability that a task enters high criticality is minimum. Same as before, we use graph based exploration to obtain a global optimal. The response times are obtained for each task in each exploration and an optimal one is selected.

Now that we reduce the probability, what is the optimal schedule if the system actually does enter high criticality. The probability that system enters high criticality exists each time the tasks execute. In order to be certain of the schedulability when the system enters high criticality, the schedule must be upper bounded. In addition, the schedule must be optimal in system high criticality mode. In this, we will not talk in terms of probability and only use WCET in order to consider worst cases. WCET is the information given in the pWCET. The graph model is extended to obtain an optimal schedule in this worst case. Same as before, the schedule is obtained which is prepared with the appropriate action for any task to enter higher criticality which is resource efficient.

We also perform a first step towards understanding dependence in the case of mixed criticality pRTS. These dependencies between tasks in their timing behaviour can arise due to a shared resource or a shared bus, etc. A task entering high criticality necessarily causes another task to enter high criticality; thus, the dependence affects the decisions taken about the task and system criticality modes. Thus, a task can enter high criticality because of itself as well as another task, each case with its own probability of occurrence. The dependence that we consider is not the same notion as dependence between probability distributions. In our knowledge, our approach is a first step towards understanding dependence in MC probabilistic environment. Our work intends to include dependence between tasks and couple those with scheduling decisions. This is presented in Chapter 6. Chapter 7 concludes the thesis and presents the overall perspectives.

1.7 State Of The Art

Beginning with the Apollo Guidance Computer [Int09], we are surrounded by real-time systems which function with real-time constraints. A good early overview of the complexity of the systems in then future was given by [Moo65]. As a rough measure of increasing complexity, the Moore's Law follows that every two years, the number of transistors in an integrated circuit doubles. In 2015, Intel said in the article 'Intel Rechisels the Tablet on Moore's Law' in Wall Street Journal, "The last two technology transitions have signaled that our cadence today is closer to two and a half years than two". The complexity arises from the ever-increasing demand of functionality and performance. The technical challenges that exist with growing real-time systems has been reviewed by [Sta88]. One should go through [Sta88] to gain insights about some common misconceptions for real-time systems. For example, having faster computers will solve real-time needs is a misconception; the wrong assumption that real-time systems will function in application as in a static environment, etc.

A comprehensive text on the fundamentals of real-time computing systems is given in [But11]. The book ranges from basic concepts of real-time systems which extend to periodic and aperiodic scheduling. The notions of fixed priority servers and dynamic priority servers have deserved a chapter each which describe various scheduling policies.

Most of the modern research in this field is performed to cope with the challenges mentioned in [Sta88], like scheduling, verification, etc. The ever growing research is always driven by to one of the *Murhpy's Laws*, "Anything that can go wrong will go wrong". Scheduling is a major research area because scheduling is directly responsible for managing resource usage. Under the guidance of Andrew van Tilborg, the Real Time Systems Initiative by the United States Office of Naval Research contains important early results about scheduling theory.

Timing analysis itself is an area of extensive research. Timing analysis results are the input for scheduling theories. [EE07] is a good discussion on how to obtain WCET. [Wil+08] elaborates on the meaning of WCET and the challenges associated. It discusses the need for timing analysis, the various factors affecting the WCET like software behaviour, memories, etc. and various methods to obtain WCET like measurement based analyses, static analyses, etc. In the probabilistic domain, [BCP02] presents such a WCET analysis for pRTS. They do so by defining an Execution Time Profile (ETP) as the time of a piece of code. This is followed by classical probabilistic combinations of the ETPs through convolution. They extend their work towards biased convolution in order to include dependencies. It is an overview of the methods like static analysis and measurement based. Some commercial WCET analysis tools are also discussed.

Plethora of research exists in the field of real-time analysis. [Vic01] presents a static analysis of a system dependent on time. They use a Time Petri Net to represent system states and timed triggers, and reachability is verified. This followed by changing from one model to another as demanded by the safety and operational constraints. Work by [VMF03] exhibits a use of real-time systems as a control system The control system must perform with real-time constraints in order to control a physical system. [VMF03] provide a model for self-triggered task by adjusting the task period according to system dynamics. [BLHS91] present a general mathematical modelling of scheduling problems. They classify the scheduling problems into criteria based on completion time, due dates or inventory cost

and utilization. The equations for optimization are formalized.

The tasks in the system are scheduled to allocate resources for their execution. In order to ensure an allocation methodology ensures complete and safe execution of system, schedulability analysis is performed. [LL73] is considered as the foundational work on Fixed Priority (FP) Scheduling. An analysis of Earliest Deadline First (EDF) algorithm is also shown in [LL73]. [ZB09] present an overview and foundation of schedulability analysis of real-time systems with EDF scheduling. [SB94] extend EDF scheduling algorithm to four new directions. Dynamic Priority Exchange server with its improved version, Total Bandwidth Server and Earliest Deadline Late methods are presented. They are aimed at deadline with soft, aperiodic and hard periodic tasks.

Formal methods are also used to perform real-time system analysis. [LR09] is a work on formal verification of real-time systems with preemptive scheduling. It proposes a method for verifying schedulability using Time Petri Net. [BD91] contains a framework for modelling a time dependent system through Time Petri Net. It being a Petri Net, usual properties are proved like boundedness, reachability, etc. [G. 04] extend Petri Net to Preemptive Time Petri Net to model task executions in which timeliness and reachability are verified. [Oli04] is a work which develops Time Petri Nets with inhibitor arcs. Inhibitor arcs on a trigger prevent that trigger if the inhibitor arc comes from a state with a token. [Kim05] present a test of real-time embedded control software using the UPPAAL-TRON tool. They use a Timed Automata to model a temperature controller.

Formal models of real-time systems based in graph theory also exists in the literature. [Bar98] presents a graph based representation of recurring real-time tasks. Using graph properties, a Demand Bound Function (DBF) is derived. DBF denotes the maximum cumulative execution requirements by the tasks. [MKT04] present a work using workload representation, similar to DBF. They obtain bounds on this representation using task WCETs and BCETs. This is followed by obtaining backlogs and ensuring that they lie withing those bounds. [Sti+11] is work which used Digraph to model real-time tasks. They model jobs of the tasks as graph nodes. Then the demand bound function for the same is obtained using classical graph properties.

In direction of probabilistic real-time system, [DBG17] clearly defines and then explains the meaning of pWCET distributions. It briefly discusses how they are obtained and their uses. Work by [CG13] discuss the independence of the pWCET of the tasks. It discusses the meaning of pWCET as the upper bound of all possible execution times. It encourages the readers to see the pWCET as a property which does not imply the task independence from other tasks. That is, pWCET should contain all the possible delays except the ones due to scheduling any dependence produced thereafter.

Research is being carried out to exploit the advantages of probabilistic approaches without jeopardizing the safety of the system. [Dia+02] present a strong approach towards analysis of pRTS. First, they establish a method to construct response time from given discrete pWCETs of the tasks. They call it ‘convolution and shrinking’. This is followed by proving the accumulation of backlog, that is execution delay, for the periodic tasks is a Markov Chain. Then they proceed to obtain the steady state probability function of backlog. This shows that if such a stability is proven, the probabilistic system will not accumulate an increasing backlog. Another work,[Sto02], gives a foundation of probabilistic automata which is a potential tool for pRTS analysis. It also classifies various probabilistic models without, with partial and with full non-determinism. At the same time, [CT06] present a framework for

to obtain response time of the tasks. Peculiar to their work, the probability exists in the inter-arrival time of the tasks instead of their execution time. This is an example of how probabilities can represent the system apart from the pWCET.

Works exist which exploit the benefits the probabilities in pRTS research. [BBB03] consider various aspects of probabilistic systems. They provide probabilistic guarantees for fault tolerant system by obtaining the Mean Time Between transient Faults (MTBF). This is usually given as the reciprocal of the exponential parameter which is a special case of Poisson distribution. They represent non-periodic arrival patterns by simply obtaining a response time. This is followed by a discussion of Copulas to represent execution time. Work by [SG16] provide a representation for probabilistic resource usage. They extend the DBF to the probabilistic variant called probabilistic C-Space. Such a representation allows to clearly define feasibility and confidences of a probabilistic system. [AMP12] develop a theorem to provide a bound to the probability for finishing time of tasks. An important definition of safety in pRTS is given in [Dia+04]. They elaborate on the notion of pessimism in the probabilistic systems. That is, they define how a distribution can be 'worse than' another distribution by providing pessimistic probabilities. This pessimism is desired because a system can be over-provisioned but not once it should be under-provisioned.

[Lu+12] present an approach which performs measurements directly on the system. The system is viewed as a black box and no knowledge of WCET values are required. They use EVT and a number of measurements to provide a probabilistic picture of the system. [DD99] is a work out of real-time domain but useful for probabilistic analysis. The work is on estimation of the index of Extreme Value Theorem (EVT) which is useful for pWCET estimation. [HHM09] is a work which used EVT. However, they do so to obtain WCET as an upper bound. This is also based on number of observations and obtaining an upper bound curve. Further work using EVT, [Abe+14] studies the use of Measurement Based Probabilistic Timing Analysis using EVT. They study the requirements and affecting scenarios for MBPTA analysis. Various factors are considered to gain more confidence on the approach. [CG+12] is also an example of using EVT based MBPTA approach to obtain pWCET. [MEP04] derive an analysis of pRTS by first defining priority monotonicity intervals. These are time partitions based on scheduling of tasks. They assume continuous pWCET distributions. The probability distributions for task execution in the schedule are then computed with respect to those intervals through convolution.

Considering continuous distribution is unique to the above work by [MEP04]. Another work by [Buc+10] develop Oris tool which is able to model systems using Time Petri Nets (TPN), preemptive TPNs, stochastic preemptive TPNs and stochastic TPNs. Oris tool is module supporting timed automation and subsequent model checking. They present an analysis of tasks with pWCETs defined using Erland continuous distribution in [Car+14]. They present a stochastic TPN in which the timed transition is probabilistic and follows the given pWCET distribution. They produce transient probabilities of a certain task execution at any given time.

Work by [CGV09] use Stochastic Time Petri Nets to represent task execution. They construct a calculus for the probability in the enumeration of the Petri Net. [Cia94] provide a characterization of the Stochastic Petri Net. They elaborate on various Petri Nets, namely, Generalized Stochastic Petri Net, Deterministic Petri Net, semi-Markovian stochastic Petri Net and Generalized Times Petri Net. They discuss their hierarchy and decomposition into Discrete or Continuous Markov Chain. A

probabilistic model checker we use is named PRISM Model Checker. Using PRISM Model checker, [KNP11] elaborates a real-time system application which uses the model checker for a probabilistic Timed Automata. Various properties can be formally checked using the tool. We will use this tool in one of our research studies in this thesis. [Hor+12] also develop a formal transient analysis of stochastic state classes like stochastic Petri Net. Work by [CKT94] develops a Markov regenerate stochastic Petri Net for networks. Here, a stochastic trigger in the Petri Net maps the probabilistic distributions. The model is converted into reachability problem with probability. [Caz+11] elaborates a project named PROARTIS. It is a study to obtain pWCET distributions. They elaborate on static analysis as well as measurement based approaches and develop the mathematical foundation.

Relevant work to probabilistic scheduling also exists in network analysis. [SM01] applies an activity network with a stochastic extension similar to Petri Net. Work by [A. 95] applied Markov Regenerative Stochasitc Petri Net for network analysis and obtain transient probabilities.

In last two decade, plethora of research is being performed in mixed criticality (MC) systems. [Bar09] show that the MC scheduling problem is NP-hard. It is evident that any effort to obtain a schedule, which also looks for optimal solutions in a probabilistic environment, will be complex. [BD19] is a review of mixed-criticality systems. It presents a general definition of a MC system using tasks components and each component having a level of criticality. WCET is used to represent task criticality. It overviews single and multiprocessor scheduling of real-time systems. It also presents utilization bounds, speedup factors and formal language issues relevant to MC analysis. [BG15] is short discussion on the expressiveness between the Vestal [Ves07] and the Burns [Bur15] model MC system.

[Bak+09] present a Simplex Architecture in which 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. This work is similar in idea to MC approach that we use in this thesis.

[TF13] present a MC scheduling approach which creates scheduling table and obtains a schedule from a tree. They also present a method to backtrack within the tree to remove any infeasible schedules. [AD17] perform Fixed Priority MC schedulability analysis of tasks with pWCET. This analysis is based on convolution of the pWCETs to compute the response times.

Another work by [TBW92] derives a criticality mode change protocol. They form their work around execution window of a task as given by its response time. In effect, the mode change is based on the amount of idle processor time.

[GSY15] present a schedulability analysis of mixed criticality pRTS. A permitted failure probability is assumed. They present an LLFL-clustering algorithm in which separate EDF scheduling is verified for just the high criticality and another for complete system. This leads to the notion of strongly or weakly probabilistic schedulable.

Works like [Bar+14] apply MC scheduling problem for multiprocessors/ They present an algorithm based fpEDF algorithm. fpEDF is a global EDF-based algorithm in non MC environment. The new algorithm acts by dropping the low criticality tasks from execution when required. Another work [BB13] present fixed priority MC scheduling which also lead to dropping of lower criticality tasks. [BEG15] present a MC-Fluid scheduling for MC systems. It derives execution rate of the tasks and obtains bounds for execution in high criticality mode by dropping low criticality tasks.

Works of [Liu+16]; [Bar+11] present a method called EDF-VD (Virtual Deadline) scheduling algorithm for mixed criticality systems. This work in the deterministic environment, an important work nevertheless. The approach is to establish a virtual deadline to the high criticality tasks and schedule them according to EDF algorithm. Changing the virtual deadline changes the priority of the high criticality tasks.

[BBD11] present approaches for Static Mixed Criticality (SMC) and Adaptive Mixed Criticality (AMC) analyses. In SMC, all the jobs can execute up to their representative execution time but are prevented from executing further. With AMC, the jobs are dropped or not depending the system criticality mode. SMC assumes a limited execution support to the tasks. AMC provides an a flexible schedulability guarantees with requiring additional execution support to the tasks. The present the case with FP scheduling algorithm.

[Max+17] present a probabilistic analysis of preemptive fixed priority mixed criticality scheduling. The probabilities of exceeding an execution time are determined from the pWCET distribution. A translation of deterministic response time to probabilistic response time is performed. The schedulability is performed under Static Mixed Criticality (SMC) and Adaptive Mixed Criticality (AMC) schemes.

A doctoral thesis by [MBP17] present an Directed Acyclic Graph based approach for MC pRTS systems. They proceed to obtain schedulability analysis for MC tasks in different modes on multi-core architectures. [AG16] utilize a Markov Decision Process to model MC task executions. They obtain a feasible schedule where the probability constraints are satisfied. This is based on the trajectories given from the MDP which interpret as the schedules.

Some novel ideas in MC scheduling research include [Ab+16]. They present a MC model in which they consider two additional criticality modes, transient fault and execution time overrun. These two modes aim at a more expressive MC environment which in turn improve utilization. Another work [Bhu+19] present a MC scheduling which varies processor speed through dynamic voltage and frequency scaling. [Guo+17] discuss sustainability in the MC scheduling. Sustainability is the requirement that schedulability tests remain valid if the runtime behavior of the system in better.

This thesis extends the existing state of the art in many directions. The general direction of these extensions is to increase the efficiency of resource usage by quantifying and utilizing the existing pessimism. The specific points of these directions are given as follows:

- The existing literature on obtaining the pWCET is used and it is assumed given. We question the form in which this information is present in terms of usability, like using continuous distributions. We also obtain perspectives on safety of representation or transformation of pWCET from discrete to continuous and vice versa.

- The complexity of using various pWCETs is obtained and discussed. In the cases where complexity is high, we obtain perspectives on where we gain in terms of system representation and application.
- We obtain safe and less pessimistic ways of obtaining task response times which efficiently uses classical convolution operation.
- We use formal methods for a good part of this thesis. We do so with the aim of safety of system representation. We are assured that the results obtained thereafter are safe and no information is lost in the process.
- With respect to existing literature on MC, we learn the extent of the usability of existing scheduling methods in MC pRTS.
- We follow this line to remove some of the classical MC assumptions and approaches which go against the idea of removing pessimism and increasing resource efficiency.
- While studying these directions, the safety of the MC system representation is not sabotaged and taken care of. That is, the schedulability of high criticality tasks is assured, even in the worst case scenario.
- We obtain an application oriented MC definition which uses response time instead of execution time to define criticality. Ours is the first work to undertake such an approach.
- We take a first step towards defining and quantifying inter-task execution dependence apart from scheduling in a MC environment.

CHAPTER 2

Fundamentals and Notations

Works of art make rules; rules do not make works of art.

Claude Debussy

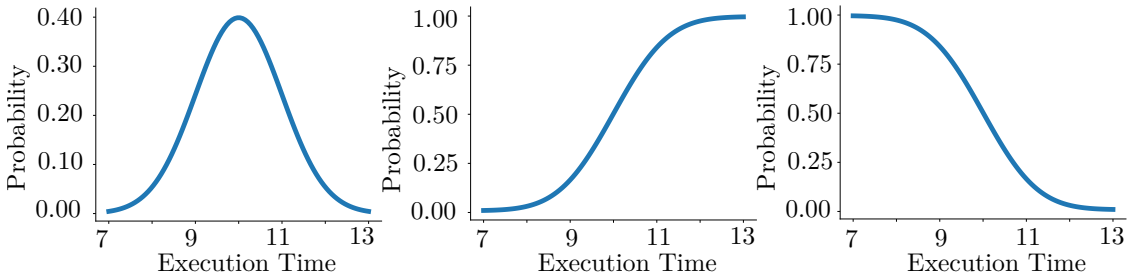
In this chapter we define the notations and revise some fundamentals related to probability theory, real-time systems and mixed criticality theory. These are general notations throughout this thesis. If required, any additions to these will be done in context within the specific chapter.

Dans ce chapitre, nous définissons les notations et révisons certains principes fondamentaux liés à la théorie des probabilités, systèmes en temps réel et théorie de la criticité mixté. Ce sont des notations générales tout au long de cette thèse. Si nécessaire, tout ajout sera fait dans le contexte du chapitre spécifique.

2.1 Probability

For any two events A and B the probability law holds $Pr(A \bigcup B) = Pr(A) + Pr(B) - Pr(A \bigcap B)$, where $Pr()$ denotes the probability of event, $\bigcup$ is logical OR and $\bigcap$ is logical AND. The probability that A occurs given B has occured is given by the conditional probability, $Pr(A/B) = Pr(A \bigcap B)/P(B)$. If the events A and B are independent, $Pr(A/B) = Pr(A)$ implies $Pr(A \bigcap B) = Pr(A)Pr(B)$. It should be noted that the events A and B come from a common sample space, for example from all the possible outcomes of roll of dice. More attention is needed to this notion of common or uncommon sample space when applying probabilistic models in real-time systems.

The possible outcomes of a trial are represented by a random variable $\mathcal{C}$ with corresponding probability functions. For example, a discrete random variable representing the outcome of roll of a dice can take values from $\{1,2,3,4,5,6\}$. For a discrete random variable $\mathcal{C}$ representing the task execution time, the Probability Mass Function (PMF) $f(x)$, or simply f, gives the probability that $\mathcal{C}$ takes a certain value x , $f \stackrel{def}{=} \mathcal{P}(\mathcal{C} = x)$; $\Sigma_{-\infty}^{\infty} f(x) = 1$. Alternative representations to $\mathcal{C}$ are the Cumulative Distribution Function (CDF) $F \stackrel{def}{=} \Sigma f$, and the Inverse Cumulative Distribution Function (ICDF) $\overline{F} \stackrel{def}{=} 1 - F$. The random variable $\mathcal{C}$ can be continuous or discrete.



(a) PDF (b) CDF (c) ICDF

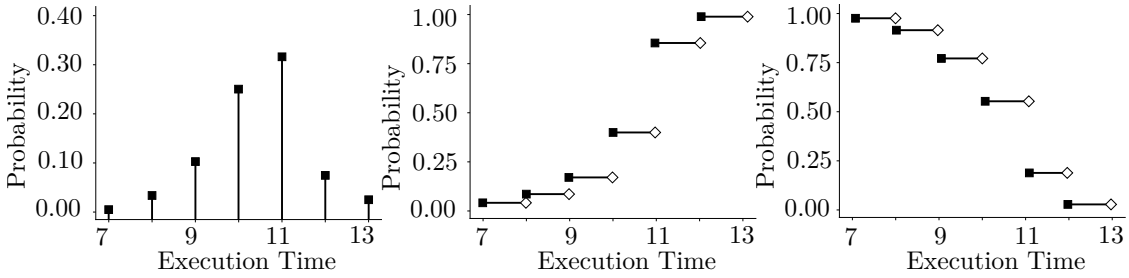
Figure 2.1: A continuous Gaussian distribution in PDF, CDF and CCDF forms.

In case of exponential distributions, $f = \lambda e^{-\lambda x}$ where λ is the rate parameter which describes the shape of the exponential distribution. Exponential distributions are continuous distributions supported on the interval $(0, +\infty]$; they are referred to as $EXP(\lambda)$.

If $\mathcal{C}$ is a continous random variable, the probability density function (PDF) $f(x)$, or simply f, gives the probability that $\mathcal{C}$ takes the value between a and b; $Pr(a \leq \mathcal{C} \leq b = \int_a^b f(x)dx$. The Cumulative Distribution Function (CDF) F gives the cumulative probability for $\mathcal{C} \leq x$. F is the integration of probability density function f: $F = \int_0^x f(x)dx$;

Figure 2.1 illustrates an example of a continuous random variable represented respectively with the PDF, the CDF and the ICDF.

Similarly, Figure 2.2 shows a distribution in discrete form in PMF, CDF and ICDF forms.



(a) PDF (b) CDF (c) ICDF

Figure 2.2: Example of PDF, discrete CDF and discrete CCDF representations of a certain distribution.

For exponential distribution of rate λ, $F(x) = 1 - e^{-\lambda x}$. The Inverse Cumulative Distribution Function (ICDF) $\overline{F}(x)$ gives the exceeding threshold probability as the probability that $\mathcal{C} > x$. $\overline{F}(x)$ is the one minus integration of probability density function $f(x)$: $\overline{F} = 1 - \int_0^x f(x)dx$, and for exponential distribution of rate λ, $\overline{F} = e^{-\lambda x}$.

The convolution of two PDFs f and g, denoted by $\otimes$, refers to the summation of the random variables they represent and is given as: $f \otimes g(z) = \int_{-\infty}^{\infty} f(z)g(t-z)dz$. The convolution of more than two PDFs is represented as $\underset{i}{\otimes} C_i$. In case of discrete distributions, the convolution of two PMFs is given

as $f \otimes g(z) = \sum_{z=-\infty}^{\infty} f(z)g(t-z)$.

2.2 Real-Time Systems

A real-time systems consists of a set of tasks and certain resources:

1. **Tasks**: The execution processes which produce certain result. Tasks are represented as τ defined using the following properties
   (a) *Worst Case Execution Time*: The maximum amount of time the task can take to finish execution in isolation.
   (b) *Period*: The time after which the task executes again.
   (c) *Deadline*: The maximum amount of time the task is allowed to take to finish execution. The deadline is always less than or equal to the period.
2. **Resources**: The resources include the processors, memory, buses which tasks require to complete their operation.

The task arrives at an instant called the *arrival time*. A task during execution can get *preempted* if there is another tasking which arrives later but must be executed immediately. The task has a *response time* which is the amount of time the task takes to finish execution after its release. The response time of the task contains all the possible delays it can have to its execution plus the execution time itself. These properties are shown in the Figure 2.3. The figure shows the pWCET as the ICDF of a task which upper bounds various possible task execution PDFs.

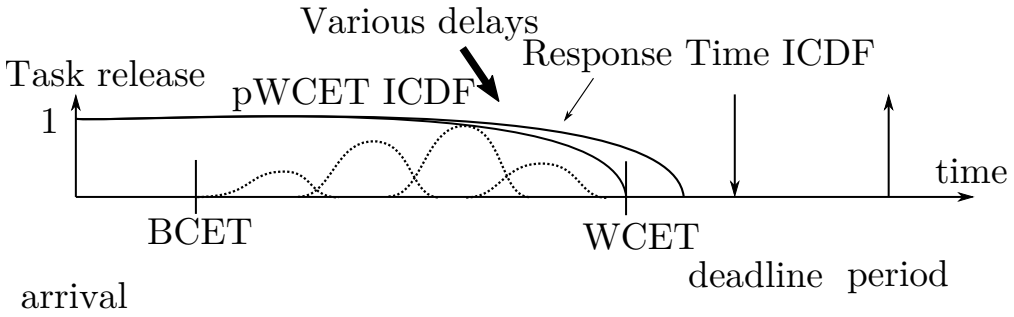


Figure 2.3: Task probabilistic execution shown as a pWCET ICDF with BCET and WCET.

In this thesis, we will focus on tasks and the resource given is a uniprocessor machine.

In our case, the execution time is described using probabilistic worst case execution time (pWCET). pWCET is the the worst case probability distribution which upper bounds all possible execution times of a task.

The task pWCET is a discrete random variable C whose PMF f represents the probability that the task takes a certain WCET. In its representation with CDF, F is the cumulative probability that the

task respects certain WCET while executing; in the ICDF representation, $\overline{F}(x)$ is the probability that the task overcome certain WCET. Figure 2.4 shows an example of a pWCET PMF. The deterministic WCET C from $\mathcal{C}$ is the maximum value of $\mathcal{C}$; for it $F(C) = 1$, and $\overline{F}(C) = 0$.

We assume a set of $m \in \mathbb{N}$, periodic tasks running in a system which is represented with the set $\Gamma = \{\tau_1, \tau_2, \ldots \tau_m\}$, $\mathbb{N}$ is set of natural numbers. The parameters of each task τ are identified by the tuple $(\mathcal{C}, T, D)$, where $\mathcal{C}$ is the pWCET with f as the PMF, T the period, D the relative deadline.

The pWCETs are assumed to be independent [CG13]; this is because the pWCET represents the worst case execution scenario of the task. Any execution delays which are caused apart from the scheduling or task dependence from another task execution must already be included in the pWCET distribution. Figure 2.4 shows an example of a pWCET PMF. The maximum value of the pWCET is the deterministic WCET and lower value is the BCET. This way, pWCET is upper and lower bounded.

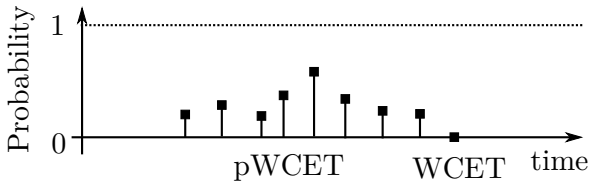


Figure 2.4: pWCET as a PMF distribution.

An instance of a task τ is called a *job* represented as J. The arrival time of the job J is a and the absolute deadline is d. We assume that the arrival of the first job of each task is always at time zero. The task set is scheduled periodically on a uniprocessor machine in which execution of jobs are suspended at their respective deadlines. The priority of the job J is p with 0 being the highest priority.

All times in capital letters, i.e. T, D, A, are time instances of the task; lower case letters, i.e. t, d, a, are absolute time instants with zero being the arrival of the first job in the hyperperiod.

We analyze the jobs in the hyperperiod because the schedule repeats each hyperperiod since the jobs are suspended at their respective deadlines. The *hyperperiod* H of the system is the minimum amount of time necessary for the 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 $(0, H)$:

$$\Lambda \stackrel{def}{=} \{J \text{ s.t. } 0 \leq a < H\}.$$

where a is the arrival time of job J.

In the hyperperiod there are n jobs. The real-time application is be represented by the set $\Lambda = \{J\}$.

2.3 Mixed Criticality Systems

Mixed Criticality systems consist of tasks of the set Γ with various levels of criticalities or importance. Classically, the execution time (WCET, or pWCET) is used to represent task criticality. In contrast,

we use response time to represent the task criticality. To recall, response time is the time a task finishes execution after its release. It is the sum of task execution time and all the possible delays to its execution.

Let l be the threshold on the response time for task criticality. If the task execution exceeds this threshold after its release, it is said to enter high criticality mode. The variables T and D are deterministic single-valued parameters, $D \leq T$ (constrained deadline). The criticality level of the job J is defined as L. The job inherits the criticality level of the task to which it belongs.

We consider pWCET described with discrete distributions. We also consider two level criticality case, HI and LO, with HI having higher importance than LO. The high criticality tasks can execute in HI or LO mode, the low criticality tasks execute only in LO mode. After its release, task executes in LO criticality mode until the execution reaches the response time threshold l. A task execution exceeding this threshold is said to execute in the high criticality mode. For tasks with $L = \text{HI}$, $0 < l < D$ and for tasks with $L = \text{LO}$, $l = D$. Evidently, l is a deterministic single-valued parameter. The task pWCET independence property allows to apply the task pWCET to each job of the same task. Because the job worst-case execution time is probabilistic, the job response time is also probabilistic. Further notations of jobs are defined in the specific chapter of their usage.

The probabilistic distribution representing execution of a job by including all the possible delays in execution is a probabilistic worst case response time. It accounts for the worst case scenarios for the execution of the task that might delay its execution. It is defined as follows.

Definition 1. *Probabilistic Worst Case Response Time pWCRT of a job J, is a random variable $\mathcal{R}$ with PMF $f_{\mathcal{R}}$ which gives the worst case probability that J will take certain random time $\mathcal{R}$, to end execution after its release.*

The response time CDF is $F_{\mathcal{R}} = \Sigma f_{\mathcal{R}}$, and the ICDF is $\overline{F}_{\mathcal{R}} = 1 - F_{\mathcal{R}}$. A certain allowed maximum probability of deadline miss for any job $\mathcal{P}_{dm}^{max}$ is given. A job J with deadline d is said to have missed its deadline if $1 - \Sigma_{x=0}^{d} f_{\mathcal{R}}(x) > \mathcal{P}_{dm}^{max}$.

Same as in the case of pWCET where the upper bound is the WCET, the upper bound of the pWCRT $f_{\mathcal{R}}$ is the deterministic Worst Case Response Time $WCRT$.

The jobs entering high criticality mode is defined using its response time as follows.

Definition 2. *A job J is said to have entered* HI *criticality if its Worst Case Response Time exceeds a threshold l, $\mathcal{R} > l$.*

There are n^{HI} high criticality jobs and n^{LO} low criticality jobs. The set of all high criticality jobs is represented as Λ^{HI} and that of low criticality jobs is Λ^{LO}. It follows that $\Lambda = \Lambda^{\text{HI}} \cup \Lambda^{\text{LO}}$, and $n = n^{\text{LO}} + n^{\text{HI}}$.

In the determinisic case, the response time of a job is the time it takes to end execution after its release. In addition to the execution time, response time also contains the duration in which it has to wait for a previous job or a preempting job to finish execution. Similarly, the pWCRT is the worst case probability distribution representing the time end of job execution after its release. It contains the pWCET as well as the probabilistic waiting times due to various delays in execution.

CHAPTER 3

Continuous Time Markov Chain Schedulability Analysis

The first step to becoming is to will it.

Mother Teresa

In this Chapter we present our first step towards the thesis subject. The thesis subject is formally put as follows.

Given a probabilistic real-time system with task executions described with continuous pWCET, what is the probability that the tasks miss their deadline?

We begin the contributions in this thesis by answering the question posed above which directly relates to the subject of the thesis. We are given with a task set in which task executions are described by a continuous probability distribution. The objective is to perform a schedulability analysis on the tasks execution to obtain the probability of their deadline miss. From the first look, we already see that scheduling such tasks involve a certain play of probabilities within the system which propagates and affects the system functionality probabilistically. From our discussion in the introduction, now we formally dive deeper into this idea.

We obtain a schedulability analysis model using Markov Chain, in particular, Continuous Time Markov chain (CTMCh). We model the system task executions using the CTMCh. Continuous Time Markov Chain is a set of states with unidirectional transitions between them. The transitions are labelled with the rate of the exponential distribution which corresponds to the probability of that transition as well as the probabilistic time spent in the transition. This allows a direct mapping of continuous pWCET to the Markov Chain transitions. Moreover, the formal nature of the Markov Chain allows us to evaluate the probabilities relating to the real-time system, like the response time distribution.

We obtain a schedulability analysis for already existing scheduling algorithms and we do not obtain a schedule yet. We assume a given task set with pWCET on a uniprocessor machine. We bring our attention at the job level as the periodic instances of the tasks within the hyperperiod. The pWCET is assumed to be described with exponential distribution. The scheduling analysis copes with preemptive Earliest Deadline First or Fixed Priority scheduling as given in [But11]. We begin by presenting the Continuous Time Markov Chain.

Dans ce chapitre, nous présentons notre premier pas vers le sujet de thèse. Le sujet de thèse est formellement mettre comme suit.

Étant donné un système probabiliste en temps réel avec des exécutions de tâches décrites avec pWCET continu, quelle est la probabilité que les tâches manquent leur échéance?

Nous commençons les contributions dans cette thèse en répondant à la question posée ci-dessus qui directement se rapporte au sujet de la thèse. On nous donne un ensemble de tâches dans lequel les exécutions de tâches sont décrites par une distribution de probabilité continue. L'objectif est de réaliser une analyse d'ordonnancement de l'exécution des tâches pour obtenir la probabilité de manquer leur échéance. Aux premier regard, on voit déjà que la programmation de telles tâches implique un certain jeu de probabilités au sein du système qui se propage et affecte la fonctionnalité du système de manière probabiliste. De notre discussion dans l'introduction, ici nous plongons formellement plus profondément dans cette idée.

Nous obtenons un modèle d'analyse d'ordonnancabilité utilisant la chaîne de Markov, en particulier Chaîne de Markov Temps Continu (CTMCh). Nous modélisons les exécutions de tâches système à l'aide de CTMCh. Chaîne de Markov Temps Continu est un ensemble d'états avec des transitions unidirectionnelles entre eux. Les transitions sont étiqueté avec le paramètre de la distribution exponentielle qui correspond à la probabilité de cette transition ainsi que le temps probabiliste passé dans la transition. Cela permet une cartographie directe des pWCET continu aux transitions de la chaîne de Markov. De plus, le caractère formel du Chaîne de Markov nous permet d'évaluer les probabilités relatives au système en temps réel, comme le distribution probabiliste du temps de réponse.

Nous obtenons une analyse d'ordonnancabilité pour les algorithmes d'ordonnancement déjà existants et nous n'obtenons pas un ordonnance encore. Nous supposons un ensemble de tâches donné avec pWCET sur une machine monoprocesseur. Nous apportons notre attention au niveau du travail comme les instances périodiques des tâches dans l'hyperpériode. Le pWCET est supposé être décrit avec une distribution exponentielle. L'analyse d'ordonnancement fait face à l'Earliest Deadline First (EDF) ou Fixed Priority (FP) comme indiqué dans [But11]. Nous commençons par présenter le Chaîne de Markov Temps Continu.

3.1 Continuous Time Markov Chain

This section defines the necessary fundamentals of Markov Chain as given in [Nor97] in relation to the real-time systems.

Definition 3. *A Markov chain is a set of random variables representing states* $\{P_k\}_{k\geq 0}$ *taking values* v_k *in a countable set.* $\{P_k\}_{k\geq 0}$ *is a MCh if it respects the property:* $Pr(P_{k+1} = v_{k+1}|P_0 = v_0, P_1 = v_1 \ldots P_k = v_k) = Pr(P_{k+1} = v_{k+1}|P_k = v_k)$.

This property is called the Markov property which says that the conditional probability of future state P_{k+1} depends only on the present state P_k and not on the events in the past. P_k are the states and v_k are the values assumed by the states; $P_k = v_k$ is the event v_k for the k-th state.

A Discrete Time Markov Chain (DTMCh) is a MCh in which the state transitions occur at discrete time. Each outgoing transition from a state has a probability of being chosen; the sum of the probabilities of all the outgoing transitions is one.

A Continuous Time Markov Chain (CTMCh) is a MCh in which there are exponential rates λ associated to the state transitions. This implies that there is an exponentially distributed time spent in a transition. The transitions for a CTMCh are formalized as a Q-matrix $Q = (q_{ij} : i, j \in I)$ where each elements q_{ij} is an exponential rate and describes the transition itself.

$$\begin{bmatrix} & P_0 & P_2 & P_3 & \dots \\ P_0 & -(q_{02}+q_{03}) & q_{02} & q_{03} & \dots \\ P_2 & q_{03} & -(q_{03}+q_{23}) & q_{23} & \dots \\ P_3 & q_{31} & q_{32} & -(q_{31}+q_{32}) & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

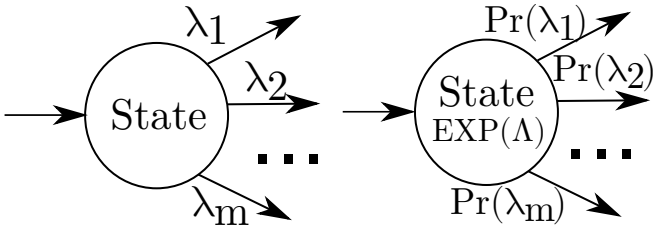
with the conditions: $-\infty \leq -q_{ii} < 0, \forall i$; $q_{ij} \geq 0, \forall i \neq j$; $\sum_{j \in I} q_{ij} = 0, \forall i$. A CTMCh model is represented with $\{X, Q\}$, where X is the set of states and Q is the transition matrix.

CTMCh can be represented as embedded DTMCh with (i) exponentially distributed time spent in a state and (ii) probability of choosing an outgoing transition from that state, given in [KNP07]. The probability of choosing a state transition with rate λ_r out of m outgoing state transitions is given by:

$$Pr(\lambda_r) = \frac{\lambda_r}{\sum_{k=1}^{m} \lambda_i}, r \leq m, \tag{3.1}$$

and the rate of exponentially distributed time spent in each state, denoted by Λ, is given as:

$$\Lambda = \sum_{i=1}^{m} \lambda_i. \tag{3.2}$$



(a) CTMCh (b) DTMCh

Figure 3.1: CTMCh and the corresponding embedded DTMCh.

This is shown in Figure 3.1 where Figure 3.1a is a portion of CTMCh with one state and m outgoing transitions. Each transition, has a probabilistic duration given by an exponential rate λ_r. In Figure 3.1b the correspondent embedded DTMCh is illustrated which is decomposed as: (i) probability of choosing a transition $Pr(\lambda_r)$, Equation (3.1); and (ii) the exponentially distributed time spent in the

state, Equation (3.2). These two equations are exploited to understand and model a pRTS.

Model checking can be performed on CTMCh models. This means verification of certain properties of the model in order to obtain probabilistic task execution characteristics, like probability of being in a state at some time. The model checking is done to verify that a model meets certain probabilistic specification. For a job J, the function $Pr(J, \text{state}, \text{time})$ expresses the probability that the CTMCh of J is in state $state$ at time $time$. For example, $Pr(J, \text{finished}, \text{deadline})$ checks the CTMCh model of the job J and returns the probability that it has finished execution at the deadline. The argument $time$ could be a time instant or a time interval. Property $Pr(J, \text{finished}, [time, \text{deadline}])$ is also a valid property which checks if the job J is in state $state$ between time instances $time$ and $deadline$.

PRISM model checker [KNP07] is the tool that we apply to formally check the CTMCh models and the probabilistic schedulability analysis. It is used to verify the timing properties of CTMCh models such as $Pr(J, \text{state}, \text{time})$ by solving the CTMCh matrices for transient probabilities.

3.2 Pessimism, Exponential Upper Bounding, and Safety

Here, we form the link between existing pWCET distributions and the exponential distributions in order to map them to the CTMCh model. We begin with definition of safety. Any schedulability analysis is safe if it does not provide optimistic results.

Definition 4 (Pessimism)**.** *Given two random variables C_1 and C_2 with PDFs f_1 and f_2, respectively. C_2 is said to be more pessimistic than ("larger than or equal to") C_1 in the time interval $[a,b]$, iff:*

$$\int_a^b f_2 \geq \int_a^b f_1. \tag{3.3}$$

The interval $[a,b]$ belongs to the support[1] *of both C_1 and C_2.*

A larger/pessimistic distribution than one given, can be used as a safe representation for it. Equation (3.3) is taken from [Dia+04] and it defines the partial ordering between distributions in a specific interval $[a,b]$. Here, the focus is in a certain interval in the support which provides a more general definition of pessimism. Throughout the thesis, by safety we mean a safe approximation of the associated probabilities. This is not to be confused with the safety of the software systems.

In order to form the link between any given continuous distribution and the exponential distribution that we use, we make the following interpretation. Given a PDF f, it is always possible to find a pessimistic representation with an exponential distribution. It has to be a larger than f, at least in the upper part of its support, the interval $[a,b]$ in Equation (3.3) as described next. We name an exponential distribution like that a safe upper bound for f.

Lemma 1 (Safe Exponential Upper Bounding)**.** *Given a PDF f and another PDF f_E of an exponential*

[1] The random variable support is the interval in which the random variable can have values.

distribution of rate λ which intersects f in P_{int}, f_E is a safe upper bound for f if:

$$\lambda \leq -\frac{1}{P_{int}} ln(1 - \int_0^{P_{int}} f); \tag{3.4}$$

$P_{int} > 0$.

Proof. Using Equation (3.3), an exponential distribution $f_E = \lambda e^{-\lambda x}$ safely upper bounds f from P_{int} to infinity if:

$$\int_{P_{int}}^{\infty} f_E \geq \int_{P_{int}}^{\infty} f$$

Since the integration of PDF gives the cumulative probability, and the total probability in $[0, \infty)$ cannot exceed one, it is:

$$1 - \int_0^{P_{int}} f_E \geq 1 - \int_0^{P_{int}} f$$

and

$$e^{-\lambda P_{int}} \geq 1 - \int_0^{P_{int}} f.$$

In order to satisfy the former inequality, λ has to be such that:

$$\lambda \leq -\frac{1}{P_{int}} ln(1 - \int_0^{P_{int}} f).$$

□

The above definition and lemma state the notion of safety for pRTSs. With pessimistic representations and correct approaches, the schedulability analysis with probabilities is guaranteed to be safe. To note that the pWCET from MBPTA have shapes of either a generalized extreme value distribution or a generalized Pareto distribution [Caz+11]; [LDB16]; [SGM17].

We use a factor of safety *FoS* to define P_{int}. For a job J, P_{int} is the product of deadline d and the *FoS*; $P_{int} = d.FoS$ with $0 \leq FoS \leq 1$. *FoS* close to 1 implies P_{int} close to d, which means reducing the pessimism in $[0, d]$ with a small margin for errors in the P_{int} estimation. *FoS* close to O implies P_{int} close to 0, which means increasing the pessimism with a large margin for errors in the P_{int} estimation.

Figure 3.2 shows an exponential distribution CDF F_E upper bounding distributions with different shapes. All the distributions are defined in a support $[0, \infty)$. The distributions considered are: a Gumbel distribution, as one of the generalized extreme value distributions, a Pareto distribution, from the generalized Pareto distribution, and a Convolution distribution, resulting form the convolution of two exponential distributions.

F_E intersects the three distributions in three different points, P_{int}^G, P_{int}^P, and P_{int}^C respectively for the Gumbel, the Pareto and the Convolution distributions. In $[P_{int}^G, \infty)$, F_E upper bounds the Gumbel distribution; in $[P_{int}^P, \infty)$, F_E upper bounds the Pareto distribution; in $[P_{int}^C, \infty)$, F_E upper bounds the Convolution distribution.

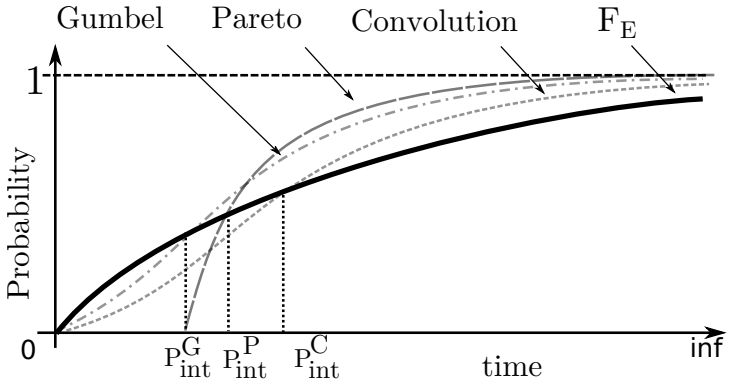


Figure 3.2: Exponential *EXP* upper bounding distributions with a certain rate.

3.3 Job Execution Interference Definitions

Using the above definitions, we evaluate the job backlogs in detail. Backlog is the delay caused to a job by the execution of certain other higher priority jobs. Because the execution is probabilistic, the backlog is also probabilistic. The notion of backlog is used to represent job interference and build CTMCh models. We define the backlog and classify various ways in which it is produced. There are three possible scenarios which impose a backlog to the victim job J as follows.

Case1 - Preceding job: A job which is released before job J, which delays the execution of J is the set $J^{prd}(J) \stackrel{def}{=} \{J' : p' < p, a' < a\}$ and $p' - p$ is minimum for all possible J'. As will be seen in later sections, the process of analysis is sequential in the order of decreasing job-priority. Because of this approach, it is enough to obtain the backlog from the previous job and we do not need to consider all the jobs in the past. The cardinality[2] of $J^{prd}(J)$ is always one.

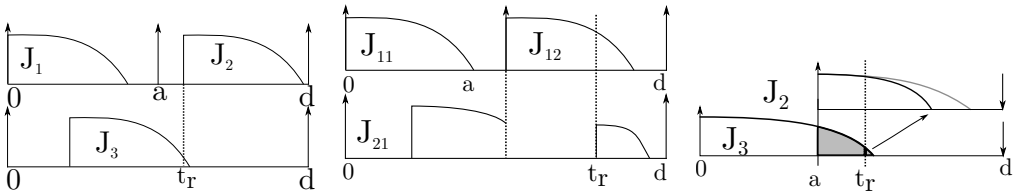
Case2 - Synchronous job: A set of higher priority jobs released synchronously to J which execute first and thus delay job J. It is the set $J^{syc}(J) \stackrel{def}{=} \{J' : a' = a, p' < p\}$. The total delay to the job J by the jobs in $J^{syc}(J)$ is given by the convolution of the pWCETs of all the jobs in the set.

Case3 - Preempting job: Higher priority jobs preempt the already executing victim job. From the preemption instant the victim job is delayed and awaits execution. The effect of backlog is that it changes the execution distribution of the victim job. The resulting new exponential distribution incorporates all the backlogs affects. The set of preempting jobs is defined as $J^{prm}(J) \stackrel{def}{=} \{J' : a' > a, p' < p, a' < d\}$. $J^{prm}(J)$ is ordered in increasing arrival times of its constituent jobs. K is the maximum number of preemptions J can have and is given as $K = card(J^{prm}(J))$ which is the number of elements in the set.

To detail the backlog scenarios, we consider the jobs from two tasks are scheduled either using EDF or FP on a uniprocessor. Three are the jobs in the hyperperiod, J_1, J_2 and J_3 with arrival times a_1, a_2 and a_3; deadline d_1, d_2 and d_3; and priorities p_1, p_2 and p_3. The jobs scheduled under EDF are shown in Figure 3.3a while the FP schedule is shown in Figure 3.3b. The executions are represented in

[2]Given a set S, the cardinality of S is represented as $card(S)$ which gives the number of elements in S

the ICDF form. The generalization of the backlog to any task set is left for the following sections.



(a) EDF scheduling with no preemption (b) FP scheduling with J_3 pre-empted by J_2 (c) Backlog Δ in $[a,d]$ for J_2 from J_3

Figure 3.3: Various possible delays to the execution of a job.

Case1 - Job released earlier: *High priority job arriving earlier and postponing*.

Job J_2 arrives at time a and is the job under investigation; Job J_3 is released earlier and delays job J_2, as shown in Figure 3.3c. The deadline for both the jobs is d.

The event that job J_2 finishes at the deadline d is equal to the events that J_2 begins at the arrival time a and ends at the deadline d.

$$Event(J_2 \text{ finishes at } d \text{ after } J_3) = Event(J_2 \text{ begins at } a) \text{ and } Event(J_2 \text{ finishes at } d)$$

The event that J_2 begins at the arrival time is equal to the event that J_3 ends execution at a.

$$Event(J_2 \text{ finishes at } d \text{ after } J_3) = Event(J_3 \text{ ends at } a) \text{ and } Event(J_2 \text{ finishes at } d)$$

The execution of J_3 is given by its response time and the execution of J_2 is given by its pWCET. Let, the random variable of execution of J_2 is $\mathcal{C}_2$, the response time random variable of J_2 is $\mathcal{R}_2$ and that of J_3 is $\mathcal{R}_3$. Because the response time of J_3 does not affect the duration for which J_2 will execute, $\mathcal{R}_3$ and $\mathcal{C}_2$ can be assumed independent. The probabilities are thus,

$$Pr(\mathcal{R}_2 = d) = Pr(\mathcal{R}_3 = d - a)Pr(\mathcal{C}_2 = d)$$

Function $f_{\mathcal{R}_2}$ is an exponential distribution because we need to map it to the CTMCh transitions. We obtain the $f_{\mathcal{R}_3}$ from the CTMCh model of the job J_3. f_2 is the pWCET of the job J_2. Therefore,

$$\int_0^d \lambda_2 e^{-\lambda_2 x} dx = (1 - \Delta^{J_3}_{[a,d]}) \int_0^d f_2$$

$$\lambda_2 \stackrel{def}{=} -\frac{1}{d} ln(1 - (1 - \Delta^{J_3}_{[a,d]}) \int_0^d f_2), \tag{3.5}$$

where $\Delta^{J_3}_{[a,d]} = \int_a^{d-a} f_3 dx$ gives the backlog from J_3 to J_2. Thus, $1-\Delta^{J_3}_{[a,d]}$ is the probability that J_3 has finished execution in $[a,d]$ as represented in Figure 3.3c. The superscript of Δ denotes the job from which the backlog exists, here J_3. The subscript denotes the time interval during which the backlog exists, here the interval $[a,d]$; $1-\Delta^{J_3}_{[a,d]} = Pr(J_3, \text{finished in } [a,d])$. This notation will be used throughout this paper. We obtain this value from the analysis of the CTMCh model of J_3.

We notate the backlog as $1-\Delta^{J_3}_{[a,d]}$ to denote that we obtain this value from the CTMCh analysis of the job. Moreover, we notate as a subtraction from one to prevent loss of accuracy due to decimal point representation in the computer program.

Case2 - Synchronously released jobs: *High priority job arriving synchronously and postponing*.
Here, J_3 is the task under investigation, while jobs J_1 and J_3 are synchronously released, see Figure 3.3a. Here, the exact finishing time of J_1 is unknown. We cannot discuss in terms of events like in the previous case because a deterministic time across which discrete observations can be made is unknown.

The random variable of execution of J_3 is $\mathcal{C}_3$ that of J_1 is $\mathcal{C}_1$, response time random variable of J_1 is $\mathcal{R}_1$ and that of J_3 is $\mathcal{R}_3$. Thus,

$$\mathcal{R}_3 = \mathcal{R}_1 + \mathcal{C}_3$$

$$\mathcal{R}_3 = \mathcal{C}_1 + \mathcal{C}_3$$

Thus, to know the combined affect of J_1 and J_3, the pWCETs of $\mathcal{C}_1$ and $\mathcal{C}_2$ are convolved. As a reminder, CTMCh only accepts exponential distributions.

$$\int_0^{P_{int}} \lambda_3 e^{-\lambda_3 x} dx = \int_0^{P_{int}} f_{\mathcal{R}_1} \otimes f_2$$

The rate λ_3 of the exponential upper bound for job J_3 is:

$$\lambda_3 \stackrel{def}{=} -\frac{1}{P_{int}} ln(1 - \int_0^{P_{int}} f_{\mathcal{R}_1} \otimes f_2). \tag{3.6}$$

where $f_{\mathcal{R}_1}$ and f_2 are the response time and the pWCET PDFs of the jobs J_1 and J_2 respectively. The choice of λ depends on P_{int}. In this specific case, $f_{\mathcal{R}_1}$ is equal to the pWCET f_1 because J_1 is the first job executing.

Case3 - Job preempting: *High priority job arriving after and preempting*.
To illustrate this case, consider Figure 3.3b with the FP scheduling with J_3 being the job under investigation. In this case, J_2 preempts J_3 at time a.

The event that job J_3 finishes at the deadline d is equal to the events that J_3 executes at the arrival time a and ends at the deadline d.

$$Event(J_3 \text{ finishes at } d \text{ after } J_2) = Event(J_3 \text{ executes at } a) \text{ and } Event(J_3 \text{ finishes at } d)$$

The event that J_2 begins at the arrival time is equal to the event that J_3 ends execution at a.

$$Event(J_3 \text{ finishes at } d \text{ after } J_2) = Event(J_2 \text{ ends at } a) \text{ and } Event(J_3 \text{ finishes at } d)$$

The execution of J_3 after preemption is given its pWCET and the execution of J_2. Let, the random variable of execution of J_3 is $\mathcal{C}_3$, the response time random variable of J_2 is $\mathcal{R}_2$ and that of J_3 is $\mathcal{R}_3$. Since the response time of J_2 does not affect the duration of execution of J_3, $\mathcal{R}_2$ and $\mathcal{C}_3$ can be assumed independent. The probabilities are thus,

$$Pr(\mathcal{R}_3 = d - a) = Pr(\mathcal{R}_2 = a)Pr(\mathcal{C}_3 = d)$$

Function $f_{\mathcal{R}_3}$ is an exponential distribution because we need to map it to the CTMCh transitions. We obtain the $f_{\mathcal{R}_2}$ from the CTMCh model of the job J_3. f_3 is the pWCET of the job J_3. Therefore,

$$\int_0^{d-a} \lambda_3 e^{-\lambda_3 \, x} dx = (1 - \Delta^{J_2}_{[0,d]}) \int_a^d f_3$$

$$\lambda_3 \stackrel{def}{=} -\frac{1}{d-a} ln(1 - (1 - \Delta^{J_2}_{[0,d]}) \int_a^d f_3), \tag{3.7}$$

where f_3 is the pWCET PDF of the job J_3; and $(1 - \Delta^{J_2}_{[0,d]})$ is the probability that the job J_2 finishes by its deadline. We obtain the value of $\Delta^{J_2}_{[a,d]}$ from the CTMCh model of J_2.

Therefore, the pWCET of job J_3 changes and the rate λ_3 of the exponential distribution accounts for the probabilistic waiting time and the remaining execution of itself.

These job classifications are depicted in Figure 3.4 for each J. The job executions are represented in the ICDF form to differentiate the case of pWCETs from deterministic WCETs. In here, the worst-case execution is described with a random variable, and the ICDF captures the distribution law as well as the probabilistic behaviour that jobs follow.

Regarding these sets classifying the backlog, the scheduling follows either Earliest Deadline First (EDF) or Fixed Priority (FP) preemptive paradigms [But11]. The scheduling policy defines the job ordering by imposing job-wise priorities. EDF or FP would have a different job ordering with effects on the job sets $J^{prd}(J), J^{syc}(J), J^{prm}(J)$ which define the task interference. We assume that the jobs are suspended if their execution reaches the deadline. We also assume deadline is always less than or equal to the periods. The hyperperiod $hp = lcm(T), \forall \tau \in \Gamma$ gives the scope of the schedulability analysis for EDF or FP.

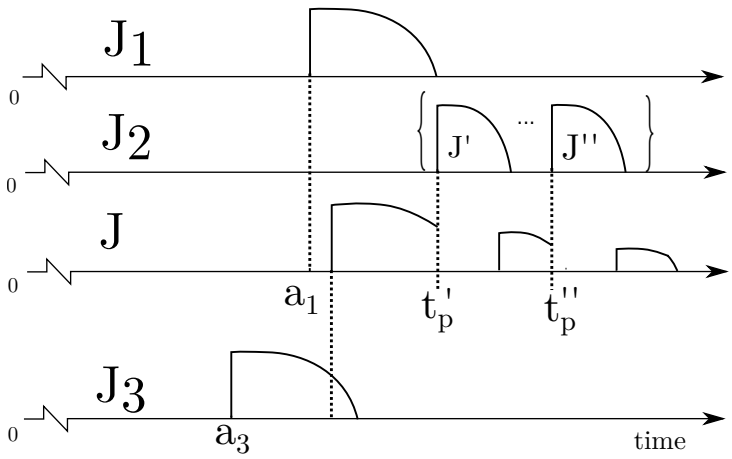


Figure 3.4: Job J, job set $J^{prd}(J)$, job set $J^{syc}(J)$, and job set $J^{prm}(J)$ are represented with interactions between them.

3.4 Deterministic Observations

The probabilistic execution of jobs is described using continuous distributions. Since we assume that the jobs are described with continuous pWCET defined in $[0,\infty)$, there exists a non-zero probability that a task executes until any time after arrival. This also implies that there exists a non-zero probability that the task gets preempted after its arrival. With a set of such executing jobs, the system behaviour is seen as continuous and uninterrupted in $[0,\infty)$. In order to observe a system composed of such tasks, we need to define certain points of observations in time. The formal properties are later checked on these points. Moreover, the assumption of suspending tasks slightly changes the case with intervals limited to $[0,d]$. The convolution does not take into account the offset between arrivals and the cutting of the distributions at the deadline. In the following sections we will show how such suspended case is treated in CTMCh models. We sample the continuously distributed probabilistic system at certain discrete points. These discrete points are defined as follows.

Definition 5 (Observation Points and Preemptions)**.** *Given a job J and its preempting jobs from $J^{prm}(J)$, the time instants at which job J gets preempted and the time instant at which it is mandatory to finish execution, are the job observation points is given in the set $T_p(J)$ as:*

$$T_p(J) \stackrel{def}{=} \forall J' \in J^{prm}(J) : \{a' - a\} \cup d - a \tag{3.8}$$

$T_p(J)$ is ordered by increasing time. If there are two or more equal time instances, they are ordered by increasing priority of the job to which they belong. $T_p(J)$ elements are referred to with superscript k as T_p^k, $k = 1,2,3,\ldots,K,K+1$, with K the cardinality of $J^{prm}(J)$ for the job J. The time instances in the set $T_p(J)$ are defined such that they are relative to the job to which it refers to, in this case J, and the arrival time a is the local zero.

It is sufficient to reduce $T_p(J)$ from Equation (3.8) to unique and increasingly ordered observation points. Of the multiple elements T_p^k which are of the same value, the ones which belong to lowest

priority jobs are kept. This is because of the two preempting jobs arriving at the same time, the CTMCh model of the lower priority job already accounts for the backlog from the higher one.

$$\nexists (J', J'') \in T_p(J) : a'' = a' \text{ and } p'' > p' \tag{3.9}$$

3.5 Modelling and Analyzing Probabilistic Real-Time Systems

This section builds CTMCh models for pRTS with continuous distributions, and how to perform schedulability analysis from those models. It combines the notions defined earlier, job backlog classification, deterministic observations and exponential upper bounding. Consider a given task set scheduled under preemptive EDF or FP on a uniprocessor, and a continuous pWCET for each task. For each job in the hyperperiod, there is a CTMCh model which accounts for all the interference specific to the job.

The scheduling policy chosen (EDF or FP) imposes a priority to each job and a job ordering. The job J to be modelled and analyzed is selected in the order of decreasing priority with the so called sequential approach. We recall the sets $J^{syc}(J)$, $J^{prm}(J)$ and $J^{prd}(J)$ which categorize delays to the job. These sets depends on the scheduling policy. The process to build CTMCh models does not depend on the scheduling policy, since all the dependence due to the order of execution are embedded into the job sets and the job ordering. All the CTMCh models are then used for the probabilistic schedulability analysis.

As shown in Figure 3.5, for each job J, in order to build its CTMCh model it is necessary to know the job sets $J^{prd}(J)$, $J^{syc}(J)$ and $J^{prm}(J)$. The three blocks 'Backlog', 'Preemption', and 'CTMCh' described in the following subsections, applies such information. Figure 3.5 illustrates the building of a CTMCh model with the inputs for a job J and the logical order between the blocks.

The construction of a CTMCh job model requires identification of states and state transitions. For a job J, an initial set of states for the CTMCh is given, $X = \{P_0, F\}$ with P_0 representing execution after release and F representing end of execution. In that case, the transition represents the execution of the job by including the backlog from all the jobs released synchronously and/or the job released earlier, sets $J^{prd}(J)$ and $J^{syc}(J)$. The exponential rate of the transitions is given by the block 'Backlog'. Then, depending on the number of preemptions to the job, given by the set $J^{prm}(J)$, new states, and corresponding transitions, are added to the CTMCh. The new transition from the preempting state to the end of execution includes the waiting time of the job due to preemption as well as the remaining execution after preemption. This is made within the 'CTMCh' block. The computation of the CTMCh transitions from preemptions requires certain probabilistic information as properties $Pr(\text{job}, \text{state}, \text{time})$, computed by the block 'Preemption'. This is detailed and formalized as follows.

3.5.1 Backlog

In the block 'Backlog', we obtain the backlog to a job when it begins execution. In this case, the only sources the backlogs can come from are the preceding job and the parallel released jobs. For J, the

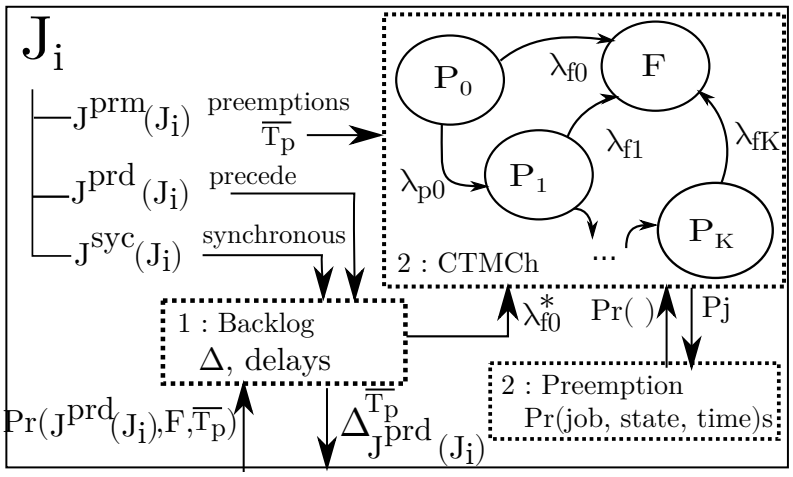


Figure 3.5: Job CMTCh model formalization with blocks and input information.

input are the job sets $J^{prd}(J)$ and $J^{syc}(J)$ which decide the type of backlog that exists. The output of 'Backlog' is the rate λ^*_{f0} which models the job execution with those backlogs.

In the case where J is the highest priority job, then it does not have any backlog. Instead, if J is released synchronously with other jobs; $J^{syc}(J) \neq \emptyset$ and the backlog is determined by Equation (3.6). Finally, if there exist higher priority jobs than J that arrived earlier than J, $J^{prd}(J) \neq \emptyset$, and the job backlog also comes from Equation (3.5). These two cases can also occur simultaneously.

At this stage of the modelling, there are two states in the CTMCh model of J, P_0 and F. As a reminder, we stated that $J^{prd}(J)$ has only one job because the sequential approach we are building requires only one preceding preceding job (and the one with the closest priority to J) to obtain the delay to J.

The exponential transition between P_0 and F is given by the exponential rate λ^*_{f0} which is determined as follows:

a) If J is the highest priority job, λ^*_{f0} is as given by pWCET PDF f.

b) If J has non empty $J^{prd}(J)$ and empty $J^{syc}(J)$, the backlog is from previously released high priority job (Case1).

$$\lambda^*_{f0} \stackrel{def}{=} -\frac{1}{d} ln(1-(1-\Delta^{J^{prd}(J)}_{[a-a',d'-a']}) \int_0^d f), \forall J' \in J^{prd}(J) \tag{3.10}$$

c) If J has non empty $J^{syc}(J)$ and empty $J^{prd}(J)$, the backlog is from synchronously released higher priority job only.

$$\lambda^*_{f0} \stackrel{def}{=} \frac{-1}{P_{int}} \int_0^d f \otimes f'. \tag{3.11}$$

d) If J has non empty $J^{prd}(J)$ and non empty $J^{syc}(J)$, the backlog is the combination of Case1 and Case2:

$$\lambda^*_{f0} \stackrel{def}{=} -\frac{1}{d_i} ln(1-(1-\Delta^{J^{prd}(J)}_{[a-a',d'-a']}) \int_0^d f \otimes (\otimes_{\forall J'' \in J^{syc}(J)} f''), \forall J' \in J^{prd}(J) \tag{3.12}$$

It is to be noted that the upper limit of the integrals on the left hand side is the job relative deadline d of the job J. This is because the backlog affects the whole job and the safety of the upper bound is evaluated in $[0,d]$, with 0 being the arrival of the job a. The term $1-\Delta$ refers to the CMTC model of the job given by the superscript and returns the probability that it has finished execution in the time interval given by the subscript. Thus $1-\Delta^{J^{prd}(J)}_{[a-a',d'-a']}$ is the probability that the job $J^{prd}(J)$ has finished execution in the time interval $[a-a',d'-a']$. It is the property $Pr(J^{prd}(J), \text{finished}, [t_{ij}-T^k_{p_i}, t_{ij}])$ which is obtained using PRISM model checker.

Equation (3.10), Equation (3.11) and Equation (3.12) generalize the backlog computation for Case1 and Case2 formerly illustrated with the example in Section 3.3.

3.5.2 Preemption $Pr()$

The 'Preemption' block helps block 'CTMCh' with evaluating the timing properties of CTMCh models. For any job, 'Preemption' refers to the model checker to compute the probability that the job exists in an executing state *state* at time *time*. The resulting probability is returned to the 'CTMCh' block.

'Preemption' is only called if there is preemption. This is because preemptions cause adding states and state transitions to the CTMCh, and 'Preemption' has to evaluate job waiting and job execution after each preemption. The probabilities $Pr(J, \text{executing}, \text{time})$ are computed within 'Preemption' using PRISM model checker. This is explained next.

3.5.3 CTMCh States and State Transitions

In this subsection, we describe the 'CTMCh' block and its interactions with 'Backlog' and 'Preemption' to build the complete CTMCh model for a job.

For the job J arriving at time a with a deadline at d, an initial set of states $X = \{P_0, F\}$ is given.

The exponential transition between them has rate λ^*_{f0} which is provided by the block 'Backlog'. If there are no preemptions to the job, $\lambda_{f0} = \lambda^*_{f0}$, then the CTMCh model is complete.

However, there can exist a preemption to J at a time T^0_p, first element of the set T_p , by another higher priority job J' arriving at a' and with deadline at d'. In that case, a new state P_1 is added. The transitions now are λ_{p0} between P_0 and P_1; λ_{f0} between P_0 and F; and λ_{f1} between P_1 and F. The rates λ_{p0} and λ_{f0} are obtained by splitting the given rate $\lambda^*_{f_0}$ using Equation (3.1), i.e. probability of

choosing a transition, and Equation (3.2), i.e. probabilistic time spent in a state:

$$\lambda_{p0} = Pr(J, p', t_p^0).\lambda_{f_0}^*; \quad \lambda_{f0} = \lambda_{f0}^* - \lambda_{p0}.$$

The rate λ_{f1} accounts for the probabilistic waiting time by the job and its remaining execution at the time of preemption. It is determined using Equation (3.7) as:

$$\lambda_{f1} = -\frac{1}{d' - t_p^0} ln(1 - (1 - \Delta^J_{[d'-a', t_p^0 - a]}) \int_0^{d'-t_p^0} f' dx),$$

where f' is the pWCET of the preempting job J'.

If a second preemption to J exists, then state P_2 is added together with new transitions λ_{p1} between P_1 and P_2, λ_{f1} between P_1 and F (former λ_{f1} updated by the presence of P_2), and λ_{f2} between P_2 and F. The rates λ_{p1} and λ_{f1} are determined by splitting the former λ_{f1}, with Equation (3.1) and Equation (3.2); λ_{f2} is computed using Equation (3.7). At each preemption, the Q-matrix grows correspondingly as:

$$\begin{bmatrix} & P_0 & F \\ P_0 & - & \lambda_{f0} \\ F & 0 & 1 \end{bmatrix}^{[2,2]} \rightarrow \begin{bmatrix} & P_0 & P_1 & F \\ P_0 & - & \lambda_{p0} & \lambda_{f0} \\ P_1 & 0 & - & \lambda_{f1} \\ F & 0 & 0 & 1 \end{bmatrix}^{[3,3]} \rightarrow \begin{bmatrix} & P_0 & P_1 & P_2 & F \\ P_0 & - & \lambda_{p0} & 0 & \lambda_{f0} \\ P_1 & 0 & - & \lambda_{p1} & \lambda_{f1} \\ P_2 & 0 & 0 & - & \lambda_{f2} \\ F & 0 & 0 & 0 & 1 \end{bmatrix}^{[4,4]}$$

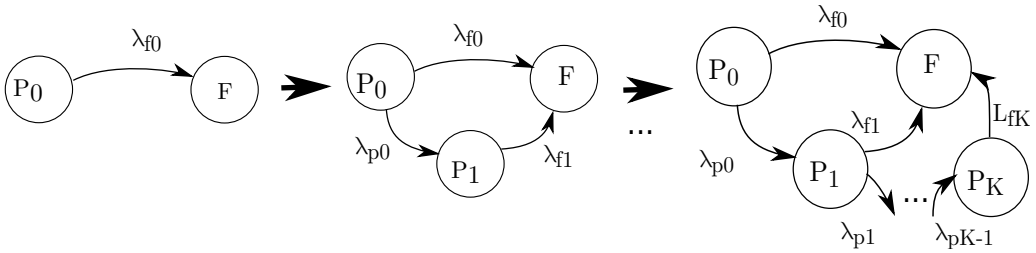


Figure 3.6: Iterative process to build a CTMCh model; the preemption effects are added and validated one preemption by one preemption.

The generalization of the CTMCh job modelling process is an iterative process, as illustrated in Figure 3.6. For the k-th ($k \leq K$) preemption, a state P_K is added to the CTMCh where K is the maximum number of preemptions to the job J. The new transitions are computed and validated from 'Preemption'.

The complete representation of the job is a set of states $X = \{P_0, P_1, ...P_K, F\}$. The final Q-matrix of size $[(K+2),(K+2)]$ is given as:

$$\begin{bmatrix} & P_0 & P_1 & P_2 & \dots & P_K & F \\ P_0 & -(\lambda_{p0}+\lambda_{f0}) & \lambda_{p0} & 0 & \dots & 0 & \lambda_{f0} \\ P_1 & 0 & -(\lambda_{p1}+\lambda_{f1}) & \lambda_{p1} & \dots & 0 & \lambda_{f1} \\ P_2 & 0 & 0 & -(\lambda_{pK}+\lambda_{f2}) & \dots & \lambda_{pK} & \lambda_{f2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P_K & 0 & 0 & 0 & \dots & -\lambda_{fK} & \lambda_{fK} \\ F & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}^{[(K+2),(K+2)]}$$

where the subscript of λ denotes i) the final state, f if it goes to state F or k if it goes to P_K state, and ii) the k-th state from which it goes out. For example, λ_{f2} denotes the rate of transition from state P_2 to state F.

Regarding the exponential rates in the Q-matrix, if $K = 0$ for the job J,it is $\lambda_{f0} = \lambda^*_{f0}$ and the model is complete. For $k = 1$ i.e. if set $J^{prm}(J)$ has one preempting job, $\lambda_{p1} = Pr(J, P_0, t^k_p).\lambda^*_{f0}$; $\lambda_{f1} = \lambda^*_{f0} - \lambda_{p1}$. And then for $1 < k \leq K$, λ_{pk} and λ_{fk} are:

$$\lambda_{pk} = Pr(J, P_k, t^k_p).\lambda_{fk-1}; \quad \lambda_{fk} = \lambda_{fk-1} - \lambda_{pk}. \tag{3.13}$$

λ_{fk+1}, as a generalization of Case3 and Equation (3.7), is:

$$\lambda_{fk+1} = \frac{-1}{t^{k+1}_p - t^k_p} ln(1 - (1 - \Delta^J_{[t^k_p, t^{k+1}_p]}) \int_0^{t^{k+1}_p - t^k_p} f^k); J^k \text{ is } k\text{-th job in } T_p(J)) \tag{3.14}$$

where $1 - \Delta^J_{[t^k_p, t^{k+1}_p]} = Pr(J, p_j, [t^k_p, t^{k+1}_p])$.

For each job, the CTMCh modelling is now complete and it accounts for all the possible interference which can delay job executions.

As an observation, there can be synchronous jobs in the set $J^{prm}(J)$ which can preempt J. For each subset of jobs in $J^{prm}(J)$, those which are synchronous between them, into the J CTMCh modeling it is sufficient to take into account for only the lowest priority job. This is because of the sequential of the approach we propose. Indeed, while modelling the preempting jobs which are synchronous between them (modelled before J), the delay of higher priority jobs is included into the model of the lowest priority job.

This also applies to the case where two synchronous victim jobs are delayed by a single preempting or earlier released job. There will be priorities assigned to the two victim jobs. The one with the higher priority will account for the preempting or the earlier released job. The one with the lower priority first accounts for the higher priority synchronous job which already contains the previous delays.

EDF vs FP. As already stated, EDF and FP impose different priorities to each job. The specific scheduling applied, changes the job ordering (job J absolute and relative priority) and the job interference $J^{prd}(J), J^{syc}(J), J^{prm}(J)$. Once accounted for that, the CTMCh modelling process we have illustrated applies with no difference between EDF or FP.

3.5.4 Schedulability Analysis

Once the job models are in hand, the probability of deadline miss and the response time curve for jobs and tasks can be extracted from them. This is what we call probabilistic schedulability analysis and is formalized as follows.

Probability of Deadline Miss. For a job J, the probability of deadline miss $Pr(DM)$ is given by one minus the probability that the job CTMCh is finished in the state F at the deadline, $Pr(J,F,\text{deadline})$:

$$Pr(DM) \stackrel{def}{=} 1 - Pr(J,F,\text{deadline}). \tag{3.15}$$

Response Time. The response time distribution $F_{\mathcal{RT}}$ of a job J can be obtained from the model checking. In particular, it comes from the probability that the job is finished by some positively increasing time t, $Pr(J,F,t)$:

$$F_{\mathcal{RT}} \stackrel{def}{=} Pr(J,F,t) = Pr(J \text{ finishes by } t),\ 0 \leq t \leq d. \tag{3.16}$$

For the job J, $F_{\mathcal{RT}}$ is evaluated in the interval $[0,d]$, since the job is suspended at the deadline and the information beyond d is not required.

Complexity. Given a task set Γ with m jobs in its hyperperiod. For each job, there are n^{syc} synchronously released higher priority jobs, n^{prm} preempting jobs, and only one preceding job (due to the sequential).

The convolution operation is applied to combine the n^{syc} jobs, Case2 in Equation (3.6) and its generalization from Equations (3.10)-(3.12). Numerical convolution requires a discretization of the distributions by a unit d; the convolution has complexity $O((D/d)^2)$, D being the job deadline. Case2 backlog is computed with also an integration; the trapezoidal rule uses the discretization of d unit, for a complexity of the integration of $O(D/d)$.

In case of preemptions, the rates are obtained with the integration operation, Case3 and Equation (3.7). Its complexity is $O(D/d)$. For each job it is $O(n^{syc}.((D/d)^2 + D/d) + n^{prm}.(D/d))$.

For the m jobs, there can be a maximum of $m-1$ synchronous jobs and $m-1$ preempting jobs, due to the sequential nature of the approach and the job ordering by decreasing priority. The total complexity of the modeling is then $O((m(m+1)/2).(D/d)^2)$. Without the sequential, the complexity would be $O(m^2.(D/d)^2)$.

Algorithm. The algorithm to build CTMCh models and perform schedulability analysis as defined in this paper, is detailed below in pseudo-code. A task set is given to be scheduled on a uniprocessor machine with scheduling policy EDF or FP, preemption enabled, and jobs that are suspended at their deadline. The factor of safety, which define the exponential upper bound is given and is same for each job.

The functions implemented are such that:

procedure MODEL_ANALYSE_PRTS(*tasks*, *policy*)
Order_Jobs(*Jobs*,*policy*) ▷ Order *Jobs* by their increasing priority
for each job in *Jobs* **do**
Define $J^{pre}(J)$, $J^{syc}(J)$, $J^{prm}(J)$, $\bar{T}_p(J)$ ▷ The higher priority jobs sets
Declare $X = \{P_0, F\}; Q = \{0,0;0,1\}$ ▷ Initial CTMCh
λ_{f0} =Backlog$(J, J^{pre}(J), J^{syc}(J))$ ▷ Backlog effects
$Q = \{-\lambda_{f0}, \lambda_{f0}; 0, 1\}$
for each preemptive job $J[k, J^{prm}(J)]$ in $J^{pre}(J)$ **do** ▷ Preemption effects
$Pr = P(J, p_j, t_p^k)$ ▷ k-th preemption
$\lambda_{fk} = \lambda_{f0}.Pr$
$\lambda_{pk} = \lambda_{f0} - \lambda_{fk}$
λ_{fk+1} = Delta_Pre$(J, J[k, J^{prm}(J)])$
Update(X, Q, k)
$Pr(DM) = 1 - Pr(J, F, t_{p_i}^{K+1})$ ▷ Probability of deadline miss
for time t **do**
$F_{RT}(t)$ =PRISM_Verify(J, F, t) ▷ Function of response time curve

Backlog$(J, J^{pre}(J), J^{syc}(J))$: Determines the backlog to the job (J) as defined in Equation (3.5)and Equation (3.6) or their generalized version from Equations (3.10)-(3.12), and depending on the sets $J^{syc}(J)$, $J^{pre}(J)$, $J^{prm}(J)$. It uses PRISM model checker to find the backlog from $J^{pre}(J)$.
Delta_Pre$(J, J[k, J^{prm}(J)])$: Calculates the exponential rate for execution after preemption. It uses PRISM model checker to calculate the remaining execution of J using its model.
Update$((X, Q, k))$: Update the CTMCh matrix by adding the new state and the corresponding rates at the appropriate positions.
$P(job, state, time)$: Uses PRISM Model checker to verify Pr(job, state, time); it returns the probability of job being in state $state$ at time instance or time interval $time$.

The code that implements the algorithm and the whole approach for CTMCh modeling and probabilistic schedulability analysis is available at `https://gitlab.com/MAUVE/RTStoc`. Is has been implemented in Python and includes the interface to PRISM model checker.

3.6 Experimental Results

This section presents some numerical results for the method presented in this paper. A real-time case study composed of 4 test cases is investigated . For each test case, our method is applied to model all the jobs in the test case hyperperiod and perform probabilistic schedulability analysis. The 4 test cases are used to explain different benefits from CTMCh applied to pRTSs.

3.6.1 Test Case 1

The first test case is a relatively simple real-time task set. It is Γ_1 that we apply to explain the overall process. It composes of three tasks $\Gamma_1 = \{\tau_1, \tau_2, \tau_3\}$; τ_1 with pWCET $\mathcal{C}_1 = EXP(5)$ and implicit deadline $D_1 = 2$; τ_2 with pWCET $\mathcal{C}_2 = EXP(6)$ and implicit deadline $D_2 = 3$; τ_3 with pWCET $\mathcal{C}_3 = EXP(7)$ and implicit deadline $D_2 = 6$. The hyperperiod is equal to 6 time units, which is also the scope of analysis. The task parameters are detailed in Table 3.1, and the times are expressed in time units. Γ_1 is scheduled on a uniprocessor machine using EDF, the jobs are suspended if they reach their

deadline, and can get preempted by higher priority jobs. For all the jobs, the *FoS* applied is 0.5.

Table 3.1: Task set Γ_1 parameters.

	C_i	$T_i = D_i$	$Pr(DM_i)$
τ_1	$EXP(5)$	2	1.28E-04
τ_2	$EXP(6)$	3	3.34E-05
τ_2	$EXP(7)$	6	4.98E-08

The ordered set of jobs, from the highest priority to lowest priority, is $\{J_1, J_3, J_2, J_{31}, J_{22}, J_{13}\}$. In case of priority ambiguities, the disambiguation mechanism applied bases on the job arrivals: first arrived are higher priority. The construction of CTMCh models is performed sequentially following that order, beginning from J_1.

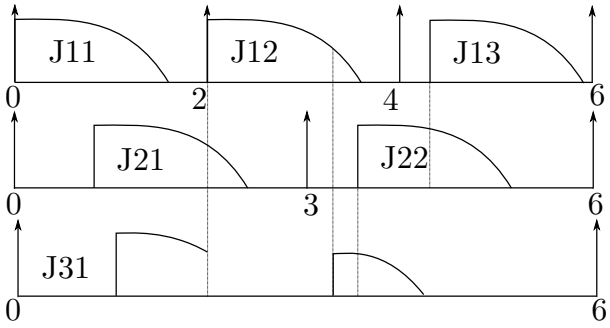


Figure 3.7: EDF scheduling

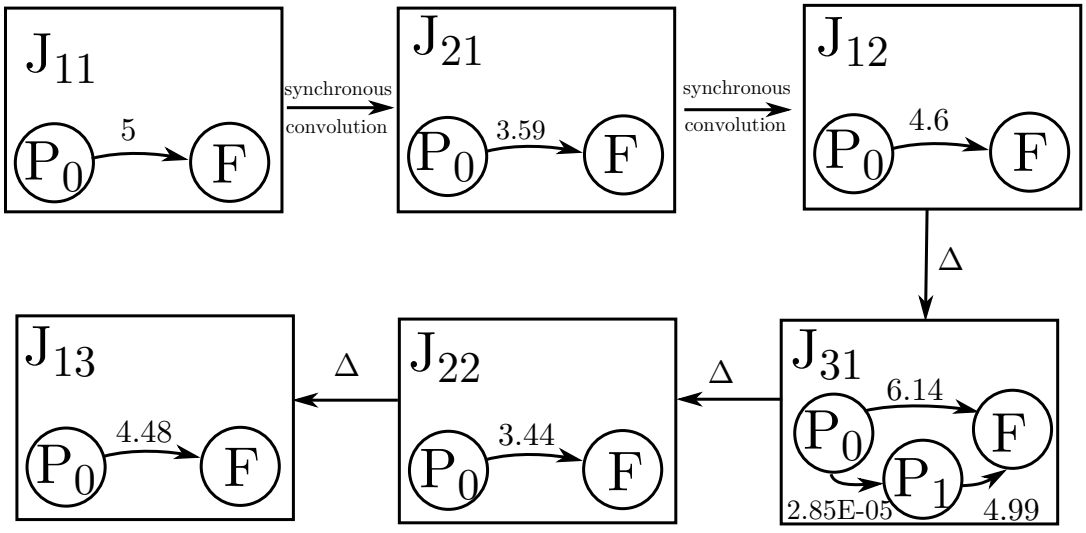


Figure 3.8: CTMCh job models

The job schedule is represented in Figure 3.7, with task executions as ICDF. The CTMCh model for each job is shown in Figure 3.8 as well as the models interconnections. In the following we detail interesting cases.

First job (highest priority): J_1 has the highest priority and cannot get preempted. Its CTMCh model is such that $X_1 = \{P_0, F\}$, with P_0 representing execution and F representing end of execution. J_1 Q-matrix is $Q_1 = \{-\lambda_{f0}, \lambda_{f0}; 0, 1\}$, with the rate of state transitions given by the job pWCET, since it not preempted nor pushed by any job. $\lambda_{f0} = 5 \implies Q_1 = \{-5, 5; 0, 1\}$.

Job with higher priority synchronous job, Case2: For J_3, J_1 is the job that precedes it and from which the backlog has to be computed. The rate of transition λ^*_{f0}, which is the pWCET for J_3 modified to include the delay caused by J_1, given by Equation (3.6) as: $\lambda^*_{f0} = -\frac{1}{3} ln(1 - \int_0^3 5e^{-5x} \otimes 6e^{-6x} dx) = 3.59$. For J_3 it is $X_1 = \{P_0, F\}$ and $Q_3 = \{-\lambda^*_{f0}, \lambda^*_{f0}; 0, 1\} = \{-3.59, 3.59; 0, 1\}$.

Job preempted, Case3: To model J_{31}, we have to take into account for the backlog from the higher priority synchronously released jobs J_1 and J_3, and the preempting job J_2.

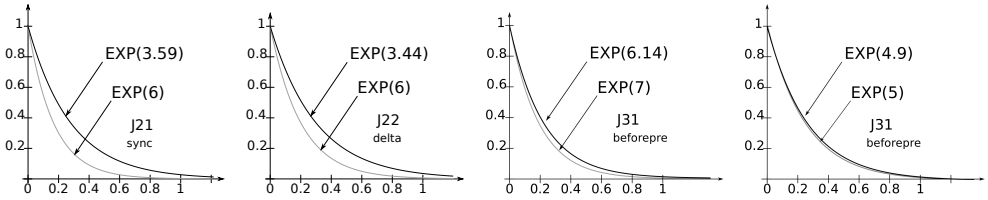
The pWCET of J_{31} that take into account the backlog from J_1 and J_3 is 6.14, computed like in Equation (3.6). Initially, the CTMCh for J_{31} it is $X_{31} = \{P_0, F\}$ and $Q_{31} = \{-6.14, 6.14; 0, 1\}$.

As there is the preemption from J_2, a state P_1 is needed to model that and to append to the CTMCh model. As a result, it is $X_{31} = \{P_0, P_1, F\}$. Using 'Preemption', the probability $Pr(J_{31}, P_0, 2)$ is calculated; it is the probability that job is executing at the time of preemption. $Pr(J_{31}, P_0, 2) = 4.38E-06$ and using Equation (3.1) and (3.2): $\lambda_{f0} = 6.14; \lambda_{p0} = 2.85E-05$. Using Equation (3.14) it is $\lambda_{f1} = 4.99$.

The complete CTMCh model for J_{31} is $X_{31} = \{P_0, P_1, F\}$ and $Q_{31} = \{-6.14, 2.85E-05, 6.14;$
$0, -4.99, 4.99; 0, 0, 1\}$.

Job with preceding job, Case1:

J_{22} it the job which gets postponed by J_{31}, job J_{31} is the previous job in the sequence of decreasing priority jobs. Thus J_{22} has a backlog from J_{31} in the time region $[3,6]$, $[0,3]$ relative to the job arrival a_{22}. The initial set of states is $X_{22} = \{P_0, F\}$; the rate of transition λ^*_{f0}, which represents the pWCET for that job by including the delay caused by J_{13}, is computed using Equation (3.10): $\lambda^*_{f0} = -\frac{1}{3} ln(1 - (1 - \Delta^{J_{31}}_{[3,6]}) \int_0^3 6e^{-6x} dx) = 3.44$, where $\Delta^{J_{31}}_{[3,6]} = 7.31E-08$ is the amount of backlog. Therefore, the Q-matrix is given as $Q_{22} = \{-\lambda_{f0}, \lambda_{f0}; 0, 1\} = \{-3.44, 3.44; 0, 1\}$, since $\lambda_{f0} = \lambda^*_{f0}$.



(a) J_3: synchronous jobs - Case2 (b) J_{22}: preceding job - Case1 (c) J_{31}: execution before preemption (d) J_{31}: execution after preemption

Figure 3.9: Job executions of Γ_1 before and after execution interference.

Figure 3.9 illustrates the job executions before and after the effect of the interference: 3 representative jobs of Γ_1 are detailed. The job execution distributions before and after the interference are

compared to describe the effect of the interference. All the executions are represented with exponential upper bounds.

In case of Figure 3.9a, the interference to J_3 comes from a synchronous job; the computed λ account for such backlog slowing down the job execution (larger probabilities at the deadline). In Figure 3.9b, it is illustrated the case of a preceding job; J_{22} is delayed by J_{31}. Figure 3.9c and Figure 3.9d illustrate J_{31} execution; in particular, it is the execution before being preempted by J_2 with the backlog received from J_1 and J_3 Figure 3.9c. Figure 3.9d instead, presents the execution after preemption including the preemption postponement with respect to the remaining execution at preemption time.

3.6.2 Comparison and Computation

Comparison test case. A task set Γ_2 is shown in Table 3.2 and scheduled using FP scheduling policy.

Task	C_i	$T_i = D_i$	$Pr(DM_i)$ Oris-tool	$Pr(DM_i)$ CTMCh model
τ_1	EXP(19)	1	8.34E-14	3.71E-08
τ_2	EXP(20)	1	6.66E-13	3.73E-08
τ_3	EXP(20)	3	0.0	4.73E-08

Table 3.2: Parameters for task set Γ_2 and task probability of deadline miss.

Γ_2 is modelled and analyzed using sPN by [Buc+10] as well as using our approach. For CTMCh and all the jobs, the *FoS* used is 0.5. Table 3.2 shows the probability of deadline miss as obtained from the two approaches, Oris-tool[3] for sPN and CTMCh model for our approach.

The Oris-tool provides exact result in terms of probability of deadline miss. Compared to it, the CTMCh modeling is more pessimistic due to exponential upper bounding. On the other hand, our approach has low complexity and thus can analyze larger task sets with much lower exponential rates.

To observe that the pessimism of the results from our tool depends also on the limit of the floating point computation of PRISM. Without that, we could have achieve better probability of deadline miss. To verify that, we have computed the probability of deadline miss as one minus the probability of executing at the deadline. The results are $Pr(DM_1) = 5.61E-09$, $Pr(DM_2) = 1.057E-06$ and $Pr(DM_3) = 1.23E-35$, unexpectedly different from those of Table 3.2 for CTMCh modelling. This confirms the floating point limits of PRISM in computing the probability of deadline miss.

Computation time test case. Here we present 3 cases to illustrate the computational complexity of our approach. They relate to different types of backlog that can exist in a task set. All the times are in time instants.

First, Γ_3^A is a set of n jobs, such that J_1 arrives at time 0 with deadline 2, J_2 arrives at time 1 with deadline 3, and J_n arrives at time n with deadline $n+1$, see Table 3.3. Except for the first job, each job is delayed by the previous job in the relative interval $[1,2]$. Such a set is scheduled under FP, with J_1 the highest priority and J_n the lowest. In Figure 3.10a, the computation time of the task backlog (Case1) is plotted against n, showing a linear trend.

[3]The Oris-tool is developed by the University of Florence for the analysis of timed and stochastic Petri nets, `http:`

Γ_3^A - Job	A_i	$T_i = D_i$
J_1	0	2
J_2	1	3
...	...	...
J_n	n	$n+1$

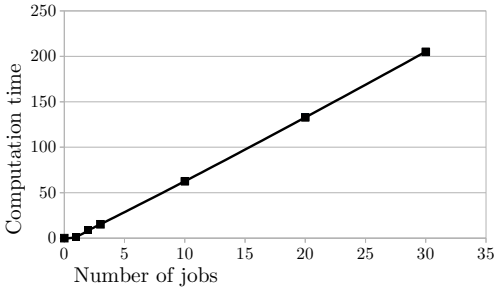
Γ_3^B - Job	A_i	$T_i = D_i$
J_1	0	1
J_2	0	1
...	...	...
J_n	0	1

Γ_3^C - Task	A_i	$T_i = D_i$
τ_1	0	1
τ_2	0	n

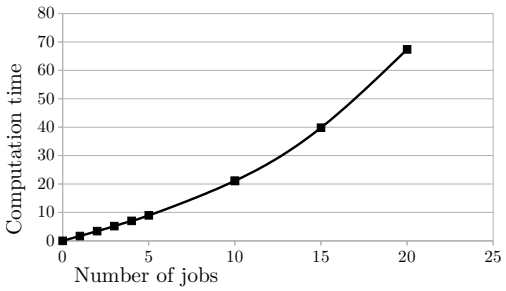
Table 3.3: Task sets Γ_3^A, Γ_3^B, and Γ_3^C to test computation time.

Second, Γ_3^B is the task set. It is a set of n jobs such that all the jobs arrive at 0 and have deadline equal to time 1. Except the first job, each job is delayed by the previously released synchronous jobs, Table 3.3. Γ_3^B is scheduled using FP with J_1 the highest priority and J_n the lowest. In Figure 3.10b, the computation time of this backlog (Case2) is plotted against n showing an exponential trend due to the convolution.

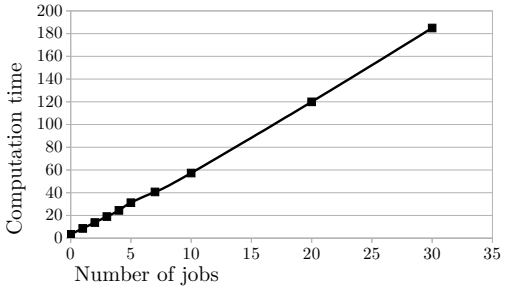
Third, Γ_3^C is a set of tasks such that τ_1 has period 1 and τ_2 has period n, Table 3.3. Γ_3^C is scheduled using FP with the n jobs of τ_1 having all higher priority than τ_2; τ_2 is preempted $n-1$ times. In Figure 3.10c, the computation time from preemption (Case3) is plotted against n, showing a linear trend.



(a) Computation time for the backlog from previous job



(b) Computation time for the backlog from synchronous jobs



(c) Computation time for the interference from preempting jobs

Figure 3.10: Complexity of CTMCh approach with types of interference.

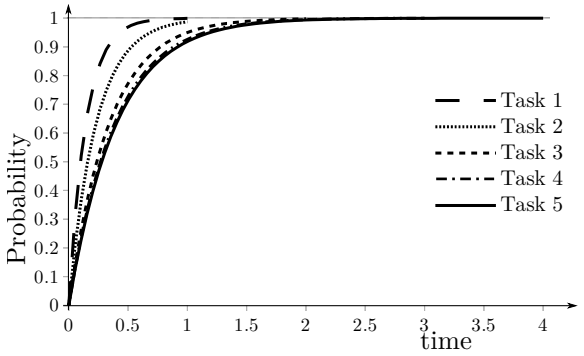
//www.oris-tool.org/.

3.6.3 Realistic real-time task set

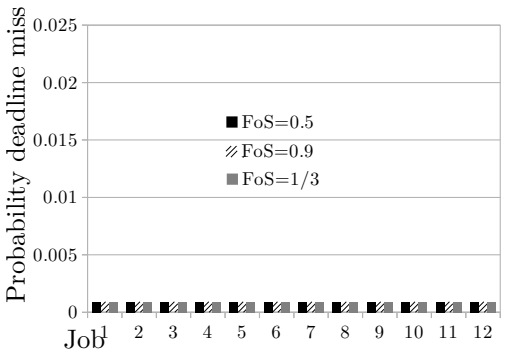
The fourth test case is a large task set Γ_4 composed of five tasks which parameters are shown in Table 3.4. For this task set, there are 36 jobs in the hyperperiod. Γ_4 is analyzed using CTMCh for both EDF and FP scheduling policies; the analysis for EDF takes 123.63 seconds; 147.77 seconds are necessary for FP.

Table 3.4: Task set Γ_4.

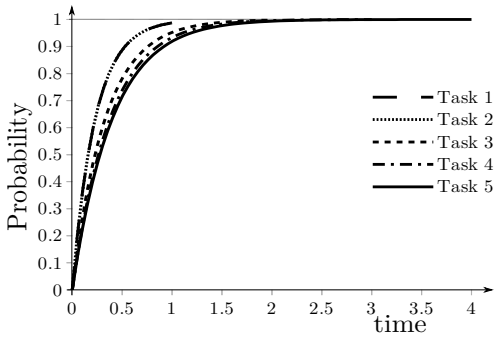
Task	C_i	$T_i = D_i$	$Pr(DM_i)$ EDF	$Pr(DM_i)$ FP
τ_1	EXP(7)	1	2.42E-02	9.12E-04
τ_2	EXP(8)	1	2.42E-02	2.42E-02
τ_3	EXP(5)	2	8.11E-03	8.84E-03
τ_4	EXP(6)	3	2.32E-03	3.67E-03
τ_5	EXP(6)	4	1.26E-03	1.27E-03



(a) FP: response time for all the tasks, computed with constant $FoS = 0.5$



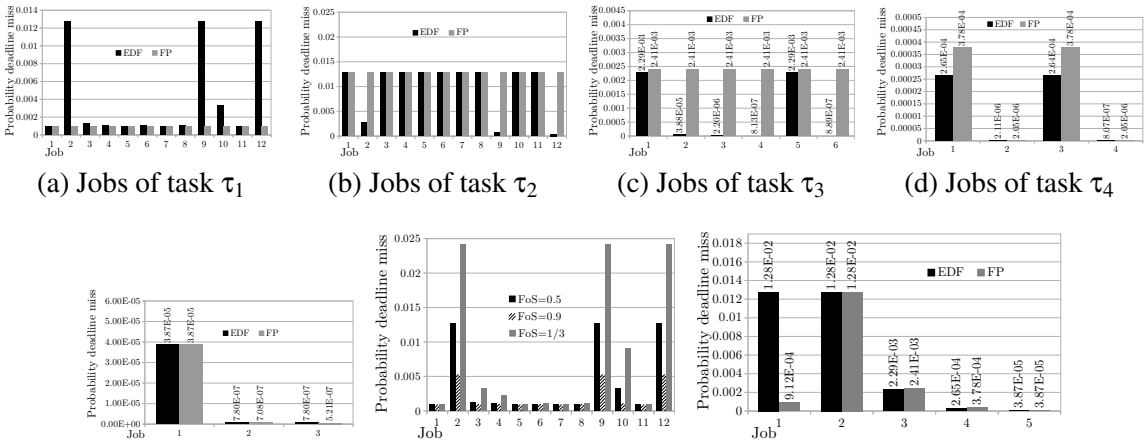
(b) FP: probability of deadline miss for all the jobs of τ_1 for different FoS



(c) EDF: response time for all the tasks, computed with constant $FoS = 0.5$

Figure 3.11: Probabilities of deadline misses for the tasks of the task set Γ_4 with EDF and FP.

Figure 3.11a and 3.11c depict the task response times in the CDF form for FP and EDF, respectively. Figure 3.11b and 3.12f depict the probability of deadline miss of all the jobs of τ_1 for FP and EDF, respectively. In there, we have plotted the probability of deadline miss with different *FoS*; the obvious dependence on the FoS does not apply to all the jobs. In particular, under EDF there are jobs for which the pessimism introduced by low FoS does not reflect into large probability of deadline miss. For FP,



(a) Jobs of task τ_1 (b) Jobs of task τ_2 (c) Jobs of task τ_3 (d) Jobs of task τ_4

(e) Jobs of task τ_5 (f) EDF: probability of deadline miss for all the jobs of τ_1 for different FoS (g) All the tasks

Figure 3.12: Proabilities of deadline misses for the jobs given by the task set Γ_4.

since τ_1 has the highest priority, the jobs are not at all affected by *FoS*. Lower priority jobs shows obvious pessimism by *FoS*, it still does not apply to all the jobs of a task.

Figure 3.11 compares the probability of deadline miss of all the jobs and tasks. $FoS = 0.5$ for all the cases here. It is noticeable how the dominance of EDF in deterministic real-time systems and deterministic WCETs, is not confirmed with pRTS and continuous pWCETs. There exist jobs for which the probability of deadline miss is larger under EDF than under FP, in particular for τ_1, Figure 3.12a and Figure 3.12g. In deterministic case, EDF is proved to be better than FP in terms of maximum possible utilization with maximum EDF utilization being 1 and that of FP being 0.693. In the probabilistic case, this cannot be deemed a deciding factor while choosing a scheduling policy.

3.7 Perspectives

With this work, we used CTMCh for modelling and analyzing pRTSs with continuous pWCETs for task execution. A CTMCh model for each job in the task set has been built and properties formally extracted through model analysis. The models take into account the probabilistic behaviour of the jobs and the probabilistic interference they subdue to. From all the models, a safe probabilistic schedulability analysis has been developed for preemptive EDF and FP scheduling. It computes the probability of deadline miss and probabilistic response time of the jobs and the tasks. The complexity and the accuracy of this approach have been theoretically studied and evaluated with test cases.

pWCET is a probabilistic quantity which can be described through discrete or continuous distributions. pWCET described as continuous probabilistic distributions is a result of MBPTA timing analysis [CG+12]. On the other hand discrete pWCET is a result of SPTA timing analysis. Whenever

using either of the distributions, careful observation should be made in order to use an appropriate modelling method in order to ensure that no information is lost. This is explained as follows.

When many tasks execute, periodically or aperiodically, each task has an arrival time and a deadline. Since each task is described with a pWCET distribution, in real-time, the pWCET starts at the time of arrival and ends at the deadline. Thus, when task executions are scheduled such that they overlap (like parallel releases), one task can probabilistically delay the execution of the other in a uniprocessor machine. This information of probabilistic delay is contained in the pWCET. But due to arrival times and deadline, the pWCET distributions overlap only partially. This phenomenon poses certain challenges when pWCET distribution is continuous.

First, if we use continuous pWCET distributions they need mathematical representation. This can be done as exponential, Pareto, Gaussian, etc. distributions. These continuous pWCET distributions are defined in the interval $[0,\infty)$. In the real-time context, arrival times and deadlines impose a threshold to these distributions which are less than infinity. Thus, there is a potential risk of loosing probabilistic information in the distribution beyond the deadline. Second, the difference in arrival time implies a shift when the pWCET distribution actually 'begins', By the term 'begin', we mean the time after which the probabilistic affect of a pWCET on other tasks starts to come into play. This is what we have called 'backlog' in this chapter. Now, classically the backlog is evaluated simply by convolving. However, the convolution operation does not take into account this shift due to arrival times and the deadlines. Therefore, using continuous distribution requires careful handling, and possibly pessimistic approaches, to ensure that no information is lost during any operation.

We can argue that we should convert a continuous pWCET distribution to a discrete pWCET distribution. This requires a deeper understanding. Mathematically, continuous and distributions represent probability in a different manner. Probability for a single value of the random variable as given by discrete probability can be non zero but it is always zero when extracted from continuous distribution. In the real-time context, we must proceed as follows. The continuous pWCET distribution must first be discretized for the values of execution times. For each value, the probability at the point must be the area of the continuous distribution between itself and the discrete point before.

Through the above method of discretization, the probability in an interval of the continuous distribution is placed at the upper value of the interval. This means, the worst case probability that the task takes the execution time is given as the summation of the probabilities in the interval preceding that. The value is worst case because the probabilities are added and placed at the discrete value on the upper limit. This is shown in Figure 3.13. For a job J with deadline d given with a continuous pWCET PDF $f_{continuous}$, the discrete PDF $f_{discrete}$ is given as:

$$f_{discrete}(x) = \int_{x-\delta}^{x} f_{continuous}(x); 0 \leq x \leq d$$

where δ is the discretization interval.

On the other hand, we do not have to consider the probability at each execution time when using a continuous distribution. This is unlike the discrete case where the probability at each point is given. Thus, instead of computing and convolving the matrices as in the discrete case, the continuous distributions can be evaluated using the parameters of their mathematical representations. For example,

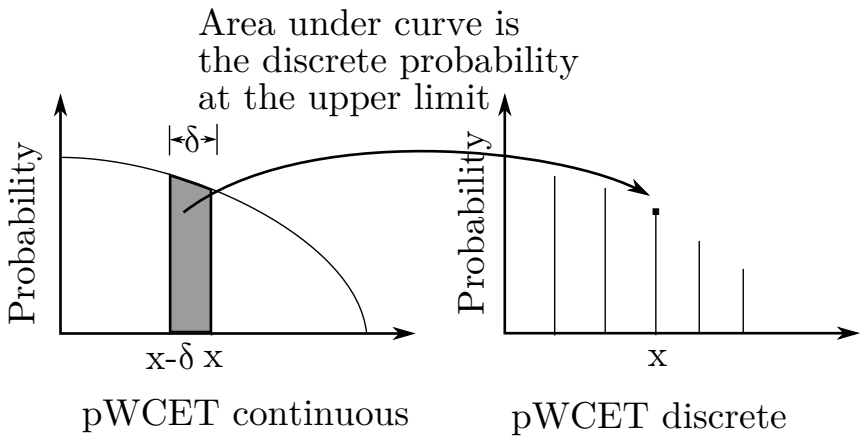


Figure 3.13: Continuous to Discrete pWCET

the continuous exponential distribution can be convolved by evaluating their rates λ while discrete convolution is classical matrix multiplication. This way using continuous distributions have a potential to reduce the complexity of the schedulability analysis.

At this point we would like to highlight a potential problem with a common approach towards probabilistic modeling which uses Monte Carlo method. Monte Carlo method is based on making random choices and feeding them to a system model and then measuring the output. When this method is applied for real-time systems, it can lead to an unsafe analysis. The method is based on generating a sequence of random inputs. Generally, this random sequence can come from a pseudo-random number generator or a physical source of randomness. Through physical source of randomness, it is difficult to ensure that all the possibilities of the system can be explored. On the other hand, a pseudo-random number generator has a period. This implies, the random numbers sequence generation will repeat after a certain point. Because of this, it cannot be guaranteed that all the possible scenarios of the system will be explored even if if it is fed with numbers from pseudo-random number source infinitely. Thus, Monte Carlo method cannot be deemed safe for probabilistic real-time system analysis as functionality at the worst cases cannot be ensured.

We also observe the following, the pessimism involved in the analysis of the real-time system is dependent on the schedule. The pessimism or the over-approximation of the associated probabilities relies on the assumption that the tasks can take longer to finish execution even if the execution time is less. This characteristic is exhibited in the case where jobs of different tasks are released asynchronously. As mentioned earlier, the objective of the probabilistic approach is to quantify the pessimism involved in the timing analysis of the tasks. In this chapter, we scheduled the tasks with pWCETs through an existing scheduling methodology like EDF or FP. Then we proceeded to quantify the task response times. To obtain the response times, we are forced to stick to the task execution order as given by the scheduling algorithms. Thus, any pessimism that results from backlog estimation operations depends on the schedule. Therefore, a more correct way of approaching the probabilistic modelling is to obtain a schedule itself. This way, the probabilities can come into picture which can be subject to a minimization or a maximization criteria. At the moment, the probabilities are only quantified. We will approach this in the Chapter 6.

Here, we must also observe the obvious. The probability of an occurrence exists each time a trial is made. In the real-time context, the probability of a task missing its deadline exists each time it executes. The affect of this is that we lose predictability of the system. Each time the task executes, we only know a probability of it missing its deadline. We are not prepared to act suitably and ensure safety when the task actually does miss its deadline. This observation calls for a modelling method where the probabilities are quantified as well as upper bounded. One step further, the method should be able to cope with a task missing its deadline, howsoever improbable it may be. We will approach this in the Chapter 5 and 6.

In the next chapter, we approach another method of quantifying pessimism in the real-time systems which is through Mixed Criticality (MC). As we will see, the MC method contains tasks of various criticality. The higher criticality tasks are defined with two WCETs, one lower than the other. The higher WCET corresponds to the task taking more time to execute. The system is then managed to accommodate a task taking more time. Later we will combine these approaches as well as the observation to obtain a schedule instead of using the existing ones in order to fully utilize the potential of MC ans probabilistic approaches.

CHAPTER 4

Discrete Time Markov Chain Mixed Criticality Schedulability Analysis

---

> We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths.
>
> *Walt Disney*

In this Chapter we move forward with a different solution towards reducing pessimism in the pRTS which is through Mixed Criticality. We pose the following question.

Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the probability that the system enters high criticality mode?

At this point we change our direction from analysis of classical pRTS to a mixed criticality pRTS. Let us recall that a mixed criticality real-time system is one which contains some tasks which are of higher importance or criticality than the others. The high criticality tasks have two worst case execution times, one for lower criticality mode and the other for higher criticality mode which is larger. If the task is not finished by the lower criticality WCET, the mode switch is triggered and more time is allowed for execution. The high criticality tasks must be ensured schedulable in the system high criticality mode. The other lower criticality tasks are allowed to execute in the time between the two WCET values and are dropped when that time is allowed to the higher criticality task. This way the most important tasks are ensured execution and the remaining time is available for other tasks to use. This chapter dives deeper into this idea.

We bring the notion of the task criticality to the job level by assigning each job its criticality. In the two-criticality-level case, each job is designated as being of either higher criticality HI-criticality or lower criticality LO-criticality. The HI-criticality mode is where the job executes in highly critical (and more demanding) conditions – critical function or fault recovery; a LO-criticality mode is the nominal working condition for the job where it executes in normal conditions. Having a higher criticality is regarded as giving more execution time to the task.

We define the job criticality based on response time. The choice is application oriented as the job entering high criticality happens at run-time. This is a direct result of the high criticality task demanding more time to execute. Because the notion of criticality is at job level, we propose a method to selectively drop lower criticality jobs whenever required.

We proceed as follows by appending mixed criticality notation . A HI-criticality job J^{HI} is the tuple: $J^{\text{HI}} \stackrel{def}{=} (\mathcal{C}, a, d, p, l, \chi)$. $\mathcal{C}$, a, p and d are as defined earlier. χ is the job criticality level defined for a job J [Bar+15] which can take two values at runtime: HI and LO; $\chi = \{\text{HI}, \text{LO}\}$. $l \leq d$ describes the threshold with which we define the job criticality mode.

A LO-criticality job J^{LO} is the tuple: $J^{\text{LO}} \stackrel{def}{=} (\mathcal{C}, a, d, p, \chi)$. χ for a LO-criticality job can take only one value, $\chi = \{\text{LO}\}$.

For the jobs, the criticality level of its task is inherited. However, the actual criticality mode of the jobs can change at runtime depending on their scheduling, in turn affecting the system criticality mode.

The real-time application is formed from these tasks which are partitioned between their HI-criticality jobs and LO-criticality jobs. $\Gamma^{\text{HI}} = \{J^{\text{HI}}\}$ is the set of high criticality jobs with n^{HI} number of HI-criticality jobs; $\Gamma^{\text{LO}} = \{J^{\text{LO}}\}$ is the set of LO-criticality jobs with n^{LO} the number of LO-criticality jobs; $\Gamma = \Gamma^{\text{HI}} \cup \Gamma^{\text{LO}}$ and n is the total number of jobs in the hyperperiod, $n = n^{\text{HI}} + n^{\text{LO}}$. With tasks, it is m^{HI} the number of HI-criticality tasks, and m^{LO} the number of LO-criticality tasks; $m^{\text{HI}} + m^{\text{LO}} = m$.

Let us recall that Markov Chain is a set of states and transitions with the Markov Property. Discrete Time Markov Chain is one in which the transitions are labelled with the probability of being chosen such that the sum of all outgoing probability from a state is one.

Dans ce chapitre, nous allons de l'avant avec une solution différente pour réduire le pessimisme dans le pRTS, qui passe par la criticité mixté. Nous posons la question suivante.

Étant donné un système temps réel probabiliste à criticité mixté avec des exécutions de tâches décrites avec pWCET, quelle est la probabilité que le système passe en mode de criticité élevée?

À ce stade, nous changeons notre direction de l'analyse de pRTS classique à un pRTS à criticité mixté. Rappelons qu'un système en temps réel à criticité mixté est un système qui contient des tâches plus importantes ou plus critiques que les autres. Les tâches à criticité élevée ont deux temps d'exécution dans le pire cas, l'un pour le mode de criticité inférieure et l'autre pour le mode de criticité plus élevée qui est plus grand. Si la tâche n'est pas terminée par le WCET de moindre criticité, le changement de mode est déclenché et plus de temps est accordé pour l'exécution. Les tâches à haute criticité doivent être assurées dans le mode de haute criticité du système. Les autres tâches de moindre criticité sont autorisées à s'exécuter dans l'intervalle de temps entre les deux valeurs WCET et sont supprimées lorsque ce délai est autorisé à la tâche de criticité plus élevée. De cette façon, l'exécution des tâches les plus importantes est assurée et le temps restant est disponible pour d'autres tâches. Ce chapitre approfondit cette idée.

Nous apportons la notion de criticité des tâches au niveau du job en attribuant à chaque job sa criticité. Dans le cas de deux niveaux de criticité, chaque job est désigné comme étant d'une criticité plus élevée HI-criticité ou d'une criticité inférieure LO-criticité. Le mode HI-criticité est l'endroit où le job s'exécute dans des conditions très critiques (et plus exigeantes) - fonction critique ou reprise après incident; un mode de criticité LO est la condition de job nominale pour le job où il s'exécute

dans des conditions normales. Une criticité plus élevée est considérée comme donnant plus de temps d'exécution à la tâche.

Nous définissons la criticité du job en fonction du temps de réponse. Le choix est orienté vers l'application, car le job entrant dans une zone de forte criticité se produit au moment de l'exécution. Ceci est le résultat direct de la tâche de haute criticité exigeant plus de temps pour s'exécuter. Parce que la notion de criticité se situe au niveau du job, nous proposons une méthode pour supprimer sélectivement les jobs de moindre criticité chaque fois que cela est nécessaire.

Nous procédons comme suit en ajoutant une notation de criticité mixté. Un job de HI-criticité J^{HI} est le tuple: $J^{\text{HI}} \stackrel{def}{=} (C, a, d, p, l, \chi)$. C, a, p et d sont tels que définis précédemment. χ est le niveau de criticité du job défini pour un job J [Bar+15] qui peut prendre deux valeurs au moment de l'exécution: HI et LO; $\chi = \{\text{HI}, \text{LO}\}$. $l \leq d$ décrit le seuil avec lequel nous définissons le mode de criticité du job.

Un job de lo-criticité J^{LO} est le tuple: $J^{\text{LO}} \stackrel{def}{=} (C, a, d, p, \chi)$. χ pour un job de lo-criticité ne peut prendre qu'une seule valeur, $\chi = \{\text{LO}\}$.

Pour les emplois, le niveau de criticité de sa tâche est hérité. Cependant, le mode de criticité réel des travaux peut changer au moment de l'exécution en fonction de leur planification, affectant à son tour le mode de criticité du système.

4.1 Mixed Criticality System

The MC definitions and modelling we propose, is slightly different than the classical ones with multiple WCET thresholds [Ves07]; [GSY15]. With the MC modeling via pWCRT, it is possible to distinguish the job behaviour at runtime, which, otherwise impossible to do with pWCETs and WCET thresholds. This allows to relate the system criticality to the actual job execution which includes the job waiting time due to preemptions and postponements. Nonetheless, the MC analysis we propose is general enough to apply to WCET thresholds also. Note that in the latter case, every job of the same task would have the same criticality mode.

Criticality threshold: We define the job criticality mode using a threshold $l \leq d$ which applies to the job pWCRT. As shown in Figure 4.1a with HI and LO criticality regions, if the job finishing time is in $[l, \infty)$, the job is considered to execute in high criticality mode, otherwise the job executes in low criticality mode, $[0, l)$.

The probability P^{HI} that a job J executes in the high criticality mode is:

$$P^{\text{HI}} \stackrel{def}{=} \int_{l}^{\infty} f_{\mathcal{RT}}(x)dx. \tag{4.1}$$

If the pWCRT is discrete as shown in Figure 4.1b with WCET as the maximum possible execution time, $P^{\text{HI}} \stackrel{def}{=} \sum_{x \geq l}^{WCET} f_{\mathcal{RT}}(x)$.

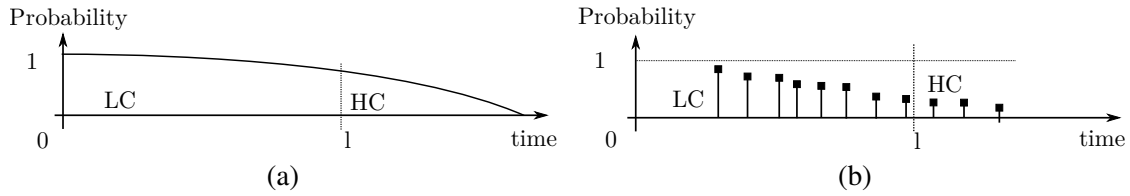


Figure 4.1: pWCRT of job J in its ICDF in (a) continuous and (b) discrete form . High and low criticality regions are separated by l. Low and High criticality zones denoted as LO and HI.

Classically, the definition of system criticality is such that: the system enters high criticality mode whenever at least one of the HI-criticality job enters high criticality mode [Bar+15]. The schedule of the high criticality jobs in the system high criticality mode is ensured. We propose the following more generic and flexible definition of system criticality. Given total of n^{HI} number of HI-criticality jobs:

Definition 6 ((k,n) System criticality)**.** *The system criticality level χ is high (*HI*) if at least k^{HI} out of n^{HI}* HI*-criticality jobs enter high criticality mode.*

$n^{\text{HI}} \leq n$ where n is the total number of jobs in the hyperperiod. Using the above definition for system criticality allows the flexibility to choose the value of k^{HI} depending on the system. This also implies that a k^{HI} greater than 1 is less pessimistic than the classical definition of system criticality. Because of the nature of the pRTSs, the event of the system entering high criticality mode is not deterministically known anymore: there exists a probability of the system entering the high criticality mode. This is why we need to use reliable probabilistic analysis tools to analyze such a system.

4.2 Discrete Time Markov Chain model

We assume that a task set is given, it is scheduled using EDF scheduling policy, and the pWCRT for each job in the hyperperiod is known. The system criticality modes are modelled as a Discrete Time Markov Chain (DTMCh). The choice of DTMCh makes it possible to simply arrange the readily available probabilities from the schedulability analysis. System depiction as states replicates the switching the real-time system between high the low criticality modes. The transitions between those states can be labelled with the probability of it being chosen. Moreover, DTMCh allows modelling of a probabilistically distributed system subject to its mathematical foundation. This is unlike a sequential system, like Petri net or automata, where discrete actions form a complex explorable tree. DTMCh is subject to formal model checking in which path properties or the probability of reaching certain states can be formally checked. As we will see, this is useful to know the probability of a path taken by the system. A DTMCh M is defined as a set of states S and state transitions given by a Q-matrix Q, $M = (S, Q)$ [Put94]. In the following, we give the basics to build DTMCh for mode changes in MC pRTSs and the probabilistic properties which are formally verified with PRISM Model Checker [KNP11]. It should be noted that property verification for a probabilistic system returns a probability of that property being true.

The DTMCh model is given as $M = (S, Q)$. To build M it is required to consider only the HI-

criticality jobs in this DTMCh. It is because only HI-criticality jobs contribute to decide the system criticality. The contribution of the LO-criticality jobs is included in the pWCRTs of the HI-criticality jobs. For each job $J^{\text{HI}} \in \Gamma^{\text{HI}}$, the set of states $S = \{J^{\text{HI}}, HC, LC\}$ is defined. State HC represents execution of J^{HI} in high criticality mode ($[l, \infty)$) and state LC represents execution of J^{HI} in low criticality mode ($[0, l)$). The state J^{HI} is simply a passing state used for the ease of modeling and representation; it has no contribution to the analysis. The set of states S for the whole system is defined as the union of the sets S for all the jobs J which are HI-criticality jobs: $S \stackrel{def}{=} (\bigcup S) : \forall J \in \Gamma^{\text{HI}}$. S is ordered in the increasing priorities of the jobs that it contains, for the ease of formalization and presentation. Any other ordering would be possible, since the DTMCh does not represent the scheduling but only the criticality configurations.

The set of states and transitions in M is shown in Figure 4.2 for an example task set Γ. There are unidirectional transitions $J^{\text{HI}} -> HC$, $J^{\text{HI}} -> LC$, $HC -> J'^{\text{HI}}$, $LC -> J'^{\text{HI}}$; such that $p' > p$ and the priority difference $|p - p'|$ is minimum, $J', J \in \Gamma^{\text{HI}}$. The initial state of the DTMCh is a job executing state J such that p is minimum amongst all $J \in \Gamma^{\text{HI}}$.

The state transitions are labelled such that the sum of the probabilities of all the outgoing transitions is equal to one. Each transition emanating from a state J to HC is labelled with the probability P^{HI} from Equation (4.1). This implies, each transition emanating from a state J to LC is labelled with the probability $1 - P^{\text{HI}}$. The transition matrix Q defined as:

$$
\begin{bmatrix}
 & J_1 & HC_1 & LC_1 & \dots & J_{n^{\text{HI}}} & HC_{n^{\text{HI}}} & LC_{n^{\text{HI}}} \\
J_1 & 0 & P_1 & 1-P_1 & \dots & 0 & 0 & 0 \\
HC_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
LC_1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
J_2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
HC_2 & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
LC_2 & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
J_{n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & P_{n^{\text{HI}}} & 1-P_{n^{\text{HI}}} \\
HC_{n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
LC_{n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & 0 & 0
\end{bmatrix}
\tag{4.2}
$$

To refer to an element of the matrix, $Q(State_r, State_c)$ gives the probability of transition from a state $State_r$ in the row to a state $State_c$ in the column, e.g. probability of transition from J_1 to HC_1 is referred as $Q(J_1, HC_1)$, equal to P_1.

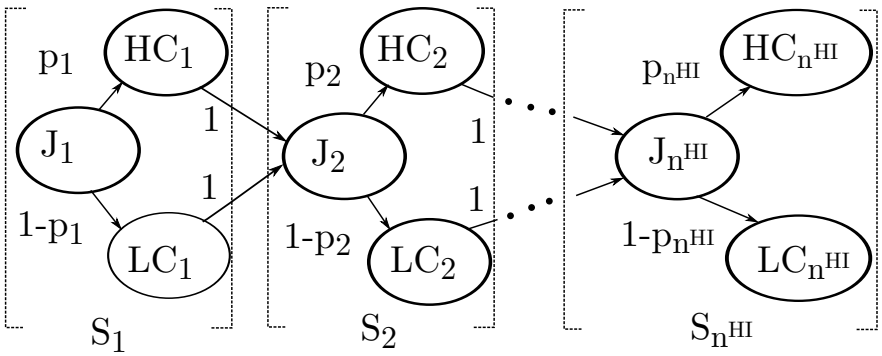


Figure 4.2: System DTMCh model M, assuming $J_1, J_2, \dots, J_{n^{\text{HI}}} \in \Gamma^{\text{HI}}$ and such that $p_1 \leq p_2 \leq \dots \leq p_{n^{\text{HI}}}$.

As we see, the DTMCh models construction is quite straightforward without involving any math-

ematical operations like convolution or upper-bounding of any kind. It is simply arranging the probabilities obtained from the schedulability analysis into formally verifiable DTMCh structure. Thus, the safety of this construction comes the safety of the given schedulability analysis used to obtain the pWCRTs.

4.3 Analysis

The DTMCh model $M = (S, Q)$ is exploited here to extract probability of system entering high criticality.

4.3.1 Criticality analysis

With DTMCh $M = (S, Q)$ Matrix (4.2), we define the meaning of system criticality from the criticality levels of the jobs. Then, we quantify the probability of the system criticality by exploring M with formal model checking. The definition of system criticality level translates into paths within the DTMCh taken by the system through certain states. Each path has a probability of being taken at runtime. This is a possibility for the system entering the high criticality mode.

A path in M represents the trace of the jobs taking high or low criticality modes at runtime.

Definition 7. *A path D is an ordered set of states* $D \stackrel{def}{=} [State_1, State_2, ...State_{n^{\mathrm{HI}}}]$*, such that* $Q(State_k, State_{k+1}) > 0$ *i.e. there exists a transition between two consecutive states.*

The probability of occurrence of the path D is $Pr(D)$, and it is computed by performing model checking on DTMCh using the property: 'the maximum probability that the next state is $State_1$ AND the next to next state is $State_2$ AND the next to next to next...'. This property is formally written as:

$$Pmax = ?[X\ state = State_1\ \&\ XXX\ state = State_2\ \&XXXXX\ state = State_3 \ldots]$$

It should be noted that for each state, the next (X) is considered one from the initial state. Doing so defines a model checking property which navigates through the states. This is done in accordance with the model checker that we use.

For M, there are $2^{n^{\mathrm{HI}}}$ possible paths which start from the initial state. Examples of a path from the model in Figure fig:DTMC1 are: $D_1 = [J_1, HC_1, J_2, LC_2 \ldots, LC_{n^{\mathrm{HI}}}]$,
$D_2 = [J_1, LC_1, J_2, HC_2, \ldots, HC_{n^{\mathrm{HI}}}]$.

(k,n) system criticality: Here, we recall the Definition 6 that system is said to be in high criticality mode if at least k^{HI} out of n^{HI} jobs enter high criticality mode. The system enters high criticality mode in paths in which there is more than or equal to k^{HI} high criticality states HC. Such paths are denoted by the superscript kn.

The q-th path D_q^{kn} has a probability of occurrence $Pr(D_q^{kn})$. There are q_k paths which pass through a minimum of k_{HI} high criticality state. The exact value of q_k follows the mathematics of partitioning of

	$\mathcal{C}$	$T_i = D_i$	χ_i
τ_1	$EXP(12)$	3	$\{\text{HI}, \text{LO}\}$
τ_2	$EXP(13)$	5	$\{\text{HI}, \text{LO}\}$
τ_3	$EXP(15)$	6	LO
τ_4	$EXP(18)$	10	LO
τ_5	$EXP(15)$	10	LO
τ_6	$EXP(16)$	15	LO

Table 4.1: Task set Γ parameters with pWCET as exponential distribution given with the rate parameter.

numbers, which is left for future discussions. Now, using Definition 6 system enters the critical region if, D_1^{kn} occurred OR D_2^{kn} occurred OR ... $D_{q_k}^{kn}$ occurred. Thus, the probability P^{kn} that the system enters high criticality region is:

$$P^{kn} \stackrel{def}{=} Pr(D_1^{kn}) + \ldots + Pr(D_{q_k}^{kn}) = \sum_{q=0}^{q_k} Pr(D_q^{kn}). \tag{4.3}$$

As special cases, (1,n) and (n,n) system criticality are the two extremes of the (k,n) definition. (1,n) is to say that the system is in high criticality mode if at least one HI-criticality job is in high criticality mode: $P^{1n} \stackrel{def}{=} Pr(D_1^{1n}) + \ldots + Pr(D_{2^n-1}^{1n}) = \sum_{q=1}^{2^n-1} Pr(D_q^{1n})$.

Where D_1^{nn} is the path taken through all the high criticality states.

Example 1. *For this and the next section, we assume a task set* Γ *as shown in Table 4.1. All the timing parameters are in unit time and the* $\mathcal{C}$ *are exponential distributions with rate* $RATE$*, represented as* $EXP(RATE)$*.* Γ *is composed of* 6 *tasks with a total of* 31 *jobs. Tasks* τ_1 *and* τ_2 *are* HI*-criticality tasks, and tasks* τ_3*,* τ_4*,* τ_5 *and* τ_6 *are* LO*-criticality tasks. The threshold* $l = 0.7D$*.* Γ *is scheduled using preemptive EDF, the jobs are aborted at the deadline. The hyperperiod of this task set is* 30 *time units; in it, there are a total of* 16 HI*-criticality jobs and* 13 LO*-criticality jobs executing. The task set* Γ *from Example 1 is applied to this methodology for different* k_{HI}*.*

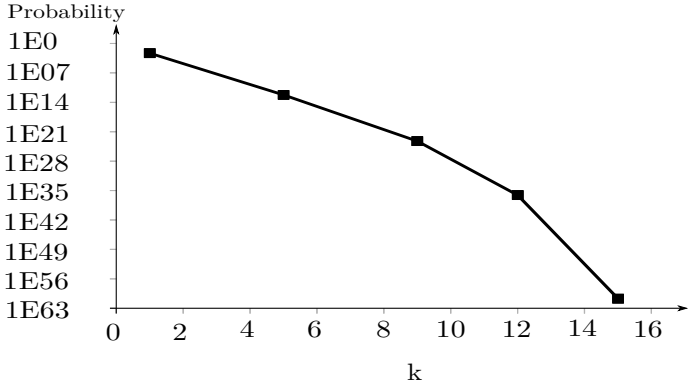


Figure 4.3: P^{kn} from (k,n) system criticality.

Figure 4.4 shows the value of P^{kn} *computed with Equation (4.3) as the value of k changes from the* (k,n) *system criticality definition Definition 6 . The probability that system enters* HI*-mode: using (1,n)*

is $6.45E-03$ *and using (n,n) is* $2.70E-62$.

4.3.2 Deadline Miss Analysis

This section focuses on the deadline miss probability of HI-criticality jobs. The deadline miss probability of a job J should be less than or equal to a given probability $P^{dm,max}$. Here $P^{dm,max}$ is assumed to be given and is the requirement to meet in order to guarantee the probabilistic schedulability. Here, the focus is on the job to reduce its probability of deadline miss.

Usually, MC scheduling directs that when the system is in high criticality mode, all the LO-criticality jobs are dropped to ensure the timing requirements of the remaining HI-criticality jobs. Instead, the scheduling algorithms we propose acts by selectively dropping LO-criticality jobs whenever the deadline miss probability constraint is not met. It is conceived to minimize the number of LO-criticality jobs to drop allowing some of the LO-criticality jobs executing with HI-criticality jobs. This way, the computational resources are better used and the scheduling of HI-criticality tasks is not jeopardized. We present one such offline method below.

Job strategy: The job strategy is the scheduling algorithm we propose to reduce the probability of deadline miss of a HI-criticality job without dropping all the LO-criticality jobs in the system. Classically, all the LO-criticality jobs are dropped in the high criticality mode. A choice to drop a single job in the ordered list of jobs requires complete re-evaluation of the whole system to prove optimality. This is because the response time depends on the execution of the previously executed jobs. The strategy to choose needs to ensure the optimality as well as the safety of the resulting schedule. Dropping a single job will lead to an analysis of all possible execution sequences of the rest of the jobs to find an optimal one. The complexity of doing so for every job to prove optimality and safety is $O(n^n)$, where there are n jobs in the hyperperiod. Such complexity does not include the complexity of the probabilistic schedulability analysis applied; it is only for exploring all the jobs. What we present here is not an optimal strategy to maximize the deadline miss probability reduction per job dropped. However, it is better than classical MC scheduling strategies in which all the LO-criticality jobs are dropped whenever the system enters high criticality mode.

Interference isolation: The set of LO-criticality jobs $J^{over}(J^{\mathrm{HI}})$ which overlap to the execution of a job J^{HI} is:

$$J^{over}(J^{\mathrm{HI}}) \stackrel{def}{=} \{J' : p' < p, d' > a\}; J' \in \Gamma^{\mathrm{LO}}. \tag{4.4}$$

The jobs in this set directly impose a probabilistic delay/backlog in the execution of J^{HI}, as depicted in the Figure 4.4 by LO-criticality jobs J', J'' and J'''. All the jobs in $J^{over}(J^{\mathrm{HI}})$ impose an indirect backlog to J^{HI}. A certain amount of job backlog is passed in their order of priority and thus indirectly to the job J^{HI}. The term 'interference isolation' refers to the separation of J^{HI} from the indirect backlog.

Lemma 2 (Backlog isolation)**.** *The backlog for a* HI*-criticality job* J^{HI} *reduces by the maximum amount if all the* LO*-criticality jobs in* $J^{over}(J^{\mathrm{HI}})$ *from Equation (4.4) are dropped, given that the jobs are suspended at their respective deadlines.*

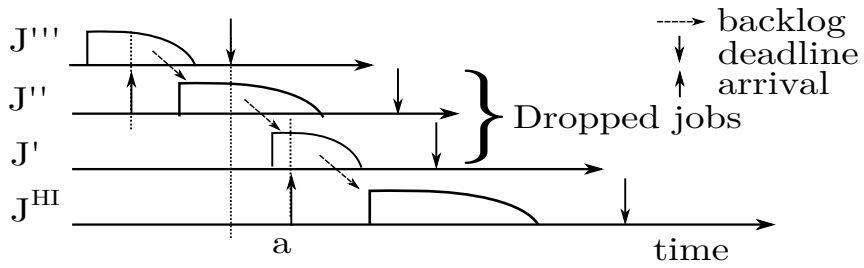


Figure 4.4: The backlog to the HI-criticality job J^{HI} reduces to zero by dropping LO-criticality jobs J' and J''.

Proof. With continuous distributions defined in $[0,\infty)$, a job imparts a probabilistic backlog after executing to next jobs until its deadline. Since the jobs are suspended at deadline, the jobs with deadline before the arrival time of J^{HI} will not impart any direct backlog. For the job J^{HI}, to ensure a maximum reduction in deadline miss probability, all the jobs which impart backlog must be dropped from execution. All such jobs are contained in the set $J^{over}(J^{\text{HI}})$. It is the best possible effort to reduce the probabilistic backlog to J^{HI}. Thus, the backlog to J^{HI} from all the preceding jobs is minimized, once all the jobs in $J^{over}(J^{\text{HI}})$ are dropped. □

The job-level scheduling strategy we propose reduces the probability of deadline miss of a HI-criticality job by dropping all the LO-criticality jobs in $J^{over}(J^{\text{HI}})$. Lemma 2 proves that dropping all those jobs ensures the maximum possible deadline miss probability reduction for J^{HI}. Thus, the strategy is:

- Identifying $J^{over}(J^{\text{HI}})$;
- Dropping all the jobs in $J^{over}(J^{\text{HI}})$.

Referring to the Figure 4.4, the backlog to the HI-criticality job J^{HI} minimizes by dropping jobs J' and J'': there is no effect from the job J''' to the job J^{HI} because J''' suspends at the deadline. The HI-criticality job in observation still retains its own execution after dropping the jobs J'' and J' of the set $J^{over}(J^{\text{HI}})$, that is after removing maximum interferences. The P^{dm} obtained once dropping all the jobs in $J^{over}(J^{\text{HI}})$, is the best (minimum) deadline miss probability we can achieve for J^{HI}. If it is not enough to meet the constraint $P^{dm,max}$, P^{dm} is still larger than $P^{dm,max}$, the problem for this job is unsolvable.

Example 2. *The task set Γ from Example 1 is analyzed. Figure 4.5 shows the probability of deadline miss P^{dm} for* HI*-criticality jobs J_{26}, J_{14} and J_{24} when no* LO*-criticality job is dropped as $\{\emptyset\}$. Their corresponding set $J^{over}(J^{\text{HI}})$ is shown and the jobs in it are dropped. The reduced value P^{dm} for each job is also shown. We use the CTMCh model to obtain the probability of deadline miss of the jobs from Chapter 3.*

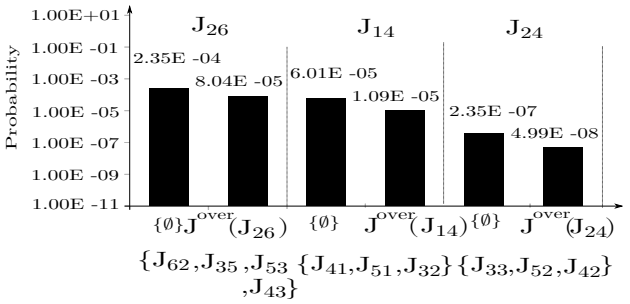


Figure 4.5: Probability of deadline miss of the job J_{14} vs when the LO-criticality jobs dropped.

4.4 Perspectives

With this work, we have proficiently applied probabilistic formal methods (DTMCh) to model and analyze MC pRTSs. The pWCRT is used to define the MC behaviour of the jobs with a threshold parameter (l) to distinguish between low criticality mode and high criticality mode for HI-criticality jobs. The DTMCh representations of the runtime behaviour are studied to quantify the probability of the system entering HI-criticality mode. DTMCh is also used to obtain deadline miss probability of HI-criticality jobs, and it can be easily extended to tasks in the future works. We have proposed method to selectively drop LO-criticality jobs in order to achieve the required maximum probability of deadline miss. This study does not yet focus on the scalability. They are enhancements to existing MC schedulability that allow for a better use of the computational resource by reducing the number of dropped LO-criticality jobs.

Here again we observe that we have same hurdles as in Chapter 3. We see that the probabilities that arise in the system are from the pWCET. Since pWCET does not depend on the schedule of the tasks, nor do the probabilities that are computed from them. Therefore, the probability can only be quantified and not be used to make scheduling decisions. For the same reason, the schedule does not affect the probability of the system entering high criticality. The schedule only affects the response time of the tasks and not the pWCETs. So, we need to be free of the classical scheduling methods and obtain a schedule that fits our global object of quantifying and minimizing the pessimism.

At the same time, the probabilities that are quantified exist each time there is a trial. That is, each time the system is executed, the probability that it can enter high criticality mode exists. While this information gives us a global picture of the system, it is not sufficient to deduce the correct action when the system does enter high criticality mode. In order to do so, we need to obtain a system model which is more granular in representation of job executions. That is, the model should be able to represent the job executions at any given time and be able to foresee the appropriate actions whenever a job enters higher criticality.

Therefore, we need an approach which uses the MC as well as the probabilistic approaches. We need a model which is able to do both, quantify the probabilities as the global property of the system; and upper bound these probabilities to be prepared whenever the system does enter high criticality, as seen in the next chapter. In the later chapter, Chapter 6, we bring probability in the decision making process.

CHAPTER 5

Graph Based Mixed Criticality Schedulability Analysis: Execution Time

---

> The true sign of intelligence is not knowledge but imagination.
>
> *Albert Einstein*

At this point in the thesis research, we imagine a new solution to solve the global issue of pessimism. We approach MC and probabilistic models together. We do not adhere to classical scheduling algorithms. At the same time, we do not adhere to certain existing MC norms. We formalize the problem as follows.

Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the optimal schedule when the system enters high criticality mode?

In this chapter we continue the notion of mixed criticality probabilistic real-time system. Earlier, we performed schedulability analysis on existing scheduling algorithms, like EDF and FP. Now, we approach the problem of finding the schedule itself in a MC pRTS.

As we saw in the earlier chapter, we can model probabilistically and obtain the probability of something occurring within the system. However, we need a model which can provide a more precise picture of those occurences. For example, the probability that a task enters higher criticality exists each time the task is executed. When the task actually does enter high criticality, how should the scheduling cope with that. Since scheduling is a sequence of jobs, using graphs provides a way to model such sequences. From those graphs, necessary schedulability information is extracted.

In this chapter, we consider three criticality system as LO, MI and HI. The jobs belonging to those criticalities are defined as follows. A HI-criticality job is:

$$J \stackrel{def}{=} (\{(C^{\mathrm{LO}}, C^{\mathrm{MI}}, C^{\mathrm{HI}}), (\mathcal{P}^{\mathrm{LO}}, \mathcal{P}^{\mathrm{MI}})\}, a, d, \mathrm{HI})$$

a MI-criticality job is:

$$J \stackrel{def}{=} (\{(C^{\mathrm{LO}}, C^{\mathrm{MI}}), (\mathcal{P}^{\mathrm{LO}})\}, a, d, \mathrm{MI})$$

a LO-criticality job is:

$$J \stackrel{def}{=} (\{(C^{\mathrm{LO}})\}, a, d, \mathrm{LO})$$

where a is the job arrival time, d is its deadline, C^{LO}, C^{MI} and C^{HI} are WCET values for the correspond-

ing criticality and $\mathcal{P}^{\text{LO}}$ and $\mathcal{P}^{\text{MI}}$ are the probabilities of the job exceeding LO and enter MI or exceeding MI to enter HI criticality, respectively.

As the jobs execute, any job can take more time to finish execution and thus enter higher criticality mode. This is only 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 directions which can be taken by the system. Then, through graphs we can perform explorations in a space of all possible schedules and aim at optimality.

À ce stade de la recherche de thèse, nous imaginons une nouvelle solution pour résoudre le problème globale de la these du pessimisme. Nous abordons ensemble les modèles MC et probabilistes. Nous n'adhérons pas aux algorithmes de planification classiques. Dans le même temps, nous n'adhérons pas à certaines normes MC existantes. Nous formalisons le problème comme suit.

Étant donné un système temps réel probabiliste à criticité mixte avec des exécutions de tâches décrites avec pWCET, quel est l'ordonnance optimal lorsque le système passe en mode de criticité élevée?

Dans ce chapitre, nous continuons la notion de système temps réel probabiliste à criticité mixte. Plus tôt, nous avons effectué une analyse d'ordonnancabilité sur des algorithmes d'ordonnancement existants, comme EDF et FP. Maintenant, nous abordons le problème de trouver le programme lui-même dans un MC pRTS.

Comme nous l'avons vu dans le chapitre précédent, nous pouvons modéliser de manière probabiliste et obtenir la probabilité que quelque chose se produise dans le système. Cependant, nous avons besoin d'un modèle qui puisse fournir une image plus précise de ces événements. Par exemple, la probabilité qu'une tâche entre dans une criticité plus élevée existe à chaque exécution de la tâche. Lorsque la tâche atteint réellement une criticité élevée, comment la planification doit-elle y faire face. La planification étant une séquence de travaux, l'utilisation de graphiques permet de modéliser ces séquences. À partir de ces graphiques, les informations de programmation nécessaires sont extraites.

Dans ce chapitre, nous considérons trois systèmes de criticité comme LO, MI et HI. Les emplois appartenant à ces criticités sont définis comme suit. Un job de HI-criticité est:

$$J \stackrel{def}{=} (\{(C^{\text{LO}}, C^{\text{MI}}, C^{\text{HI}}), (\mathcal{P}^{\text{LO}}, \mathcal{P}^{\text{MI}})\}, a, d, \text{HI})$$

un job de MI-criticité est:

$$J \stackrel{def}{=} (\{(C^{\text{LO}}, C^{\text{MI}}), (\mathcal{P}^{\text{LO}})\}, a, d, \text{MI})$$

un job de LO-criticité est:

$$J \stackrel{def}{=} (\{(C^{\mathrm{LO}})\}, a, d, \mathrm{LO})$$

où a est l'heure d'arrivée du job, d est sa date limite, C^{LO}, C^{MI} et C^{HI} sont des valeurs WCET pour la criticité correspondante et $\mathcal{P}^{\mathrm{LO}}$ et $\mathcal{P}^{\mathrm{MI}}$ sont les probabilités que le job dépasse LO et saisisse MI ou dépasse MI pour entrez HI criticité, respectivement.

Au fur et à mesure que les travaux s'exécutent, tout job peut prendre plus de temps pour terminer l'exécution et ainsi entrer en mode de criticité plus élevée. Cela n'est vrai que pour les travaux de criticité MI et HI. Cela nécessite une analyse précise pour déterminer quand cela peut se produire. De plus, le problème demande une optimalité, au moins dans l'utilisation des ressources. La signification exacte de l'optimalité sera présentée dans la section suivante. Dans tous les cas, l'optimalité globale ne peut être prouvée que lorsque toutes les possibilités sont explorées.

Dans cette section, nous proposons un modèle de système à criticité mixté basé sur une structure de graphe. Avec les graphiques, il est possible de représenter un réseau de directions possibles qui peuvent être prises par le système. Ensuite, grâce à des graphiques, nous pouvons effectuer des explorations dans un espace de tous les possibilites d'exécution des tâches et viser l'optimalité.

5.1 Graph and Tree Model

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. After the definition of a graph, trees are derived from the graph model which form a forest to be explored. This tree model will be exploited in Section 5.3 to build the scheduling algorithm.

5.1.1 Graph structure and construction

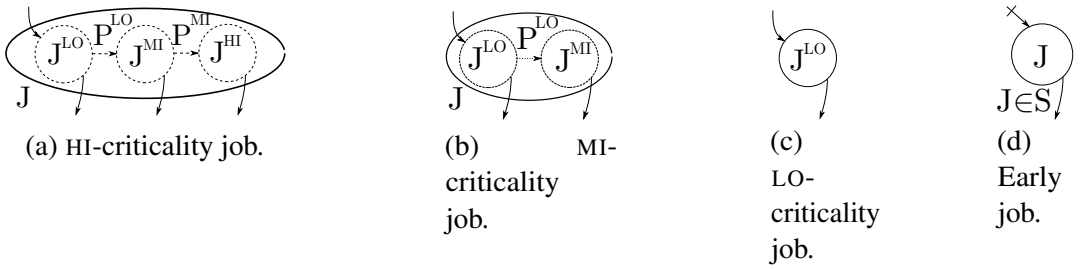
The nodes of the graph represent the execution of the jobs in a hyperperiod; the arcs correspond to the possible sequence relations of the jobs.

5.1.2 Nodes

In the graph, each node represents a job. We directly use the symbol J to represent the node. Nodes for jobs belonging to LO, MI or HI criticalities are shown in Figure 5.1.

Definition 8 (Subgraph)**.** *For each node $J \in V$, where V is the set of nodes in the graph G, 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*



(a) HI-criticality job. (b) MI-criticality job. (c) LO-criticality job. (d) Early job.

Figure 5.1: Graph nodes for jobs of various criticalities.

the job criticality modes: J^{LO}, J^{MI}, J^{HI};

- *if J is* MI*-criticality, i.e.* $L = \mathrm{MI}$*, 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.* $L = \mathrm{LO}$*, the node J does not contain a subgraph because this job does not enter higher modes.*

Figure 5.1 also shows these subgraph of the nodes for the jobs. Since the nodes with subgraphs represent job criticalities, there are associated WCETs in each of the criticalities. 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 9 (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 5.1d. These arcs are not considered part of the E set.

5.1.2.1 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 5.1, 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

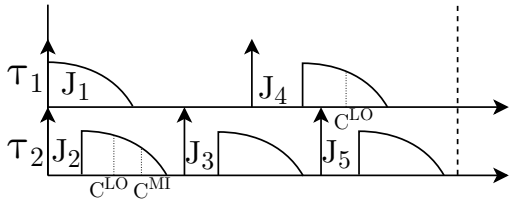


Figure 5.2: Probabilistic job executions of jobs of task set Γ_1 in a hyperperiod.

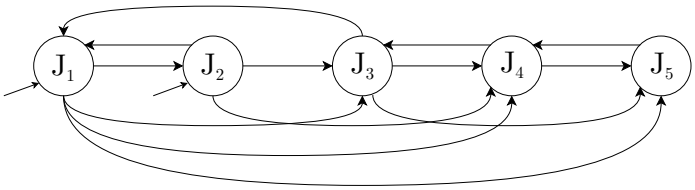


Figure 5.3: Graph for task set Γ_1.

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 10 (Exiting arcs from a subgraph)**.** *A node J of a subgraph is connected to the subgraph of a node J' according to the following specification:*

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

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.

Example 3. *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 5.2. 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 $\Lambda^{\mathrm{LO}} = \{J_1, J_3, J_5\}$. Thier WCETs for entering are labelled* MI *and* HI *criticality are $\mathcal{C}^{\mathrm{LO}}$ and $\mathcal{C}^{\mathrm{MI}}$ with $\mathcal{C}^{\mathrm{HI}}$ as the maximum WCET.*

The graph for the jobs is built as shown in Figure 5.3. Each arc represents a possible ordering between two jobs. For example, the arc $(J_1, J_2) \in E(G)$ represents the possibility to execute J_2 after J_1. Another example is the missing arc $(J_3, J_2) \notin E(G)$: we can not execute J_3 before executing J_2,

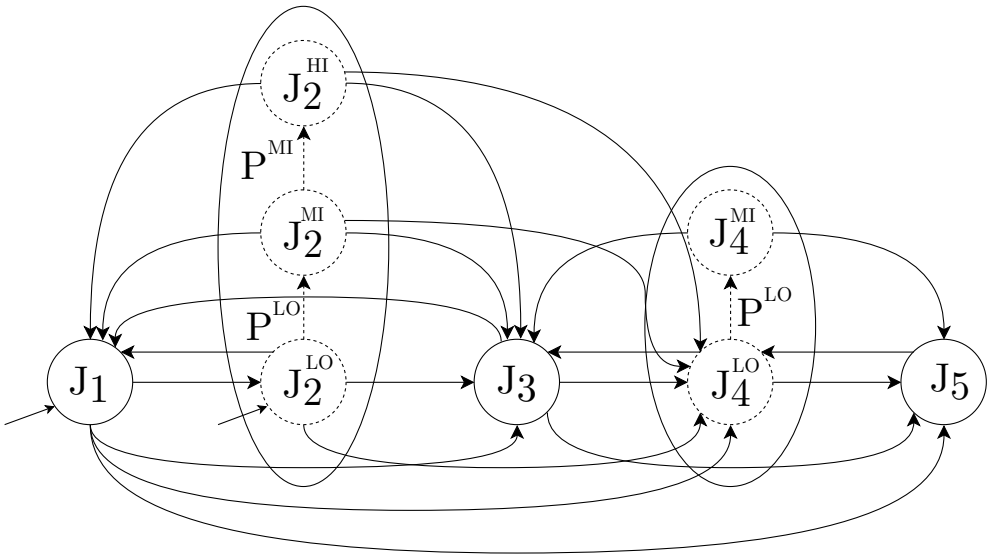


Figure 5.4: Graph with subgraphs for task set Γ_1.

otherwise J_2 would for sure miss the deadline. The early node set $S(G) = \{J_1, J_2\}$ 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 $J_4 \not\rightarrow J_1$.

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 as shown in Figure 5.4. Inside each subgraph, the mode changes are depicted with dotted arrows which are labelled with the probability. These mode changes represent an event happening at run-time, that can not be forecast during the offline scheduling analysis.

5.1.3 Graph properties

The arcs in the graph between nodes represent the possible ordering of execution of the jobs. Arcs in the subgraph represents uncertain transition to criticality modes of a job.

Any arc represents a possible sequence of events. Some properties of the graph are presented, in order to make the graph model consistent with the mixed-criticality real-time definitions.

Property 1 (Job order of the same task conserving)**.** *For any job J and J' of the same task, if J precedes J' in the job ordering, i.e. $a < a'$, then the arc (J', J) shall not exist because J' cannot start before its predecessor. This is implicit in the definition of arcs.*

Property 2 (No lower mode transitions)**.** *No job J can switch from a higher criticality L to a lower criticality L':*

$$\nexists (J^L, J^{L'}) \in E : L > L'$$

Property 3 (No LO to HI transitions)**.** *No job can directly switch from* LO*-criticality mode to* HI*-criticality mode. Because $C^{\mathrm{HI}} > C^{\mathrm{MI}} > C^{\mathrm{LO}}$, a job begins execution in* LO*-criticality mode, then*

Algorithm 1 Exploration forest building

```
1:  procedure ForestBuilding(G)
2:      for all J_s ∈ S(G) do                          ▷ For all jobs in early node set we build a tree
3:          J ← newNode()
4:          label(J_s) ← J_s
5:          V ← {J_s}
6:          E ← ∅ ExplTreeBuilding(J_s, Λ \ {J_s})
7:          T ← {V, E}
8:          F ← F ∪ {T}                                ▷ The new tree is added to the forest
9:  procedure ExplTreeBuilding(J_s, Λ̄)
10:     J_s ← label(J_s)
11:     A ← succ(J_s)
12:     A ← A ∩ Λ̄
13:     for all J' ∈ A do
14:         J' ← newNode()
15:         label(J') ← J'
16:         V ← V ∪ {J'}
17:         E ← E ∪ {J, J'}
18:         ExplTreeBuilding(J', Λ̄ \ {J'})
```

possibly switches to MI*-criticality mode and then possibly switch to* HI*-criticality mode. In the graph, it means that no arc of the form* $(J^{\mathrm{LO}}, J^{\mathrm{HI}})$ *exist. This is implicit in the definition of arcs in the subgraph: The arc* $(J^{L}, J^{L'})$ *exists if* $L < L''$ *and* $\nexists L' : L < L' < L''$.

5.2 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 acyclic graphs. The trees represent all possible sequences of job execution; we call them *exploration trees*. First, the complete construction process of the tree is presented; later, we will provide a method to reduce the construction complexity.

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 6.3. 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 11 (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 job 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 the unique

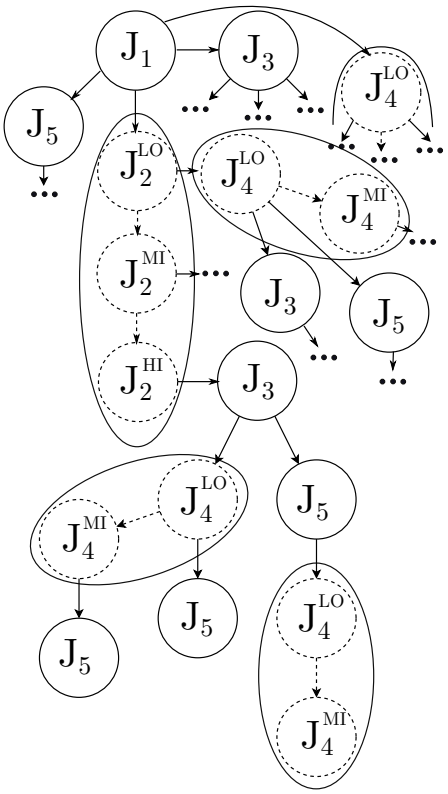


Figure 5.5: Exploration tree of task set Γ_1.

path with job J and leaf node J''' unless otherwise specified. Two paths are the same if their elements are the same. 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 the tree T if it already exists between the root $\bar{J}$ 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 4. *Using the example test case shown in Figure 5.2 and its graph in Figure 5.4, a portion of the exploration tree is shown in Figure 5.5. 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.*

5.2.1 Tree and Path properties

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 scheduled 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.

5.2.2 Comments

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.

Let there be n the number of jobs in one hyperperiod. In the worst case, all the jobs are connected by arcs in both directions and the cardinality of the arc set is the cardinality of a connected first-layer graph multiplied by 4 for two incoming and two outgoing transitions: $card(E) \leq 4 \cdot n(n-1)$ and thus $card(E) = O(n^2)$.

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 zero arrival time. This is because all time intervals overlap and in the graph all the nodes are interconnected. In this case, all the possible schedules have to be represented and the number of complete path is $O(n!)$. We introduce some metrics in the next section to decide on an optimal schedule. We also use a metric, the response time, to reduce the complexity of the process of building a tree such that the real complexity is much less than $O(n!)$.

5.3 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 offline the 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 they come from the pWCETs and pWCET 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 task parameters are requested.

5.3.1 Tree Analysis

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 12 (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 $WCRT''^{L}$ is defined as:*

$$WCRT''^{L} = \max(0, WCRT'^{L} + a'') + C''^{L}$$

where $L = \{\text{LO}, \text{MI}, \text{HI}\}$, 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 13 (Utilization)**.** *The utilization U 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 a certain

criticality mode. The pWCET of a job 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 a 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 [Nor97].

Definition 14 (System Criticality Probability)**.** *For a certain path $path(J, J''')$ for some J, the probability $\mathcal{P}_{sys}^{\mathrm{MI}}$ that the system enters* HI *criticality is given as*

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

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.

5.3.2 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 15 (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:*

$$WCRT(J'^{\mathrm{HI}}) \leq d' \forall J' \in path(J,J''') \text{ and } \Lambda^{\mathrm{HI}} \in path(J,J''') \tag{5.1}$$

The above definition filters all the possible paths to keep the ones in which no HI-criticality job misses a deadline. These paths are the only ones which are candidates for an optimal schedule. 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 16 (Dangerous Valid Path)**.** *A valid path* $\overline{path}(J,J''')$ *is said to be a dangerous path if:*

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

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 17 (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.*

From all the valid and non-dangerous paths, we keep one path which adheres to above conditions. These conditions allow maximum number of jobs to execute even when a HI-criticality job enters HI-criticality mode. This way, we optimize the schedule for resource usage by allowing maximum number of jobs to execute. 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 17. The tree $T \in F$ with certain root node $\overline{J} \in S$ is chosen in which there is 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.

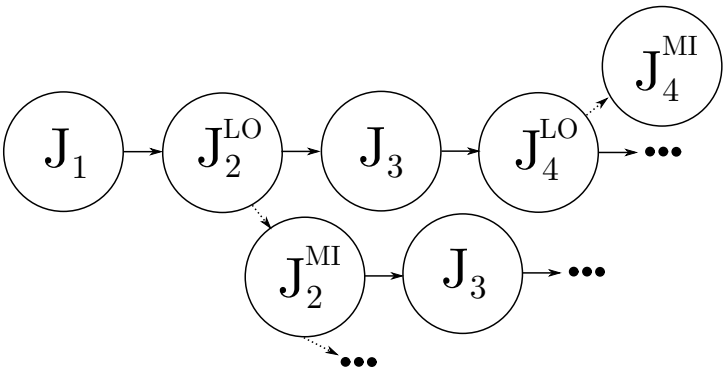


Figure 5.6: A portion of the optimized scheduling tree T_s^{schd} for task set Γ_1 with root node J_1.

5.3.3 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.

5.3.4 Strategy:

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 5.6. 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 an 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 5. *For the task set Γ_1, the scheduling tree T_s^{schd} is shown in Figure 5.6. 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 scheduled if J_4 remains in* LO*-criticality and does not enter* MI*-criticality.*

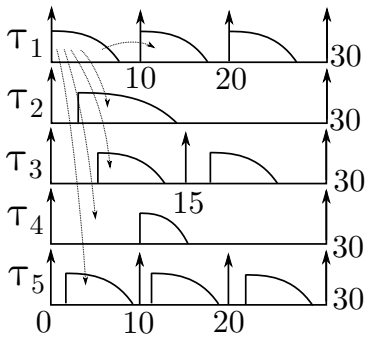


Figure 5.7: Jobs in Γ_2.

	$T_i = D_i$	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 5.1: Task set Γ_2.

5.4 Experiments

In this section we propose a non exhaustive but 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 5.1. 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^{LO}, C^{MI} and C^{HI} are also shown. The task set has 10 jobs in the hyperperiod equal to 30 time units. These jobs are shown in Figure 5.7. As an example, if first job of τ_1 executes, it can be followed by first jobs of the rest of the tasks 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 5.8. There are 134 nodes in the graph. It is developed into a forest of trees. A portion of the one of the trees in the forest the with J_{11} as root is shown in Figure 5.9. 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 5.2. 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

Job	Criticality	Schedule	Utilization
J_{11}	MI	$J_{51},J_{41},J_{31},J_{52},J_{12},J_{32},J_{13},J_{21}$	0.883
J_{11}	HI	$J_{51},J_{41},J_{31},J_{52},J_{12},J_{32},J_{13},J_{21}$	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 5.2: Schedules of the jobs of Γ_2 enter MI or HI criticality with schedule J_{11} ,J_{51} ,J_{41} ,J_{31} ,J_{52} ,J_{12} ,J_{32} ,J_{13} ,J_{21} ,J_{53}.

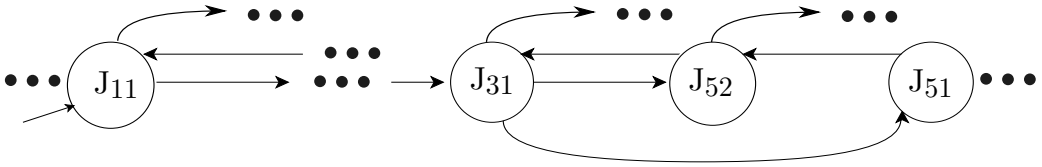


Figure 5.8: Graph for the jobs in Γ_2.

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.

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 5.3. 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_{ij}^{LO}, C_{ij}^{MI} and C_{ij}^{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.

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:

	$T_i = D_i$	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 5.3: Task set Γ_3.

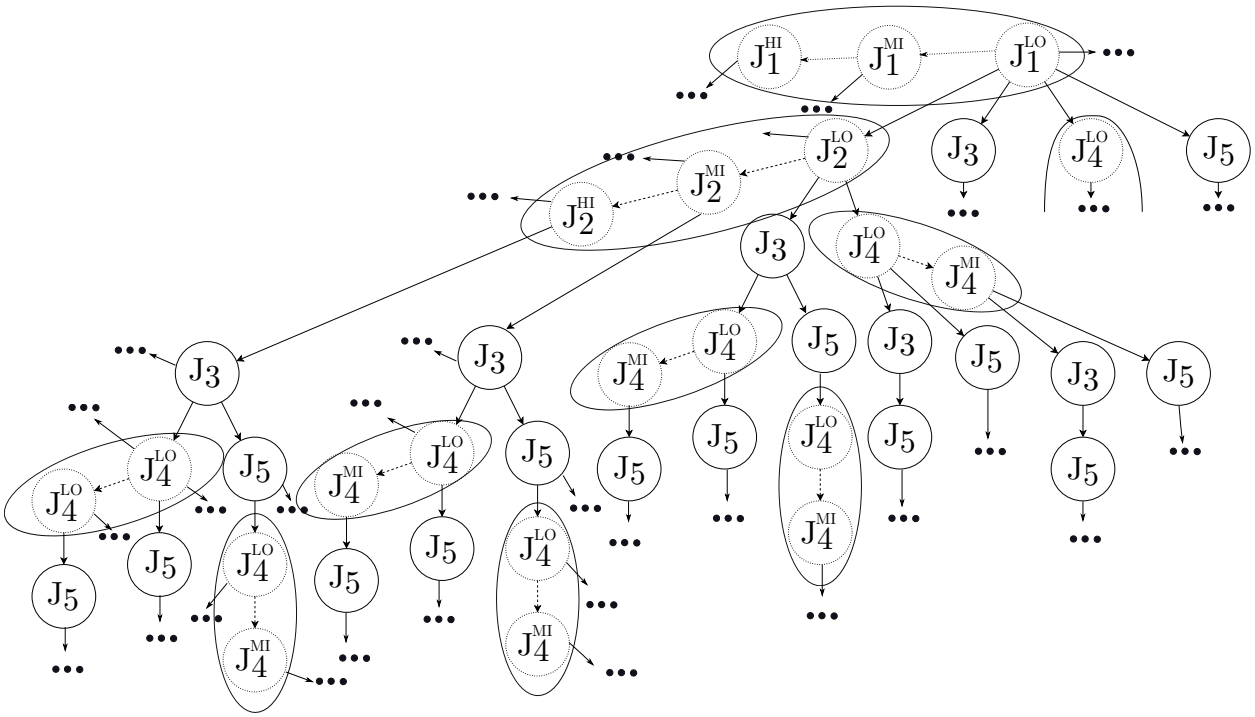


Figure 5.9: A portion of an exploration tree for task set Γ_2 with the root node J_1.

$J_{11}, J_{21}, J_{12}, J_{13}$. 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_{ij}^{LO}, C_{ij}^{MI} and C_{ij}^{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 5.4. 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.

5.5 Perspectives

In this chapter 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 criticalities. Each

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 5.4: Complexity as number of jobs vs number of nodes in exploration tree.

of those schedules are optimized in resource usage and ensured 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. In the next chapter, we intend to introduce mixed criticality defined using the response time which is application oriented, where probabilities come into play. 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.

Here we have partially tackled the observations made in the previous chapter. In the Chapter 3 and 4, we observed that on the one hand we are trying to exploit the ability of probabilistic models to quantify pessimism, and on the other, we are forced to do so through strict scheduling methodologies. This is mainly because the scheduling strategies like EDF or FP are not designed for probabilistic purposes in the first place.

In this chapter we obtained a schedule by keeping safety in mind. We mainly approached the problem through MC approach. Through the approach in this chapter, we removed the classical action in the MC domain that if one job enters high criticality, the system enters high criticality mode and all the lower criticality jobs are dropped from execution. We removed the notion of system criticality and brought the notion of criticality to the job level. By doing so, we are saved from forming a general rule for the whole system like the classical action of dropping the low criticality jobs. Moreover, we obtained a schedule by ensuring that, in no situation, the schedulability of the high criticality jobs is sabotaged. The schedule is an adaptive one where a job entering high criticality or not decides which jobs to schedule next.

The classical idea of dropping all the low criticality jobs does not take into account if there is room to execute some low criticality jobs, making it a pessimistic decision. In this thesis, we are trying to improve the pessimism. The classical idea has two more problems apart from being pessimistic. First, when the criticality levels are more than two, this action needs further clarification. That is, in multi-criticality systems, are all the lower criticality jobs dropped and only the highest is kept, or only the equal and higher criticality jobs are kept. In either case, the schedulability needs to be ensured at each level. In the current state of the art, the industry performs hardware separation and assigns tasks to different set of hardware altogether. However, as the number of criticalities increases, the set of hardware required will also increase. This makes the system simple yet bulky and expensive. This is particularly important when each gram to be launched in a satellite has a high cost associated.

Second, this action does not consider the instant when the job enters high criticality. If the jobs enters high criticality too late in the schedule, dropping all the low criticality jobs which have already executed may not be successful. Therefore we need a method which knows when exactly does a job

enter higher criticality and adapt accordingly.

We observed in Chapter 3 that the discrete representation of the pWCET can lead to complexity issues as each data point is preserved and used to compute system properties. We see such complexity in this chapter as the possible scenarios are enumerated and searched through. On the other hand, from the continuous representation, it is possible to reduce the complexity through mathematical representation But, there we lacked the knowledge of the job execution at each instant of time. We need to know the probabilistic job behaviour for a given time so that a schedule can be obtained and schedulability can be ensured. The schedulability that is ensured must be deterministic for the safety of the system. At this crossroad of complexity vs representativity, we chose to continue with discrete representation at the cost of complexity. We intend to deal with complexity in the future works.

In this chapter as well, the probabilities do not play a role in deciding the schedule. The probability is only quantified. This is because, as observed in the previous chapters, the probabilities come from the pWCET and pWCET does not change from the schedule. We will now proceed to the next and final chapter of the thesis where we make an observation on MC which naturally brings the probabilities into play. We will observe the MC to be application oriented and not task oriented where runtime behaviour of the jobs must decide the criticality and not the pWCET. We will also see it as an extension of this chapter where the graph model presented here becomes the model for providing upper limits to the probabilistic evaluation.

CHAPTER 6

Graph Based Mixed Criticality Schedulability Analysis: Response Time

---

> It is entirely possible that behind the perception of our senses, worlds are hidden of which we are unaware.
>
> *Albert Einstein*

In this chapter we further the work from the previous chapter by changing the fundamental definition of MC systems. We see the MC problem in a whole new perspective. We will see how this perspective is closer to the real case and how it benefits us. We state the problem as follows.

Given a Mixed Criticality probabilistic real-time system with task executions described with pWCET, what is the optimal application oriented schedule when system enters high criticality mode?

Our objective is to find a schedule which causes minimum probability of the system entering high criticality. We will re-utilize the graph model in the previous chapter. In doing so, the probabilities will come into play and help us make a scheduling decision. We will also upper bound this model for safety critical applications. The solution that we propose is based on an exploration of graph models for possible job execution combinations.

Using graphs allows us explore all the possible combinations of job scheduling. Only by exploring all the combinations, we can confidently conclude for a schedule with the least probability of the system switching mode to high criticality among all the possibilities. Moreover, using graphs enables us to precisely predict occurrences of certain events, like a job entering high criticality, and foresee the appropriate action thereafter which is best for the system. We are able to extract crucial schedulability information like pWCRT and the probabilities associated. We begin by recalling the graph model.

Dans ce chapitre, nous approfondissons les travaux du chapitre précédent en modifiant la définition fondamentale des systèmes MC. Nous voyons le problème MC dans une toute nouvelle perspective. Nous verrons comment cette perspective est plus proche du cas réel et comment elle nous profite. Nous énonçons le problème comme suit.

Étant donné un système temps réel probabiliste à criticité mixté avec des exécutions de tâches décrites avec pWCET, quelle est la planification optimale orientée application lorsque le système passe

en mode de criticité élevée?

Notre objectif est de trouver un ordonnance qui entraîne une probabilité minimale que le système entre dans une criticité élevée. Nous réutiliserons le modèle graphique dans le chapitre précédent. Ce faisant, les probabilités entreront en jeu et nous aideront à prendre une décision de planification. Nous allons également limiter ce modèle pour les applications critiques pour la sécurité. La solution que nous proposons est basée sur une exploration de modèles de graphes pour d'éventuelles combinaisons d'exécution de travaux.

L'utilisation de graphiques nous permet d'explorer toutes les combinaisons possibles de planification des tâches. Ce n'est qu'en explorant toutes les combinaisons que nous pouvons conclure en toute confiance pour un ordonnance avec la moindre probabilité du mode de commutation du système à une criticité élevée parmi toutes les possibilités. De plus, l'utilisation de graphiques nous permet de prédire avec précision les occurrences de certains événements, comme un travail entrant dans une forte criticité, et de prévoir ensuite l'action appropriée qui est la meilleure pour le système. Nous sommes en mesure d'extraire des informations cruciales d'ordonnancabilité comme pWCRT et les probabilités associées. Nous commençons par rappeler le modèle graphique.

6.1 Graph and Tree Model

We quickly recall the graph model from the previous chapter. We consider two level criticality model of the jobs, HI and LO.

Graph: The graph represents the possible job combinations as 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.

Nodes: Each node $J \in V$ represents the execution of a job $J \in \Lambda$.

The *early nodes set* S is a subset of the node set V, $S \subseteq V$ such that $\forall J \in S : a = 0$. Graphically, we identify the early nodes set S with extra arcs entering in $J \in S$ without the source node.

Arcs: 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 $a < d'$. A *successor node* as $succ(J) = J'$ and a *predecessor node* $pred(J') = J$ for (J, J').

Scheduling 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. The *exploration tree* is constructed from the graph beginning with a root node of the graph as the first job to execute in the schedule. Each node of the tree is labelled with job J it represents. A node is only added to the tree if it does not miss its deadline and a corresponding arc exists in the graph given the corresponding node exists the graph. The deadline miss is checked for at each time it is added to the tree. If there is a deadline miss, the node/job is not added because the system should never be scheduled beyond this job.

The set of all exploration trees for different roots is called the *exploration forest* F. The trees

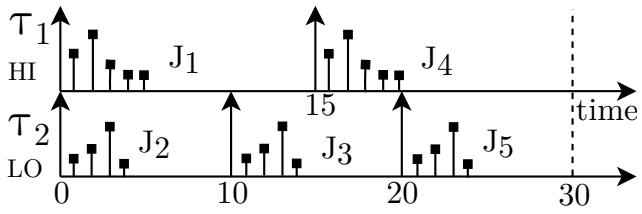


Figure 6.1: Jobs of task set Γ_1.

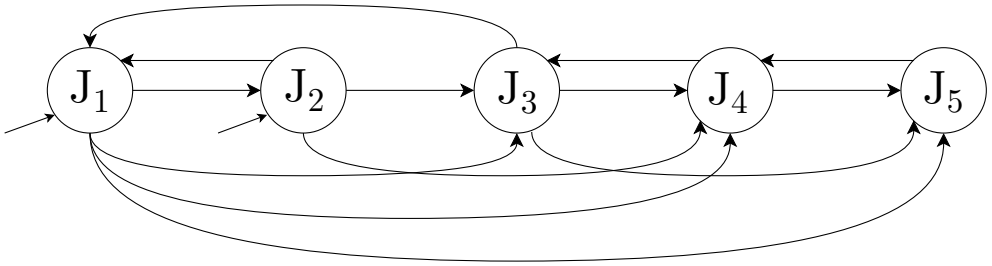


Figure 6.2: The graph of the jobs in Γ_1.

represent possible orderings or sequences of jobs beginning with an early job at the root. A *leaf node J* is a node without successors, i.e. $\text{succ}(J) = \{\emptyset\}$. In order to navigate the tree to obtain a schedule, we define a path.

We also recall the definition of a 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.*

Also, A node J' does not exist in tree if it already exists between the root and the desired point of addition. $J' \neq succ(J)$ if $\exists (J,J') \in path(\bar{J},J'''), J'''$ is a leaf node, $\bar{J} \in S$

Example 6. *We use a task set Γ_1 shown in Figure 6.1 in parallel through this paper to explain our method. It consists of five jobs of two tasks in the hyperperiod equal to 30 time units. The jobs of the tasks are shown with their pWCET PMF in the Figure, the exact values of PMFs are not yet important. The jobs of τ_1 are* HI *criticality and those of τ_2 are* LO *criticality.*

The graph, which is a set of nodes connected by arcs, is a direct representation of all the possible ordering of the jobs in the system. It is shown for the task set Γ_1 in the Figure 6.2. There are nodes J_1, J_2, etc. for each job which are interconnected by uni-directed arcs which refer to the order of executions. E.g. if J_3 is executing, then it can be followed by the jobs J_1, J_4 or J_5. The system can begin execution with J_1 or J_2. This set of early nodes of the jobs J_1 and J_2 are represented by the extra arcs entering those nodes. In order to explore this graph to look for a schedule, it is unfolded into a forest of trees.

A portion of the tree is represented in the Figure 6.3 for the task set Γ_1. The tree begins with a root J_1 followed by possible jobs which can execute, namely, J_2, J_3, J_4, J_5. This goes on until the leaf nodes where any more addition of nodes will mean a repetition of the same job in the schedule on the path. For the tree task set Γ_1 the Figure 6.3, one possible path is shown by a dotted line. The dotted line represents a path J_1,J_2,J_4,J_3,J_5. It also shows another possible tree with just its root J_2.

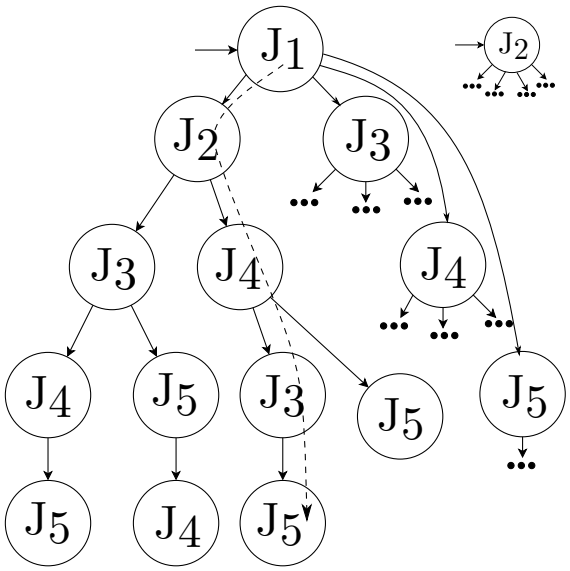


Figure 6.3: Graph unfolded into a tree for jobs of Γ_1.

At this point, the construction of the exploration model is complete. We now proceed extracting certain metrics from these exploration trees in order to decide for a MC schedule.

6.1.1 Properties

We first the define the metrics necessary for system criticality and then apply them to our model. We begin with obtaining a non-preemptive pWCRT for a job. Classically, the pWCETs of the jobs are simply convoluted in order to obtain the pWCRT. The convolution operation does not take different arrival times into account. Thus, performing a convolution contains a hidden assumption that all the jobs arrive at the same time, i.e. at a critical instant. This results in a pessimistic pWCRT. Our approach to obtain the pWCRT involves handling the discrete distributions in a piece-wise manner. To do so, we first define a tail PMF as follows.

Definition 18 (Tail Probability Mass Function)**.** *A tail distribution of the probabilistic Worst Case Response Time $f_R(x)$ of a job J with Worst Case Execution Time WCET for some time $a'(0 < a' < WCET)$, is a PMF $f_{tail}^{[a',WCET]}(x)$ given as:*

$$f_{tail}^{[a',WCET]}(x) = \begin{cases} f_{\mathcal{R}}(x+a') & \text{if } 0 < x \leq WCET \\ \Sigma_{y=0}^{a'} f_{\mathcal{R}}(y) & \text{if } x = 0 \end{cases}$$

The tail distribution represents a complete PMF which probabilistically delays the execution of the next job in the schedule where the next job arrives at time a'. The upper limit of the tail function is the WCET because it is the maximum possible execution time by the definition of pWCET. The probability accumulated at the instant a' ($x = 0$ in the function) in the tail PMF represents the probability that the job J has finished execution by then. As in Figure 6.4, the tail distribution of job J_1 (from Γ_1)

accumulates $0.1+0.15+0.15+0.2=0.6$ time zero. By doing so we prevent any loss of information in the distribution. We use this to obtain the pWCRT of the delayed job.

Theorem 1 (Non-Preemptive Probabilistic Worst Case Response Time)**.** *Non preemptive Probabilistic Worst Case Response Time f_R' of a job J' is represented by convolution between its probabilistic Worst Case Execution Time f, and the Tail Probability Mass Function $f_{tail}^{[a',WCET]}$ of the job J executing immediately before.*

Proof. Consider two jobs, J arriving at time a and J' arriving at time a', $a \leq a'$. These are shown in Figure 6.4 as J_1 and J_3 for the task set Γ_1. Probabilistically, J can continue to execute after the arrival time of J'. That means, there exists a probabilistic delay to the execution of the job J' due to the execution of the job J.

$$Event(J' \text{ finishes at some time } x) = Event(J' \text{ finishes at some time } x) \text{ and } Event(J' \text{ finishes after } a'd)$$
$$\text{or } Event(J' \text{ finishes at some time } x) \text{ and } Event(J' \text{ finishes before } a'd)$$

Let, the random variable of execution of J is $\mathcal{C}_J$, the response time random variable of J is $\mathcal{R}_J$ and the response time random variable of J' is $\mathcal{R}_{J'}$.

$$Pr(\mathcal{R}_{J'} = x) = Pr(\mathcal{R}_{J'} = x \text{ and } \mathcal{R}_J \leq a'-a) + Pr(\mathcal{R}_{J'} = x \text{ and } \mathcal{R}_J > a'-a)$$

Now consider a random variable $\mathcal{R}_J'$ such that,

$$\mathcal{R}_J' = \begin{cases} 0 & \text{if } \mathcal{R}_J \leq a'-a \\ \mathcal{R}_J - (a'-a) & \text{if } \mathcal{R}_J > a'-a \end{cases}$$

The PMF $f_{\mathcal{R}_J'}$ of $\mathcal{R}_J'$ is,

$$f_{\mathcal{R}_J'} = \begin{cases} f_{\mathcal{R}_J'}(0) = Pr(\mathcal{R}_J' \leq a'-a) = \Sigma_{x=0}^{a'-a} f_{\mathcal{R}_J}(x), & \text{for } x = 0 \\ f_{\mathcal{R}_J'}(x) = f_{\mathcal{R}_J}(x+a'-a), & \text{for } x > 0 \end{cases}$$

Therefore we have,

$$Pr(\mathcal{R}_{J'} = x) = Pr(\mathcal{C}_{J'} + \mathcal{R}_J' = x \text{ and } \mathcal{R}_J' = 0) + Pr(\mathcal{C}_{J'} + \mathcal{R}_J' = x \text{ and } \mathcal{R}_J' > 0)$$
$$= Pr(\mathcal{C}_{J'} + \mathcal{R}_J' = x)$$

This leads to the convolution of the PMFs,

$$f_{\mathcal{R}_{J'}} = f_{\mathcal{C}_{J'}} \otimes f_{\mathcal{R}_J'}$$

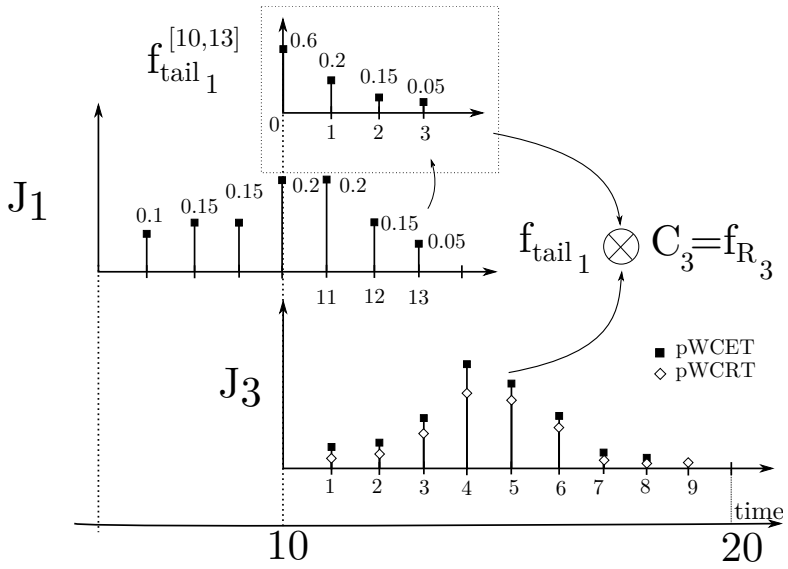


Figure 6.4: pWCRT from convolution of Tail distribution and the pWCET.

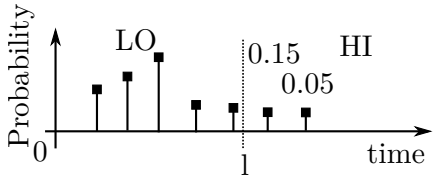


Figure 6.5: High criticality from a threshold l_1 on pWCRT of a job J_1 of Γ_1.

From the definition of $f_{\mathcal{R}'_J}$, it is the tail distribution of the response time of J. Simplifying the notations, the response time of J' is,

$$f'_{\mathcal{R}} = f' \otimes f_{tail}^{[a',WCET]}$$

□

Our approach to obtain the pWCRT separates the part of the pWCRT PMF which affects the execution of the next job. We do not lose any information in the distribution as the probabilities before the time a' are accumulated [1]. Moreover, we retain the information within the time intervals in the distribution. This approach is similar to the ones used by [Kim+05]; [Max+17] but applied to every job with respect to their arrival times.

Using the pWCRT we apply the job criticality Definition 2 on the paths through the exploration tree.

Definition 19 (Job Criticality)**.** *A job J' in a path $path(J,J''')$ is said to have entered high criticality if its response time crosses the threshold l', $f_{\mathcal{R}'}(x) > l'$, the probability $\mathcal{P}^{\mathrm{HI}}(path(J,J'''))$ of which is*

[1] The distribution function accumulates the probabilities in the intervals of discretization at the worst case, e.g. probabilities at execution times $0.2, 0.5, 0.7$,etc. are accumulated the time 1.

given as

$$\mathcal{P}^{\mathrm{HI}}(path(J,J''')) = 1 - \Sigma_{x=0}^{l'} f_{\mathcal{R}'}(x)$$

where $l' < d'$ and J''' is a leaf node.

We pause to exhibit these definitions in the following example.

Example 7. *Figure 6.4 shows a scenario where job J_3 executes after J_1 from the task set Γ_1 For explanation, a pWCET PMF is assumed for J_1. The Tail PMF for the same is shown in the box. To obtain the pWCRT of J_3, this PMF is then convoluted with the pWCET of the job J_3.*

A pWCRT of the job J_1 with its threshold is shown in Figure 6.5. The threshold is shown by the dotted line labelled $l = 12$. From the Figure, the probability that this job enters HI*-criticality mode is* $0.15 + 0.05 = 0.20$.

Continuing our discussion, from the job criticality defined using its pWCRT, we define the probability of the system entering high criticality as follows.

Definition 20 (System Criticality)**.** *The system enters high criticality mode if at least one high criticality job enters high criticality mode.*

We apply these definitions to the graph and tree model. In our model, the schedule is represented as paths. Because the paths represent the schedule, the probability that the system enters high criticality is also a function of paths. However, not all the paths in the exploration tree are available for scheduling because not all paths contain all the high criticality jobs. The set of available paths is defined as follows.

Definition 21 (Available paths)**.** *Available paths is a set of all possible paths in a tree from the root node $J \in S$ to a leaf node J''', $P_{avail} = \{path(J,J''')\}$ such that*

$$\forall J' \in \Gamma, J' \in path(J,J'''); \quad \textit{and}$$
$$\forall J' \in path(J,J''') \implies 1 - \Sigma_{x=0}^{d'} f_{R'}(x) < P_{dm}^{max}$$

where P_{dm}^{max} is the maximum allowed probability of deadline miss of any job, $f_{R'}(x)$ is the pWCRT of job J' and d' being its deadline.

P_{avail} is a set of possible schedules of the jobs in MC through the trees as the available paths. The available paths are the paths which contain all the jobs (low as well as high criticality) and all those jobs meet their respective deadlines. In our context, the set of available paths represent the possible candidates to find a schedule. It should be noted that the criteria of all jobs meeting their deadlines is already met while constructing exploration tree. A node is not added to the tree if it misses its deadline. To quantify the probability of these paths we apply the definition of system criticality on the available paths as follows. We elaborate the definition of system criticality. The system enters high criticality if the first high criticality job enters high criticality OR the second high criticality job enters high criticality OR the third..., and so on. This represents a summation of probabilities. 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 [Nor97].

Definition 22 (Probability of system entering high criticality)**.** *For an available path* $path(J,J''') \in P_{avail}$*, the probability* $\mathcal{P}_{sys}(path(J,J'''))$ *that the system enters high criticality by taking this path is given as:*

$$\mathcal{P}_{sys}(path(J,J''')) = 1 - \prod(1 - \mathcal{P}'^{\mathrm{HI}}), \forall J' \in path(J,J''') \ \ \& \ \ \Gamma^{\mathrm{HI}}$$

Using the available paths and the system criticality metric defined above, we finally obtain a *mixed criticality (MC) schedule* as follows.

Definition 23 (Mixed Criticality Schedule)**.** *A path* $P_{MC} = path(J,J''')$ *is the mixed criticality schedule if* $\mathcal{P}_{sys}(path(J,J'''))$ *is the minimum among all possible* $path(J,J''') \in P_{avail}$ $\exists J \in path(J,J''')$*, and* $J \in \Lambda$ *where* $J \in S$*.*

The MC schedule is a path P_{MC} through the exploration tree which contains all the low as well as high criticality jobs, no job misses its deadline and the probability of the system entering high criticality is minimum. It represents the schedule that should be taken by the system as the ordered sequence of execution of the jobs. To recall, the probability is minimum given the criticality is defined using the response time. Thus, we have found a solution to our problem which is a schedule represented by the path P_{MC}. The path P_{MC} by definition is from root node to a leaf node, thus we do not need to notate them (unlike jobs J and J''' in $path(J,J''')$). Since this method is based on complete exploration, the schedule is guaranteed to be the optimal by minimizing the probability of system entering high criticality. Any impossible schedule is the one in which a job does not meet its timing constraints. These impossible schedules are already excluded while building the tree as we calculate response time in parallel.

Therefore, as the input we assume a mixed criticality task set Γ on a uniprocessor machine. As the output, a non-preemptive schedule P_{MC} is provided for all the jobs of the tasks with minimum probability of system entering high criticality mode.

Complexity: The complexity of our approach depends on the number of jobs n. The maximum complexity of building the tree is $O(n!)$. However, this might not always be the case because the tree is not built in the direction of a node which missed its deadline. Thus, the complexity also depends on the maximum allowed probability of deadline miss. The complexity of analyzing the tree and finding the paths is linear to the number of leaf nodes in the tree. In addition to this, there exists computational complexity of the convolution operation, which in turn depends on the possible values a random variable can take. Assuming that all jobs release at one critical instant, the convolution complexity is $O(n^n)$.

We apply this model to a non-exhaustive but realistic test case as follows.

Example 8. *We analyze the task set* Γ_2 *shown in Table 6.1 which consists of 4 tasks and 15 jobs with pWCET and implicit deadline as shown. There are 6 high criticality jobs from task* τ_1 *and 11 low criticality jobs from the rest of the tasks. To recall, the task set is to be scheduled non-preemptively on a uniprocessor system and the jobs are suspended at their deadlines. The threshold for the high criticality jobs of task* τ_1 *is set at* 4 *time units. The maximum allowed probability of deadline miss* P_{dm}^{max} *for any job is set at* $1E-03$*.*

Task	Deadline	pWCET	Criticality
τ_1	10	$\begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.094 & 0.005 & 0.001 \\ 1 & 2 & 3 & 4 & 5 & 8 \end{bmatrix}$	HI
τ_2	20	$\begin{bmatrix} 0.1 & 0.4 & 0.4 & 0.1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$	LO
τ_3	15	$\begin{bmatrix} 0.1 & 0.4 & 0.3 & 0.2 \\ 1 & 2 & 3 & 4 \end{bmatrix}$	LO
τ_4	30	$\begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 1 & 2 & 3 \end{bmatrix}$	LO

Table 6.1: Task set Γ_2.

The proposed schedule P_{MC} is:
$J_{11}, J_{41}, J_{31}, J_{32}, J_{21}, J_{12}, J_{13}, J_{42}, J_{33}, J_{22}, J_{14}, J_{15}, J_{34}, J_{23}, J_{16}$. *The probability that the system enters high criticality is* 0.0.00509. *The pWCRT of some of the jobs is shown in Figure 6.6. The pWCRT jobs of the high criticality task τ_1 remains unchanged as their pWCET as shown in Figure 6.6a. The pWCRTs of jobs which have been affected by probabilistic delay in execution are J_{21}, J_{22}, J_{23}, J_{31}, J_{33}, J_{34} and J_{41} whose pWCRTs are shown in Figures 6.6b,6.6c,6.6d,6.6e,6.6f and 6.6g, respectively.*

We obtain this result along with other possible schedules in which there is no deadline miss but the probability of the system entering high criticality is higher. For example, another possible schedule is: $J_{11}, J_{21}, J_{12}, J_{31}, J_{41}, J_{32}, J_{13}, J_{14}, J_{22}, J_{42}, J_{15}, J_{33}, J_{34}, J_{23}, J_{16}$. In this case the probability of system entering high criticality is 0.0.00605.

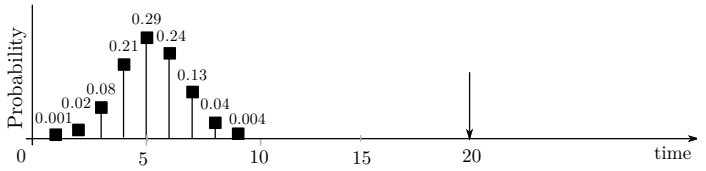
On computation, the tree for the task set Γ_2 contains 716,132 nodes.

Overall, we observe that in this section we can only quantify the probability of something occurring in the system, like deadline miss or entering high criticality. This gives us a global picture of the MC system with pWCETs in terms of risk involved of system entering high criticality when applying such a system. In addition to obtaining this picture in terms of probabilities, we can also control it through observing the pWCRTs and making the scheduling decisions accordingly. If there exists a probability of occurrence of an event, it exists each time it occurs. That is, each time a MC system is applied, the probability of it entering high criticality exists at each time instant. This does not tell us, exactly when does the system enter high criticality. However, for safety critical systems in high criticalities, we need a much more granular observation of the job execution instead of a global probabilistic view. For a job executing in high criticality, we need to know whether the system is still schedulable in the worst case even though probabilistically it might seem unlikely. Because of this requirement, we go one step further into the graph representation in the next section.

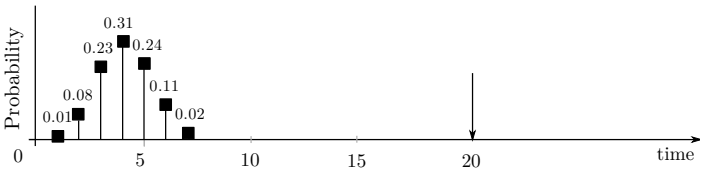
6.2 Worst Case Scheduling Model

We recall the second problem from the beginning of this chapter. If a mode change does occur for a given schedule, what is the optimal (adjusted) schedule from the mode change instant onward, such that resource usage is maximized. Without calculating the utilization, we simply interpret it as allowing

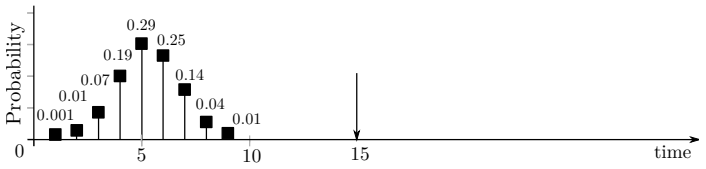
(a) Jobs of task τ_1



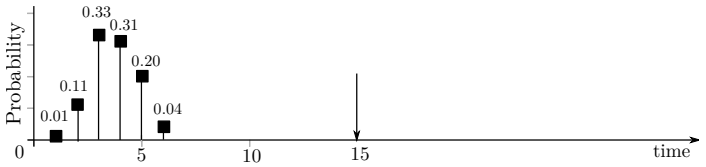
(b) Job J_{22}



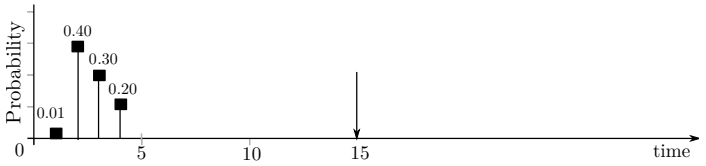
(c) Job J_{23}, J_21



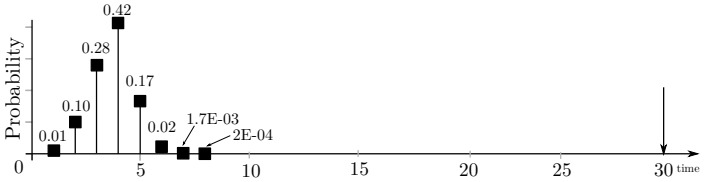
(d) Job J_{31}



(e) Job J_{33}



(f) Job J_{34}



(g) Job J_{41}

Figure 6.6: pWCRT distributions of some of the jobs of the task set Γ_2.

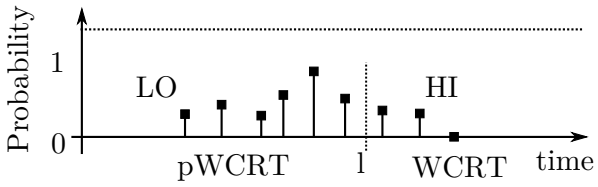


Figure 6.7: pWCRT of a job with threshold for criticality mode.

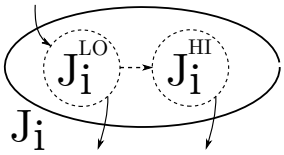


Figure 6.8: HI and LO subnodes within the node for high criticality jobs.

maximum number of low criticality jobs to execute. This also extends to ensuring that all the scheduled jobs meet their deadlines in all the criticalities. Because this is a safety critical scenario we no longer talk in terms of probabilities. We use the WCET of the job as the worst case of the given pWCET to analyze the system in high criticality mode. We use the schedule obtained in the previous section in the worst case in order to maximize the number of low criticality jobs to execute, without compromising the scheduling of high criticality jobs.

6.2.1 Model

We recall the Worst Case Execution Time $WCET$ for a job J with C, is the maximum execution time with a non-zero probability, $WCET = max(C)$. The threshold l on the pWCRT shown in Figure 6.7 is assumed as in the previous section.

We recall the nodes defined for the graph and the exploration trees which represent the execution of a job. Now, inside the node of each high criticality job, we define two subnodes. The subnodes represent the job executing first in low and then it can enter high criticality by taking more time to execute. This is shown in Figure 6.8 as a high criticality job J and the subnodes J^{LO} and J^{HI} inside.

The J job exits J^{LO} LO subnode to enter HI subnode J^{HI} if its response time crosses the threshold l, $WCRT_i(path(J',J''')) > l$, for any path $path(J',J''')$. It can exit LO subnode also if it finished execution before crossing the threshold. The J job exits HI subnode J^{HI} if its response time crosses the deadline d, $WCRT_i(path(J',J''')) > d$, for any path $path(J',J''')$.

Here we recall the result of the previous section which is a MC schedule with the lowest probability of system entering high criticality, represented with a path P_{MC}. This path forms the intialization of our scheduling tree as a tree trunk.

Scheduling Tree Trunk: The scheduling tree trunk is a directed graph $T_{tr} = \{V(T_{tr}), E(T_{tr})\}$ where $V(T_{tr})$ is the set of nodes representing the jobs and $E(T_{tr})$ is the set of arcs connecting them, such that all the jobs in Γ are exist in $V(T_{tr})$.This represents the schedule obtained in the previous section as the path P_{MC}. The system executes this schedule in this sequence until a high criticality job enters high

criticality mode. This happens when the job execution time is more than the criticality threshold. In that case, another path is taken which is characterized as a branch from this trunk.

Scheduling Tree Branches: A scheduling tree branch with an initial job node $J \in P_{MC}$ is a directed graph $T_{br} = \{V(T_{br}), E(T_{br})\}$ such that $\exists E(T_{br}) \in P_{MC}$ and $J \in \Gamma^{\text{HI}}$. This means that the job J is a high criticality job and the arcs in this branch exist in the schedule P_{MC}.

$$\forall J'' : J'' \neq J' \wedge J'' \neq pred(J') \ \wedge$$
$$J'' \neq pred(pred(J')) \wedge \cdots \implies J'' \in V(T_{br})$$

This means that the branch must contain all the HI criticality jobs which remain to be executed. This implies that there are branches which can omit some jobs. Thus, there are many possible branches from a high criticality job in the schedule, ranging from all the jobs to only one job scheduled thereafter. By combing the trunk and the branches, we define the scheduling tree.

Definition 24 (Scheduling Tree)**.** *A scheduling tree is a directed graph $T_{sched} = T_{tr} \cup T_{br}$ where T_{tr} is the trunk and T_{br} are the branches from the high criticality nodes of the high criticality jobs of the trunk.*

The scheduling tree represents the system schedule, first in system low criticality, and then possible schedules when certain high criticality job enters high criticality mode. The system begins in the low criticality mode and starts to execute the jobs in the trunk starting from the root. If any job in the trunk enters high criticality, there are many possible branches which can be taken. The system schedule must take one of these possible branches when the job enters high criticality. We prune this tree in the next subsection to obtain an optimal tree as an optimal schedule.

Example 9. *A portion of the scheduling tree for the task set Γ_1 is shown in Figure 6.9 where the trunk is represented with the thick nodes as schedule J_1, J_2, J_4, J_3, J_5. Because jobs J_1 and J_3 are high criticality jobs, there are high criticality subnodes for those jobs. To these high criticality subnodes, all the branches added.*

6.2.2 Properties

We reuse the definition of a Path which begins at a certain node and ends at a leaf node by navigating through the tree. There are such paths through the scheduling tree T_{sched}. In order to obtain an optimized schedule, we extract the following information from the scheduling tree T_{sched}.

Definition 25 (Worst Case Response Time)**.** *The Worst Case Response Time (WCRT) for a job J_k in a path $path(J, J''') \in T_{sched}$ is defined as the summation of its Worst Case Execution Time (WCET) and the execution delay from the previous job J',*

$$WCRT''(path(J, J''')) = WCET'' + max(WCRT_j(path(J, J''')) + a'', 0)$$
$$: (J', J'') \in path(J, J''')$$

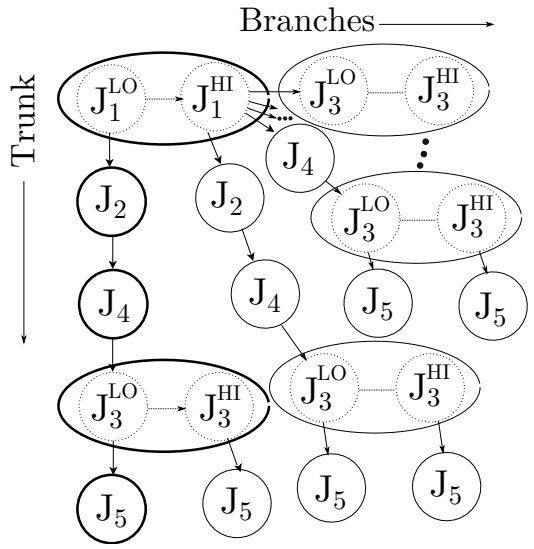


Figure 6.9: The scheduling tree T_{sched} for the schedule in P_{MC}.

We use this definition to say that a job J'' in a path $path(J,J''')$ is said to have missed its deadline if $WCRT''(path(J,J''')) > d''$. It should be noted that we do not define the WCRT as the worst case of pWCRT.

In a high criticality mode of the system, it is essential to ensure that there is no scheduled job which misses deadline. Moreover, in a MC system, all the high criticality jobs should essentially be scheduled. Therefore, using the definition of deadline miss, we define a valid schedule as follows.

Definition 26 (Valid Schedule for a Job)**.** *For a high criticality job $J' \in P_{MC}$, a valid schedule is a path $P_{valid}(J',J''')$ through the scheduling tree T_{sched} such that it necessarily contains all the high criticality jobs which remain to be executed; and no job misses its deadline in all of its criticality modes. For all $J' \in \Lambda^{\mathrm{HI}}$,*

$$\begin{aligned} \forall J'' : L'' = \mathrm{HI} \wedge J_k \neq J' \wedge J'' \neq pred(J') \wedge \\ J'' \neq pred(pred(J')) \wedge \cdots \implies J'' \in P_{valid}(J',J''') \\ and \\ WCRT'' \leq d'' \ \ \forall J'' \in P_{valid}(J',J''') \end{aligned}$$

This definition filters out any branch which does not contain all the high criticality jobs and the ones which have a deadline miss. A valid path denotes a possible schedule from a high criticality job without any deadline miss. Now after the filtering, there can be many possible valid paths for any job in higher criticality. These paths represent the schedule if a job enters high criticality from that point onward. Any such schedule must contain all the high criticality jobs in the least. Thus, there can be more than one valid path through the scheduling tree by changing the number of low criticality jobs included in the schedule. We choose one valid path which is optimal defined as follows.

Definition 27 (Optimal Schedule for a Job)**.** *For a high criticality job J', an optimal schedule $P_{opt}(J',J''')$ is a valid schedule $P_{opt}(J',J''') \in T_{sched}$ which contains the maximum number of low*

criticality jobs which remain to be executed.

This definition filters out the valid paths from a high criticality job to keep one. The system schedule through this one valid path which ensures that all the deadline requirements are met even when a job enters high criticality mode. Moreover, it maximizes the number of low criticality jobs kept for execution. This removes the pessimism that exists in the classical case where all the low criticality jobs are dropped when a job enters high criticality, even if there could be room to execute the low criticality jobs. Therefore, we finally define an optimal scheduling tree.

Definition 28 (Optimal Scheduling Tree)**.** *An optimal scheduling tree T^{opt}_{sched} is a pruned scheduling tree T_{sched} with one optimal schedule for each high criticality job.*

$$E^{opt}(T_{tr}) = E(T_{tr}) - P_{valid}(J', J''') + P_{opt}(J', J''')$$
$$\forall J' \in P_{MC} \wedge J' \in \Lambda^{\mathrm{HI}}, T^{opt}_{sched} = T^{opt}_{br}(G) \cup T_{tr}$$

where $T^{opt}_{br}(G) = \{V(T_{br}), E(T_{br})\}$ and $P_{valid}(J', J''')$ are all the possible paths beginning with the job J'.

This way, the system simply has to follow the scheduling of the jobs defined by T^{opt}_{sched}. As soon as a high criticality job enters high criticality mode, the schedule from that job onward is defined in T^{opt}_{sched}. We have already minimized the probability that the system enters high criticality in the previous section. Then, in the deterministic case for safety critical system in this section, we have maximized the number of low criticality jobs which can execute.

Example 10. *Referring to the same task set Γ_2 as in the previous section, we further it to obtain an optimal scheduling tree. To recall the best schedule P_{MC} is:*
$J_{11}, J_{41}, J_{31}, J_{32}, J_{21}, J_{12}, J_{13}, J_{42}, J_{33}, J_{22}, J_{14}, J_{15}, J_{34}, J_{23}, J_{16}$.

The high criticality jobs are those belonging to the task τ_1. For each of those jobs, there exists an optimal schedule from that point onward. For jobs, J_{11}, J_{13} and J_{15}, some valid paths are shown in Figure 6.10. If J_{11} enters high criticality, we can choose to schedule either J_{21} or J_{41} or J_{32} until J_{12} is scheduled. After J_{12}, job J_{13} is scheduled, so there is nothing to allow or suspend there. If J_{13} enters high criticality, J_{22} can be allowed to execute until J_{14}. Job J_{14} is followed b J_{15}. If J_{15} enters high criticality, either J_{34} or J_{32} can be allowed to execute before job J_{16} executes. We choose to allow J_{21}, J_{22} and J_{32} among the low criticality jobs. Thus, we are able to safely execute 3 low criticality jobs apart from all the high criticality jobs in the system high criticality mode.

6.3 Dependence

We take a step further and study the effect of the task dependence on a schedule. In this section we present a metric to introduce dependence in a mixed criticality probabilistic real-time system. Dependence can exist in various ways between jobs, e.g. a job demanding higher execution time shares a resource with another job, the result of one job is used by another and the result takes longer time to obtain, etc. We assume for this work that there is no more than one job with dependence

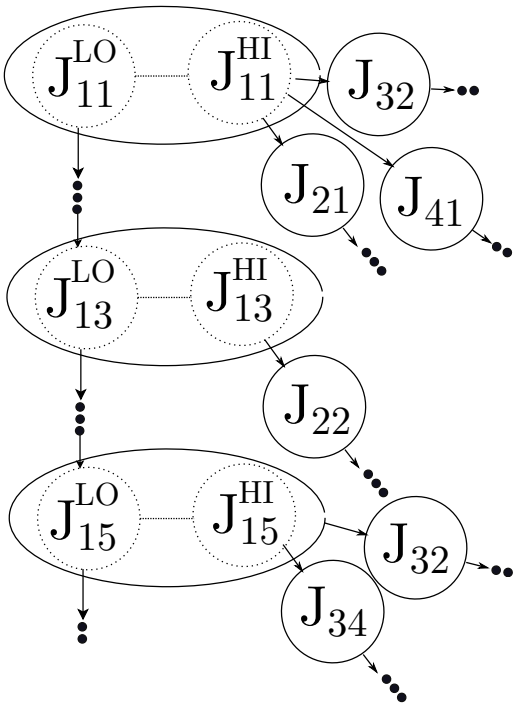


Figure 6.10: Optimal scheduling tree for the task set Γ_2.

on another. We also assume that there is a deterministic dependence. That is, a job entering high criticality does not cause a probable but deterministic high criticality mode of another job. Since the job execution is described by pWCET and dependence effectively causes probabilistic delay in execution, the dependence affects the pWCRT of the job. We quantify this as follows.

A job J' is said to have dependence from another job J, $a \leq a'$, if J entering high criticality mode causes J' to enter high criticality mode. This is represented as $J \rightarrow J'$. This dependence affects the pWCRT $f'_{\mathcal{R}}$ of the job J' and results in a dependent pWCRT $f'^{d}_{\mathcal{R}_j}$.

Theorem 2 (Dependent Response Time)**.** *The pWCRT of a high criticality job J' having dependence from another high criticality job J is a piece-wise PMF with the pWCRT of J' until the criticality threshold l' and convolution between tails of pWCRTs of J and J' beyond the criticality threshold.*

Proof. Job J has dependence on job J'. If the response time of J crosses the threshold l, J' enters high criticality. Now, there exists a probability that the job J as well as J' crosses their respective thresholds. Job J' enters high criticality if it exceeds its threshold or J enters high criticality. That is,

$$Event(J' \text{ enters HI-criticality}) = Event(J \text{ enters HI-criticality }) \text{ or } Event(J' \text{ enters HI-criticality of its own execution})$$

The execution of J is given by its response time and that of J' is given by its pWCET. Let the repsonse time random variable be $\mathcal{R}$ for J, response time random variable $\mathcal{R}'$ for J' and execution time random variable for $\mathcal{C}'$ for J', let the dependent response time be $f'^{d}_{\mathcal{R}}$. Since execution of J' does not affect the

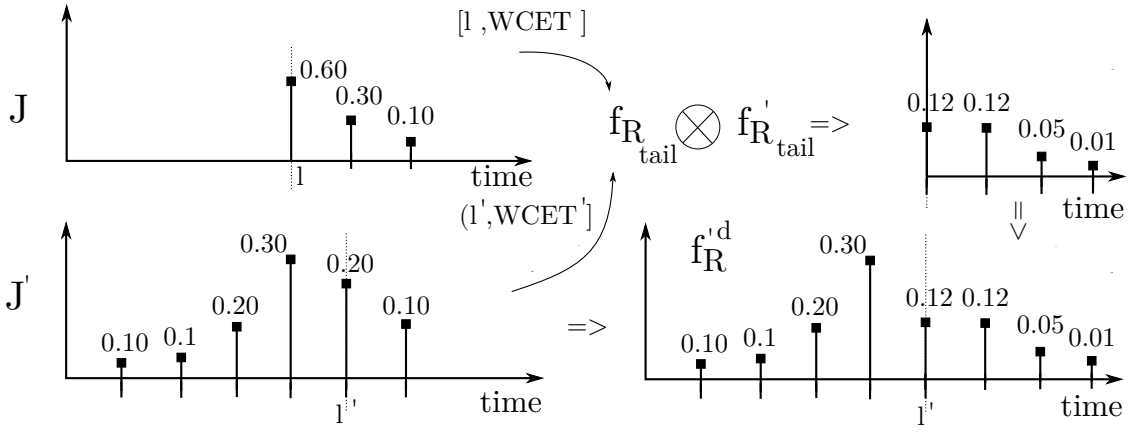


Figure 6.11: Dependence convolution.

duration for which J will execute, $\mathcal{R}$ and $\mathcal{C}'$ can be assumed independent. The probabilities are thus,

$$Pr(\mathcal{R}' > l') = Pr(\mathcal{R} > l + \mathcal{C}' > l')$$

The summation leads to a convolution. The values of $\mathcal{R}'$ less than or equal to l' are not affected by $\mathcal{R}$. That is, if the execution of J' ends before or at l', job J entering HI-criticality does not affect J'. Therefore, only the probabilities after l' or the tail of J' needs to be convolved. This leads to a piece-wise response time distribution. The dependent response time $f^{'d}_{\mathcal{R}}$ is thus given as

$$f^{'d}_{\mathcal{R}}(x) = \begin{cases} f_{\mathcal{R}}(x) & \text{if } x \leq l' \\ f^{\prime(l',WCET']}_{tail} \otimes f^{[l,WCET]}_{tail} & \text{if } x > l' \end{cases}$$

It should be noted that $(l', WCET']$ is different from $[l', WCET']$ by not including the probability at l' in tail PMF. Same as the previous definition of response time, we do no lose any probabilistic information by this operation. This is because the probabilities until l' are included in the piece of the function with $x \leq l'$. The probabilities after the threshold are convolved. This is shown in Figure 6.11 by assuming a PMF for the jobs in task set Γ_1. The dependent pWCRT $f^{'d}_{\mathcal{R}}(x)$ is formed piece-wise from the pWCRT of the jobs J and J'.

□

It should be noted that the dependent pWCRT is obtained after obtaining the pWCRT. This way, first the delay caused due the previously scheduled job is included, followed by any dependence. We reuse the task Γ_2 to exhibit this.

Example 11. *Referring to the task set Γ_2, say there exists dependence from J_{11} to the job J_{12}. In that case the pWCRT for J_{12} is shown in Figure 6.12. We see that the pWCRT changes for J_{12} in the dependent case by accommodating for the probabilities from the job J_{11}. The same schedule remains feasible which is: $J_{11}, J_{41}, J_{31}, J_{32}, J_{21}, J_{12}, J_{13}, J_{42}, J_{33}, J_{22}, J_{14}, J_{15}, J_{34}, J_{23}, J_{16}$.*

Because of dependence, the probability that the system enters high criticality is 0.006005 *which is*

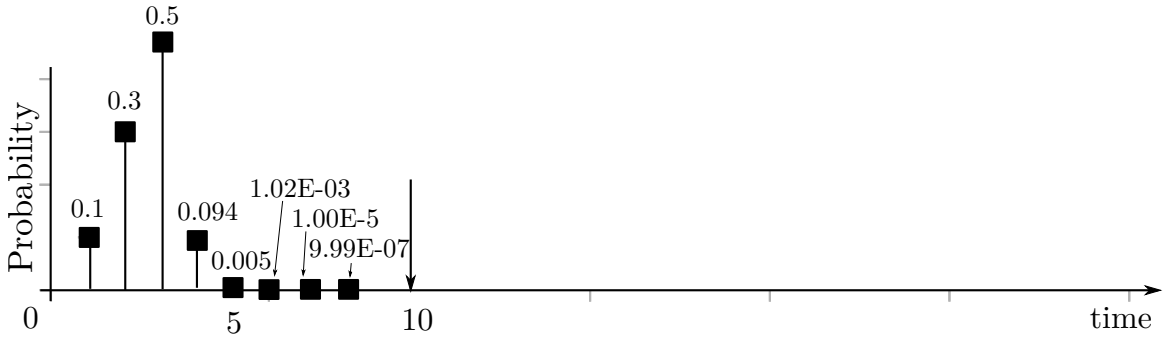


Figure 6.12: Dependent pWCRT PMF of job J_{12} in Γ_2 .

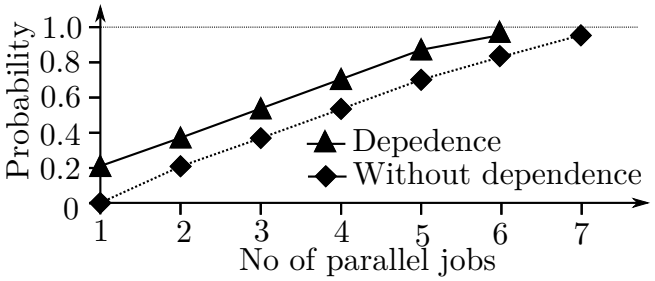


Figure 6.13: Probability of system entering high criticality

higher than 0.0050 *in the non-dependent case.*

In order to understand the relation between dependence and the system probabilities, we analyze a set of jobs released at a critical instant, i.e. their arrival times are equal. The jobs characteristics are same as that of J_{11} *in* Γ_2*. The dependence exists from the first job to all the subsequent jobs. Figure 6.13 shows the affect of this dependence on the probability of system entering high criticality as the number of jobs increases. The rate of increase in probability with number of jobs is same as the non-dependent case with dependence probabilities being higher. The probability of deadline miss of the jobs are shown in Figure 6.14. We see the cascade affect of the dependence in addition to the parallel execution of jobs resulting in higher probabilities of deadline miss. They quickly cross the maximum allowed limit of* $1E-03$ *into the region in which they are no more schedulable. This will be reflected in the graph model as no valid path will exist, implying that a feasible schedule will not exist.*

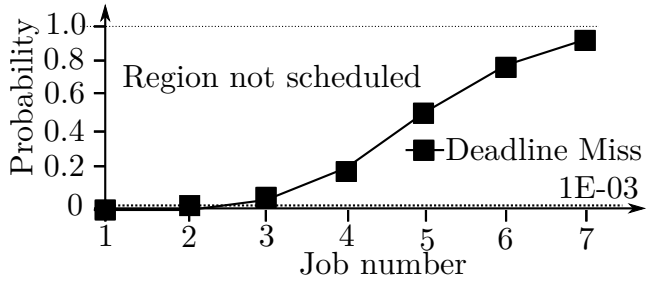


Figure 6.14: Probability of jobs missing deadline

It should be noted that we evaluate probabilistic dependence, i.e. the probability affected by dependence. We quantify the probabilistic effect on the response time of the jobs which have dependence from others. We do not need to apply it on the Worst Case model because the worst case model already considers the worst cases of jobs entering high criticality mode. The probability of job entering high criticality mode is unlikely but possible. The worst case model incorporates these unlikely occurences, i.e. the schedules obtained from worst case model ensures timing constraints for all the high criticality jobs.

6.4 Perspectives

We have utilized graph base exploratory method to obtain a schedule for Mixed Criticality probabilistic real-time system. The task criticalities are defined using the response time because tasks enter higher criticality by demanding more resource at run-time. This is as opposed using the execution time for task criticalities. The obtained schedule minimizes the probability of system entering high criticality mode. We extended this model in the deterministic worst case scenario where system does enter high criticality. In this worst case, the maximum number of low criticality jobs are allowed to execute along with high criticality ones without jeopardizing the schedulability of either. This way we have maximized resource usage through scheduling. We also take a first step towards studying task dependence in the probabilistic MC system. We will further explore the dependence in the mixed criticality environment in the future. We also aim to perform a sensitivity analysis on the criticality definition using the task response times.

In this final chapter of the thesis, we have tackled most of the problems we realized in the Chapter 3. We used discrete method of modelling through graphs to model the job execution scenarios. It is the method of modelling which is discrete and not the distributions. This method can take both discrete and continuous distributions as the input. The conversion from continuous and discrete distribution has been discussed in Chapter 3.

We made the following observation. The mixed criticality model represents an adjustment to a task demanding more execution time. This demand occurs at run-time and is not known beforehand. Since response time represents the run-time behaviour of the task, the response time should be used to decide the system criticality for schedulability analysis. This observation leads to a probabilistic response time in our case, which in turn gives us a probability of jobs entering higher criticality. This probability now depends on the schedule because response time depends on the schedule as well. This way probability is brought into the decision making process.

We obtain a schedule and not remain affixed to EDF or FP, like in Chapter 3. The schedule obtained minimizes the probability of system entering higher criticality. This way, the schedule is subject to a probability criteria. Since the schedule uses probability as well as MC, we have combined both the approaches and quantified the pessimism.

Once we have minimized the probability of system entering high criticality, we use the modelling in Chapter 5 to provide the bounds to this probability. That is, we prepare for the scenario that the system actually does enter high criticality. In that case, we ensure the schedulability of the high criticality jobs.

We do so without dropping all the lower criticality jobs. Here we obtain a schedule as a tree, as a result the schedule adapts depending on whether a job enters high criticality or not.

Where we lack at the current stage is the ability to handle high complexity of our approach. In the previous chapters we discussed where the complexity arises from. We decided to take the course for higher complexity further to ensure safety of the schedule and the schedulability analysis. We intend to reduce the complexity in the future works. Where we gain is the higher representativity of the execution of jobs of different criticalities. We gain a flexible representation which also gives us more control over job executions depending on the states of the system.

CHAPTER 7

General Perspectives and Conclusion

> Simplicity is the final achievement. After one has played a vast quantity of notes and more notes, it is simplicity that emerges as the crowning reward of art.
>
> *Frédéric Chopin*

In the course of this thesis we obtained perspectives in many directions relating to pRTSs and MC systems. The objective of this thesis was to perform schedulability analysis of probabilistic real-time systems. Through our discussion, this objective transformed into quantifying and minimizing pessimism in the analysis of the system. This is because the real-time system is applied in the real-world with design constraints and over-allowance of resources. This over-allowance is quantified using probabilistic representations of the system.

The task execution is described using a pWCET which is a probability distribution. The pWCETs are assumed given. The tasks are scheduled to ensure that all tasks are allowed a processor time. In order to ensure that all tasks are scheduled, schedulability analysis is performed. In our case when task execution is described probabilistically, the schedulability analysis must also be probabilistic. This implies that the probabilistic modelling of the system to probabilistically ensure that all tasks are scheduled.

In Chapter 3, we began with a formal approach towards the probabilistic analysis of the system through CTMCh. We used CTMCh to models jobs of the tasks which are scheduled using EDF or FP scheduling algorithm. We assumed continuous pWCET distribution. We observed the differences between continuous and discrete pWCET distributions with the advantages and disadvantages of each. Following this, we approached another direction in search for reducing pessimism through MC real-time systems.

In Chapter 4, we obtained a DTMCh model for a MC pRTS. DTMCh model was used to quantify the probability of the system entering high criticality. At this moment, we observed that pessimism can be further reduced in a probabilistic environment by letting go of the classical schedulability algorithms. Moreover, the objective of the probabilistic as well as the MC approaches is to reduce the resource over-allowance. These two approaches can be combined toghether to further refine our research.

In Chapter 5, we obtained a graph based exploratory model for MC pRTS. Using the graph model, we obtained a schedule which is safe and ensures schedulability of high criticality jobs. At the same time, the low criticality jobs are scheduled whenever possible. Through this approach, we dropped the

classical idea of suspending all low criticality jobs when a high criticality job enters high criticality mode. At the same time we observed that the probabilities can only be quantified because they arrive from pWCET and pWCET does not depend on the schedule.

In the final Chapter 6, we made the observation that tasks enters high criticality at runtime. This implies that response time, and not execution time, must be used to decide criticality modes. Since the response time is affected by the schedule.The schedule can then be subject to a probability minimization or a maximization criteria. We observed the usability and hidden assumptions in the classical convolution operation between pWCET distributions in the context of real-time systems. In the end we obtain a MC schedule in which the probability of system entering high criticality is minimum. At the same time, if the system does enter high criticality, the schedulability of high criticality jobs is ensured with allowing low criticality jobs to execute. The schedule is also adaptive, which means that depending on the when and which job enters high criticality, the following schedule is decided accordingly. The schedule is ensured in the worst case. We also made a first step towards dependence between the tasks which is apart from the scheduling.

Globally, probabilities have the potential to quantify the pessimism which exists in real-time scheduling. With probabilities, many safety concerns arise. As a a general rule, pessimistic probabilities are deemed safe and acceptable instead of optimistic probabilities even if it leads to resource over-allowance. We assumed a pWCET given and we did not enter the details of obtaining a pWCET. However, the pWCET estimations and the methods of obtaining them must be ensured to be safe.

Care must be taken if the pWCET is continuous or discrete. At the same time, if it is required to transform from one to the other, it must ensured that no information is lost in the process and the transformation is safe. Continuous distribution represented by mathematical expressions can be evaluated with less complexity. On the other hand, continuous distributions may lead to representation problems. Discrete distributions are accurate is representing probabilities with respect to an instant of time. Discrete distributions are preferable because the probabilities are available for every possible occurrence at the instant of its occurrence. However, discrete probabilities lead to large computational complexity. Eventually, because of high information content in the probabilistic representations, probabilistic analysis leads to high computational complexity.

MC approaches tend to reduce the pessimism in the system by assigning criticalities and criticality modes to the system. These approaches should be more flexible in resource usage. For example, the classical approach to drop all low criticality tasks when system enters high criticality does not take into consideration any possible extra room for execution. It is understandable because it is a simple solution which has one system parameter of criticality mode and one simple action to cope with it. However, to increase resource usage efficiency, the idea of system-wide mode switch should be dropped. On the other hand, an adaptive schedule, as one presented in this thesis, should be implemented.

Another classical concept is that of defining MC using task execution time. Since higher execution demand by a task happens at runtime, and criticality is formalization of this demand, the run time information must be used to decide for criticality. That is, response time instead of execution time must be used to decide for task criticality mode. A dual effect of this idea is that the probabilities associated to the criticality get affected by the schedule. This way, the probabilities of task entering high criticalities can be used in the deciding for a MC pRTS schedule. Otherwise, in the case where

probabilities come from execution time, the probabilities can only be quantified as they are not affected by the schedule.

Probabilistic analysis must always be accompanied by the worst case probabilities. That is, similar to the case that pWCET is upper bounded by WCET, the probabilistic results from the schedulability analysis must always be upper bounded. This way, the schedulability analysis is more rich than a deterministic one. The probabilistic schedulability analysis not only explores the worst case scenarios, it also gives a picture of task execution before the worst cases.

Overall, if some classical approaches be overlooked, a combination of probabilistic and MC approached have the potential to provide a rich, safe and efficient real-time system.

In the future works, we would like to reduce the complexity of the MC pRTS graph based approaches. We plan to do so through more organized search as well as through merging similar paths in the exploration tree. We would like to further develop analysis of task dependence in the system. The next steps should be taken in the general direction of studying complexity of the MC approaches. Criticality levels based on probabilities can directly relate to some safety standards and this will be studied in the future.

Bibliography

[Ab+16] Zaid Al-bayati et al. “A Four-Mode Model for Efficient Fault-Tolerant Mixed-Criticality Systems”. In: Jan. 2016, pp. 97–102 (cit. on p. 20).

[Abe+14] Jaume Abella et al. “Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA”. In: *26th Euromicro Conference on Real-Time Systems, (ECRTS)*. 2014 (cit. on p. 18).

[AD17] Yasmina Abdeddaïm and Maxim Dorin. “Probabilistic Schedulability Analysis for Fixed Priority Mixed Criticality Real-Time Systems ”. In: *Design, Automation and Test in Europe - DATE 2017*. Lausanne, Switzerland, Mar. 2017 (cit. on p. 19).

[AG16] Bader Alahmad and Sathish Gopalakrishnan. “A Risk-Constrained Markov Decision Process Approach to Scheduling Mixed-Criticality Job Sets”. In: *Workshop on Mixed Criticality Systems (WMC 2016)*. Porto, Portugal, Nov. 2016 (cit. on p. 20).

[AMP12] Luca Abeni, Nicola Manica, and Luigi Palopoli. “Efficient and robust probabilistic guarantees for real-time tasks”. In: *Journal of Systems and Software* 85.5 (2012), pp. 1147–1156 (cit. on p. 18).

[Bar+11] Sanjoy K. Baruah et al. “Mixed-Criticality Scheduling of Sporadic Task Systems”. In: *19th Annual European Symposium on Algorithms (ESA 2011)*. Ed. by Camil Demetrescu and Magnús M. Halldórsson. Vol. 6942. Lecture Notes in Computer Science. Saarbruecken, Germany: Springer, Sept. 2011, pp. 555–566 (cit. on p. 20).

[Bar+14] Sanjoy Baruah et al. “Mixed-criticality scheduling on multiprocessors”. In: *Real-Time Systems* 50 (May 2014) (cit. on p. 20).

[Bar+15] Sanjoy K. Baruah et al. “Preemptive Uniprocessor Scheduling of Mixed-Criticality Sporadic Task Systems”. In: *J. ACM* 62.2 (2015) (cit. on pp. 56–58).

[Bar09] Sanjoy K. Baruah. “Mixed criticality schedulability analysis is highly intractable”. In: 2009 (cit. on p. 19).

[Bar98] S. K. Baruah. “A General Model for Recurring Real-Time Tasks”. In: *Real-Time System* (1998) (cit. on p. 17).

[BB13] S.K. Baruah and A. Burns. “Fixed-priority scheduling of dual-criticality systems”. In: *Proc. RTNS*. ACM, 2013, pp. 173–182 (cit. on p. 20).

[BBB03] A. Burns, G. Bernat, and I. Broster. “A Probabilistic Framework for Schedulability Analysis”. In: (2003), pp. 1–15 (cit. on p. 18).

[BBD11] S. K. Baruah, A. Burns, and R. I. Davis. “Response-Time Analysis for Mixed Criticality Systems”. In: *Proceedings of the 2011 IEEE 32Nd Real-Time Systems Symposium*. RTSS ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 34–43 (cit. on p. 20).

[BCP02] G. Bernat, A. Colin, and S. M. Petters. “WCET Analysis of Probabilistic Hard Real-Time System”. In: *IEEE Real-Time Systems Symposium (RTSS)*. 2002 (cit. on p. 16).

[BD19] Alan Burns and Robert Davis. *Mixed criticality systems – a review (12th ed.)* Tech. rep. Dept of CS, U. of York, UK, 2019 (cit. on p. 19).

[BD91] B. Berthomieu and M. Diaz. "Modeling and Verification of Time Dependent Systems Using Time Petri Nets". In: *IEEE Trans. on SW Eng.* 17.3 (1991), 259–273 (cit. on p. 17).

[BEG15] S. Baruah, A. Easwaran, and Z. Guo. "MC-Fluid: Simplified and Optimally Quantified". In: *2015 IEEE Real-Time Systems Symposium*. 2015, pp. 327–337 (cit. on p. 20).

[BG15] Sanjoy K. Baruah and Zhishan Guo. "Mixed-Criticality Job Models: A Comparison". In: 2015 (cit. on p. 19).

[Bhu+19] Ashik Bhuiyan et al. "Precise Scheduling of Mixed-Criticality Tasks by Varying Processor Speed". In: Nov. 2019 (cit. on p. 20).

[BLHS91] Shaukat Brah, John L Hunsucker, and Jaymeen Shah. "Mathematical Modeling of Scheduling Problems". In: *International Journal of Information and Management Sciences* 12 (Jan. 1991), pp. 113–137 (cit. on p. 16).

[Buc+10] G. Bucci et al. "Oris: a tool for modeling, verification and evaluation of real-time systems". In: *International Journal of Software Tools for Technology Transfer* 12.5 (2010), 391 –403 (cit. on pp. 18, 48).

[Bur15] A. Burns. "An Augmented Model for Mixed Criticality". In: *Mixed Criticality on Multicore/Manycore Platforms (Dagstuhl Seminar 15121)*. Ed. by Davis Baruah Cucu-Grosjean and Maiza. Vol. 5. 3. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 92–93 (cit. on p. 19).

[But11] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. 3rd. Springer Publishing Company, Incorporated, 2011 (cit. on pp. 16, 29, 30, 37).

[Car+14] Laura Carnevali et al. "Probabilistic Deadline Miss Analysis of Real-Time Systems Using Regenerative Transient Analysis". In: *Int. Conf. Real-Time Networks and Sys. RTNS*. 2014 (cit. on p. 18).

[Caz+11] F.J. Cazorla et al. "PROARTIS: Probabilistically Analysable Real-Time Systems". In: *ACM Transactions on Embedded Computing Systems* (2011) (cit. on pp. 19, 33).

[CG+12] L. Cucu-Grosjean et al. "Measurement-Based Probabilistic Timing Analysis for Multi-path Programs". In: *Euromicro Conf. on Real-Time Systems (ECRTS)*. IEEE, 2012 (cit. on pp. 18, 51).

[CG13] Liliana Cucu-Grosjean. "Independence - a misunderstood property of and for (probabilistic) real-time systems". In: *Real-time Systems: the past, the present and the future*. 2013 (cit. on pp. 17, 26).

[CGV09] L. Carnevali, L. Grassi, and E. Vicario. "State-density functions over DBM domains in the analysis of non-Markovian models". In: *IEEE Trans. on SW Eng.* 35.2 (2009), 178–194 (cit. on p. 18).

[CKT94] Hoon Choi, Vidyadhar G. Kulkarni, and Kishor S. Trivedi. "Markov regenerative stochastic Petri nets". In: *Performance Evaluation* 20.1-3 (1994), pp. 337–357 (cit. on p. 19).

[CT06] Liliana Cucu and Eduardo Tovar. “A framework for the response time analysis of fixed-priority tasks with stochastic inter-arrival times”. In: *SIGBED Review* 3.1 (2006), pp. 7–12 (cit. on p. 17).

[DBG17] Robert I. Davis, Alan Burns, and David 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 (cit. on p. 17).

[DD99] A. L. M. Dekkers and L. De Haan. “On the Estimation of the Extreme-Value Index and Large Quantile Estimation”. In: *The Annals of Statistics* 17.4 (1999), pp. 1795–1832 (cit. on p. 18).

[Dia+02] José Luis Diaz et al. “Stochastic Analysis of Periodic Real-Time Systems”. In: *RTSS: Proceedings of the 23rd IEEE Real-Time Systems Symposium*. IEEE Computer Society, 2002, p. 289 (cit. on p. 17).

[Dia+04] J. L. Diaz et al. “Pessimism in the stochastic analysis of real-time systems: concept and applications”. In: *25th IEEE International Real-Time Systems Symposium*. 2004, pp. 197–207 (cit. on pp. 18, 32).

[EE07] Andreas Ermedahl and Jakob Engblom. “Execution Time Analysis for Embedded Real-Time Systems”. In: ed. by Sang H. Son Insup Lee Jospeh Y-T. Leung. Chapman & Hall/CRC - Taylor and Francis Group, 2007 (cit. on p. 16).

[GSY15] Z. Guo, L. Santinelli, and K. Yang. “EDF schedulability analysis on mixed-criticality systems with permitted failure probability”. In: *Proc. RTCSA*. 2015 (cit. on pp. 19, 57).

[Guo+17] Zhishan Guo et al. “Sustainability in Mixed-Criticality Scheduling”. In: *2017 IEEE Real-Time Systems Symposium (RTSS)* (2017), pp. 24–33 (cit. on p. 20).

[HHM09] J. Hansen, S. Hissam, and G. A. Moreno. “Statistical-Based WCET Estimation and Validation”. In: *Int. Workshop on WCET Analysis*. 2009 (cit. on p. 18).

[Hor+12] András Horváth et al. “Transient analysis of non-Markovian models using stochastic state classes”. In: *Perf. Eval.* 69.7-8 (2012), pp. 315–335 (cit. on p. 19).

[Int09] Michael Interbartolo. *Apollo Guidance, Navigation, and Control (GNC) Hardware Overview*. Tech. rep. 2009 (cit. on p. 16).

[Kim+05] Kanghee Kim et al. “An exact stochastic analysis of priority-driven periodic real-time systems and its approximations”. In: *Computers, IEEE Transactions on* 54.11 (2005), pp. 1460 –1466 (cit. on p. 88).

[KNP07] M. Kwiatkowska, G. Norman, and D. Parker. “Stochastic Model Checking”. In: *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation SFM*. Ed. by M. Bernardo and J. Hillston. Vol. 4486. LNCS (Tutorial Volume). Springer, 2007, pp. 220–270 (cit. on pp. 31, 32).

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. “PRISM 4.0: Verification of Probabilistic Real-time Systems”. In: *Proc. 23rd International Conference on Computer Aided Verification CAV*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. LNCS. Springer, 2011 (cit. on pp. 13, 19, 58).

[LDB16] George Lima, Dario Dias, and Edna Barros. “Extreme Value Theory for Estimating Task Execution Time Bounds: A Careful Look”. In: *28th Euromicro Conference on Real-Time Systems, (ECRTS)*. 2016 (cit. on p. 33).

[Liu+16] Di Liu et al. “EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees”. In: *Proceedings of the 37th IEEE Real-Time Systems Symposium*. 2016 (cit. on p. 20).

[LL73] C. L. Liu and James W. Layland. “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”. In: *J. ACM* 20.1 (Jan. 1973), pp. 46–61 (cit. on p. 17).

[LR09] Didier Lime and Olivier H. Roux. “Formal verification of real-time systems with preemptive scheduling”. In: *Real-Time Syst.* 41.2 (2009), pp. 118–151 (cit. on p. 17).

[Lu+12] Yue Lu et al. “A Statistical Response-Time Analysis of Real-Time Embedded Systems”. In: *Proceedings of the 33rd IEEE Real-Time Systems Symposium, RTSS 2012, San Juan, PR, USA, December 4-7, 2012*. 2012, pp. 351–362 (cit. on p. 18).

[Max+17] Dorin Maxim et al. “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. Grenoble, France: ACM, 2017, pp. 237–246 (cit. on pp. 20, 88).

[MBP17] Roberto Medina, Etienne Borde, and Laurent Pautet. “Directed Acyclic Graph Scheduling for Mixed-Criticality Systems”. In: *Ada-Europe*. 2017 (cit. on p. 20).

[MEP04] S. Manolache, P. Eles, and Z. Peng. “Schedulability analysis of applications with stochastic task execution times”. In: *ACM Trans. Emb. Comp. Sys.* 3.4 (2004), pp. 706–735 (cit. on p. 18).

[MKT04] Alexander Maxiaguine, Simon Kunzli, and Lothar Thiele. “Workload Characterization Model for Tasks with Variable Execution Demand”. In: *DATE ’04: Proceedings of the conference on Design, automation and test in Europe*. IEEE Computer Society, 2004 (cit. on p. 17).

[Moo65] Gordon E. Moore. “Cramming more components onto integrated circuits”. In: *Electronics* 38.8 (1965) (cit. on p. 16).

[Nor97] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997 (cit. on pp. 30, 75, 89).

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. John Wiley & Sons, Inc., 1994 (cit. on p. 58).

[SB94] Spuri and Buttazzo. “Efficient aperiodic service under earliest deadline scheduling”. In: *1994 Proceedings Real-Time Systems Symposium*. 1994, pp. 2–11 (cit. on p. 17).

[SG16] Luca Santinelli and Laurent George. “Probabilities and Mixed-Criticalities: the Probabilistic C-Space”. In: *3rd International Workshop on Mixed Criticality Systems WMC at RTSS*. 2016 (cit. on p. 18).

[SGM17] Luca Santinelli, Fabrice Guet, and Jérôme Morio. “Revising Measurement-Based Probabilistic Timing Analysis”. In: *2017 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2017, Pittsburg, PA, USA, April 18-21, 2017*. 2017, pp. 199–208 (cit. on p. 33).

[SM01] William H Sanders and John F Meyer. “Stochastic Activity Networks: Formal Definitions and Concepts”. In: *Lectures on Formal Methods and Performance Analysis*. Springer, 2001, pp. 315–343 (cit. on p. 19).

[Sti+11] Martin Stigge et al. “The Digraph Real-Time Task Model”. In: *Proceedings of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*. RTAS ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 71–80 (cit. on p. 17).

[Sto02] Mariëlle Stoelinga. “An introduction to probabilistic automata”. In: *Bulletin of the European Association for Theoretical Computer Science* (2002), pp. 176–198 (cit. on p. 17).

[TBW92] K. W. Tindell, A. Burns, and A. J. Wellings. “Mode changes in priority pre-emptively scheduled systems”. In: *Proceedings of the Real Time Systems Symposium*. 1992, pp. 100–109 (cit. on p. 19).

[TF13] Jens Theis and Gerhard Fohler. “Schedule Table Generation for Time-Triggered Mixed Criticality Systems”. In: 2013 (cit. on p. 19).

[Ves07] Steve 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)*. IEEE Computer Society, 2007, pp. 239–243 (cit. on pp. 19, 57).

[Vic01] E. Vicario. “Static analysis and dynamic steering of time dependent systems using time Petri nets”. In: *IEEE Trans. on SW Eng.* 27.1 (2001), 728–748 (cit. on p. 16).

[VMF03] Manel Velasco, Pau Marti, and Josep M. Fuertes. *The Self Triggered Task Model for Real-Time Control Systems*. 2003 (cit. on p. 16).

[Wil+08] R. Wilhelm et al. “The worst-case execution-time problem - overview of methods and survey of tools”. In: *ACM Trans. Embedded Comput. Syst.* (2008) (cit. on p. 16).

[ZB09] Fengxiang Zhang and Alan Burns. “Schedulability Analysis for Real-Time Systems with EDF Scheduling”. In: *IEEE Transactions on Computers* 58.9 (2009) (cit. on p. 17).

[A. 95] A. Bobbio and M. Telek. “Markov regenerative SPN with non-overlapping activity cycles”. In: *Int. Comp. Perf. and Depend. Symp. (IPDS95)* (1995), 124–133 (cit. on p. 19).

[Bak+09] S. Bak et al. “The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety”. In: *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*. 2009, pp. 99–107 (cit. on p. 19).

[G. 04] G. Bucci and A. Fedeli and L. Sassoli and E. Vicario. “Timed State Space Analysis of Real Time Preemptive Systems”. In: *IEEE Trans. on SW Eng.* 30.2 (2004), 97–111 (cit. on p. 17).

[Kim05] Kim G. Larsen and Marius Mikucionis and Brian Nielsen and Arne Skou. “Testing Real-time Embedded Software using UPPAAL-TRON - An Industrial Case Study”. In: *ACM Int. Conf. on Embedded SW*. 2005 (cit. on p. 17).

[Oli04] Olivier H. Roux and Didier Lime. “Time Petri nets with inhibitor hyperarcs: formal semantics and state-space computation”. In: *25th Int. Conf. on Theory and Application of Petri Nets* 3099 (2004), 371–390 (cit. on p. 17).

[Sta88] J. A. Stankovic. “Misconceptions about real-time computing: a serious problem for next-generation systems”. In: *Computer* 21.10 (1988), pp. 10–19 (cit. on p. 16).