Modélisation et analyse de la criticité mixte pour le Systèmes embarqués temps réel

Luca Santinelli

Institut National Polytechnique de Toulouse

Habilitation à diriger les recherches
préparée au

Départment Traitement de l'Information et Systèmes de l'ONERA

présente et soutenue publiquement,
le 20/05/2019, par

LUCA SANTINELLI

Title: Mixed Criticality Modeling and Analysis Paradigms for Real-Time Embedded Systems

ONERA
THE FRENCH AEROSPACE LAB

Jury:

Mme. Isabelle Puaut Rapporteur Professeur Université de Rennes I (France)

M. Alan Burns Rapporteur Professeur University of York (UK)

M. Joel Goossens Rapporteur Professeur Université Libre de Bruxelles (Belgique)

M. Luis Almeida Examinateur Professeur Universidade do Porto (Portugal)

M Laurent George Examinateur Professeur ESIEE Paris (France)

M Yves Sorel Examinateur Directeur de recherche INRIA Paris (France)

M Christian Fraboul Examinateur Professeur INP Toulouse ENSEEIHT (France)

To my family, patient and supportive in the constant pursuit of the next step.

Preamble

This manuscript motivates and details the research project chosen for investigating today's and future real-time embedded systems. In it, the research activities as well as the research perspectives are described in order to outline project challenges, achievements already made, and future works. The manuscript and the research project are also for preparing and defending the "Habilitation Diriger des Recherches" (HDR).

As research project, it has been chosen to investigate mixed criticality real-time embedded systems. In few words, the project aims at guaranteeing timing constraint and schedulability of applications with different requirements/criticality that are running together. Mixing criticality tries to reconcile efficient resource usage and safety assurance, thus it is critical with today's and future multi-core and many-core implementations for real-time embedded systems. It is a complex problem and has some interesting open problems that requires to be studied.

The project is presented with respect to works already made, and more importantly with perspectives that will be elaborated with future achievements. Previous research on non-mixed critical real-time embedded systems for timing analysis and schedulability analysis is also described; it is background work for mixed criticality achievements.

The manuscript is organized into three parts: i) the first is for presenting the professional experience made before the PhD as well as the academic experience in the real-time community made since the PhD; ii) the second is for detailing with some publications the research topics approached so far and the research problem under investigation; iii) the third, concludes the manuscript with observations on works already made for the research project, and ideas for future work.

The first part of the document is where it is detailed the curriculum vitae from the Master thesis up until today. In it, there are presented the training, the different positions hold, and every involvement in the scientific community. There are also listed the students supervised, the research and technology transfer projects worked on, and the collaborations made.

The second part is more technical since there is presented the past research and some early works on mixed criticality. The past works do not approach the mixed criticality problem, but are used as background for developing solutions for mixed criticality. Moreover, there are presented recent works which contributes to timing analysis and schedulability analysis for mixed criticality.

The third part concludes the manuscript with a discussion on the mixed criticality research project. There are illustrated limitations of already existing solutions, and there are discussed future research for more effective resource usage, for reducing pessimism and complexity, and for flexible safety guarantees. In particular, it is discussed the opportunities as well as the challenges that multi- and many-core platforms brings to mixed criticality.

The research project is the results of the twofold motivation from professional as well as academic experience made on real-time embedded systems. In it, both academic and industrial perspectives to mixed criticality are considered with solutions that will benefit both. The achievements will continue to come from research and technology transfer projects, from the collaborations with academic and industrial partners, and from the supervision of PhD as well as Master students.

Contents

Table of acronyms

ASIL	Automotive Safety Integrity Level
BCET	Best-Case Execution Time
BM	Block Maxima
CDF	Cumulative Distribution Function
COTS	Commercial Off The Shelf
CTMC	Continuous Time Markov Cahin
CUDA	Compute Unified Device Architecture
DAG	Directed Acyclic Graph
DAL	Design Assurance Level
diagXtrm	DIAGnostic tool for estimating the pWCET with the eXTReMe value theory
DM	Deadline Monotonic
DTMC	Discrete Time Markov Cahin
EDF	Earliest Deadline First
EDF-VD	Earliest Deadline First Virtual Deadline
EI	Extremal Index
ET	Execution Time
ETP	Execution Time Profile
EVT	Extreme Value Theory
FP	Fixed Priority
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GEV	Generalized Extreme Value
GPD	Generalized Pareto Distribution
GPU	Graphical Processor Unit
ICDF	Inverse Cumulative Distribution Function
KPSS	Kwiatowski Phillips Schmidt Shin
MBPTA	Measurement Based Probabilistic Timing Analysis
MC	Mixed criticality
Mc	Markov chain
MCFS	Mixed-Criticality Federated Scheduler
MDA	Maximum Domain of Attraction
MIT	Minimum Inter-arrival Time
MLE	Maximum Likelihood Estimator
NC	Network Calculus
NOC	Network On Chip
OCBP	Own-Criticality-Based-Priority
POT	Peaks Over Threshold
pCalculus	probabilistic Calculus
pCRPD	probabilistic Cache Related Preemption Delays
pET	probabilistic Execution Time
pRTS	probabilistic Real-Time system
pWCET	probabilistic Worst-Case Execution Time
pWCRT	probabilistic Worst-Case Recution Time
RM	Rate Monotonic
RR	Round Robin
RTC	Real-Time Calculus
RTOS	Real-Time Operating System
SPTA	Static Probabilistic Timing Analysis
STA	Static Timing Analysis
TDMA	Time Division Multiple Access
WCET	Worst-Case Execution Time

Chapter 1

Experience with real-time embedded systems

I am **PhD Luca Santinelli** research engineer at the DTIS-SEAS ONERA, and Assistant Professor at the Institut Superieur de l'Aeronautique et de l'Espace (ISAE). I work in Toulouse, 2 Edouard Belin 31400 Toulouse, France; phone: (+33) (0)5.62.25.28.60, fax: (+33) (0)5.62.25.25.93, email: luca.santinelli@onera.fr, and home page: `https://www.onera.fr/staff/luca-santinelli`.

Fellow research engineer at the ONERA, Office national d'études et de recherches aérospatiales – French AereospaceLab since 2012, working in the SEAS team within the DTIS department, traitement de l'information et systemes departement.

Graduated in Electronic Engineering, with specialization in Computer Science, at the Polytechnic University of Marche in 2003. Master thesis on artificial intelligence approaches implemented on embedded systems.

From 2003 to 2007, professional experience at the MTS Group (now ARISTON Thermo) as software developer in the R&D department. Specialized on microcontroller, control systems, and safety critical embedded systems.

In 2010, PhD in Computer Science, curriculum in Innovative Technologies of Info. & Com. Eng. and Robotics, at the Scuola Superiore Sant'Anna working on real-time systems at the RETIS laboratory. PhD thesis on adaptive resource mechanisms for real-time systems. During the PhD thesis, in 2008 − 2009, visiting student at the ETH Zurich working within the TIK laboratory. Research activity on real-time calculus, and real-time scheduling for adaptive real-time systems.

In 2011−2012, Postdoc at LORIA, a joint unit within INRIA Nancy-Grand Est, CNRS and Nancy University. Working within the TRIO team on probabilistic timing analysis and probabilistic schedulability analysis for real-time systems.

The main research interests include real-time operating systems, schedulability and scheduling algorithms, energy aware scheduling, dynamic and adaptive resource reservation/allocation policies, probabilistic real-time modeling and analysis, real-time controls, formal methods and artificial intelligence. The research activity at the ONERA involves modeling and analysis of avionic safety critical real-time systems, performance analysis of multi- and many-core platforms, and certification arguments for mixed critical applications.

1.1 Professional and academic experience

Here the list of steps taken since 1994, the year of high school graduation; the list is in chronological order.

- 10/1994 – 06/2003 - Master in Electronic Engineering with specialization in Computer Science at the Polytechnic University of Marche, Ancona Italy. The Master studies were mainly Mathematics and Physics in the first two years, then they become more specific to Electronic and Computer Science in the last three years. The Master thesis under the supervision of Prof. Guido Tascini has been defended in 06/2018;

it is entitled: ”Analysis and development of techniques for fingerprint matching”; Master thesis grade of 110/110 cum laude. In the thesis, have been proposed artificial intelligence algorithms such as neural networks, and applied for fingerprint recognition implemented with embedded systems. A tool has been developed and tested with real fingerprints, while in [181] the publication that recaps the achievements. During the Master, 03/1995 – 10/1997, it has been completed the Military service and as Policemen, at the time mandatory. It has been exploited in Italian cities like Torino, Roma, and Napoli.

- 10/2003 – 12/2006 - Professional experience as research engineer in the private sector for the MTS group (now Ariston Thermo) company, Fabriano Italy. Main activities were on embedded systems research and development, embedded system networks and protocol development, and certification issues for safety critical embedded systems like wall-hung boilers and heating systems. During that period, collaborations and research projects with ELCO Engineers, Chaffoteaux & Maury Engineers, TUV Netherlands for SW and HW certification, and Rinnai Japan Engineers.

- 01/2007 – 12/2010 - PhD candidate at the Scuola Superiore Sant’Anna, RETIS lab, Pisa Italy. The PhD program was on Innovative Technologies and focused on Embedded Systems and the thesis has been supervised by Prof. Giorgio Buttazzo. The PhD has been about schedulability analysis, component-based design of real-time embedded systems, and adaptive real-time and resource reservation. In 12/2010, the PhD degree with the thesis entitled: ”Adaptive Resource Reservation”. The thesis has been focused on resource management issues for real-time embedded systems; in there, the notion of adaptivity has been developed and applied to different applications. During the thesis period, collaborations with Marconi-Ericsson, the CNR in Pisa for the IPERMOB project, and other Scuola Superiore Sant’Anna departments. Also, the collaboration with ETH Zurich and Prof. Lothar Thiele while visiting the ETH for one year. In 2008 – 2009, during the PhD, the visit at the ETH for one year under the supervision of Prof. Lothar Thiele; the research visit insisted on component-based design of real-time embedded systems and the use of real-time calculus for modeling and schedulability analysis. Last 6 months of the thesis (06/2010–12/2010) has been spent visiting INRIA Nancy, and under the supervision of Ph.D Liliana Cucu-Grosjean. In that period, the beginning of works and contributions on EU FP7 project PROARTIS.

- 01/2011–05/2012 - Postdoc on ”Probabilistic Real-Time Systems” at the INRIA Nancy (LORIA lab), Trio Team, Nancy France. The Postdoc was under the supervision of PhD Liliana Cucu-Grosjean, and it has been focused on developing reliable probabilistic time modeling and efficient probabilistic schedulability analysis for real-time systems. During the Postdoc, the collaboration with Barcelona Supercomputing Center, University of Padova, RAPITA in York, and the University of York on the probabilistic timing analysis topic for real-time embedded systems. In the Postdoc period the contributions to EU research project like PROARTIS and TIMMMO-2-USE. In 2012, while Postdoc in Nancy, qualification to ”maitre du conference” in computer science, class ”Informatique 27”.

- 06/2012 – today - Research engineer at the ONERA Toulouse within the DTIM department first, now at the DTIS department. The research activities relate to real-time embedded systems for timing analysis and schedulability analysis. Safety critical and mixed critical [networked] real-time systems are investigated, efficient resource reservation mechanisms are proposed. The research activity couples with technology transfer for ONERA industrial partners mainly in the Avionic and Space domains; the projects with contributions are listed in the following. The collaborations in this period have been extended to French and International partners, both academic and industrial.

Past and present professional as well as academic experiences intertwine and contributes toward the *definition*, the *implementation*, and the *verification* of real-time embedded systems.

The professional approaches driven by practical needs such as cost reductions, low complexity, low computational costs, and resource efficiency. Besides, the academic approaches focus on worst-case modeling, and analytical approached to timing guarantees. Combining both ends up into a constant focus on practical motivations, and solutions which are able to trade efficiency with worst-case guarantees.

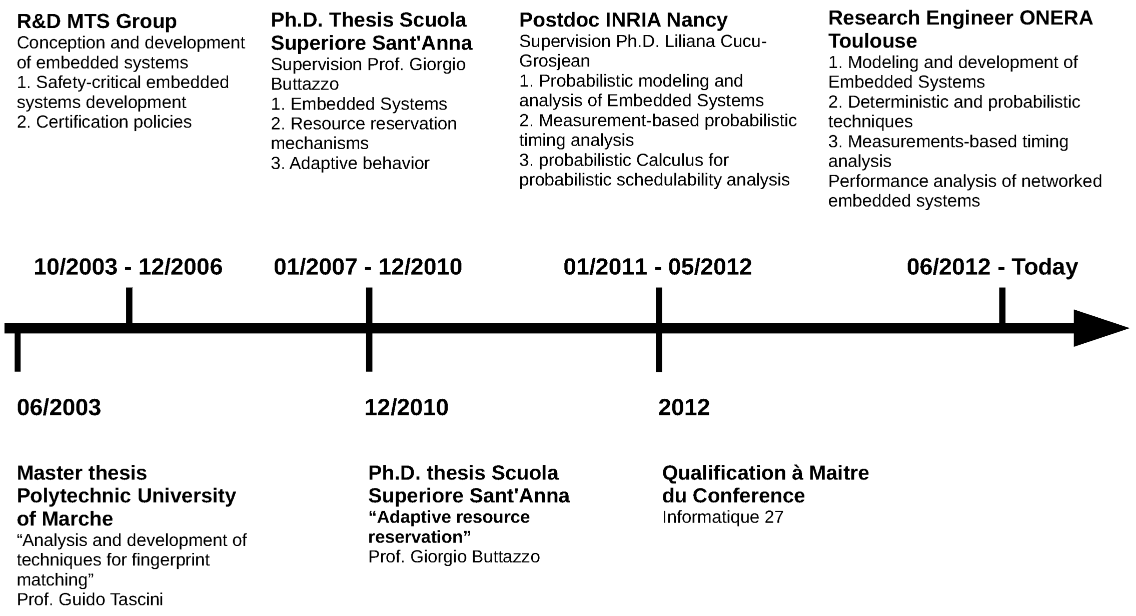

R&D MTS Group
Conception and development of embedded systems
1. Safety-critical embedded systems development
2. Certification policies
Ph.D. Thesis Scuola Superiore Sant'Anna
Supervision Prof. Giorgio Buttazzo
1. Embedded Systems
2. Resource reservation mechanisms
3. Adaptive behavior
Postdoc INRIA Nancy
Supervision Ph.D. Liliana Cucu-Grosjean
1. Probabilistic modeling and analysis of Embedded Systems
2. Measurement-based probabilistic timing analysis
3. probabilistic Calculus for probabilistic schedulability analysis
Research Engineer ONERA Toulouse
1. Modeling and development of Embedded Systems
2. Deterministic and probabilistic techniques
3. Measurements-based timing analysis
Performance analysis of networked embedded systems
10/2003 - 12/2006
01/2007 - 12/2010
01/2011 - 05/2012
06/2012 - Today
06/2003
12/2010
2012
Master thesis Polytechnic University of Marche
“Analysis and development of techniques for fingerprint matching”
Prof. Guido Tascini
Ph.D. thesis Scuola Superiore Sant'Anna
“Adaptive resource reservation”
Prof. Giorgio Buttazzo
Qualification à Maitre du Conference
Informatique 27

Figure 1.1: Time-line of the path from the Master thesis graduation to today.

Figure 1.1 zooms in on the time-line of professional and academic paths since the Master thesis. The professional experience at MTS group is included because it is about the development, validation, and certification of embedded systems, as the rest of the academic activities. In the time-line are also listed the main research that are related to embedded systems and real-time development.

1.2 Supervising and teaching activities

Since 2009, 3 PhD students, 6 Master students, and 1 Postdoc have been supervised. Here the details of the supervision activity made.

The PhD students supervised are:

1. Tomasz Kloda at ONERA Toulouse. Thesis subject on ”Conditions dordonnancabilit pour un langage dirig par le temps”; defense in 2016 and thesis director PhD Bruno d’Ausbourg. At the beginning of the thesis, Bruno D’Ausbourg was director and supervisor; at my arrival at the ONERA, I begun a strong collaboration with Tomasz Kloda and Bruno D’Ausbourg such that I become thesis supervisor. Collaboration and supervising were on ”adaptive guarantees for real-time systems” were for developing schedulability analysis for multi-mode real-time systems. In 2016 as member of the Tomasz Kloda’s thesis jury as supervisor. With Thomaz Kloda, the main publications were on deterministic scheduling with time triggered languages and adaptive behaviors guarantees for real-time systems. Thomaz Kloda is actually a Postdoc at the University of Modena and Reggio Emilia Italy.

2. Fabrice Guet at ONERA Toulouse. Thesis subject on ”Etude de lapplication de la thorie des valeurs extremes pour lestimation fiable et robuste du pire temps dexcution probabiliste”; defense in 2017. Nominated supervisor by the ONERA and EDMITT PhD school while thesis director was PhD Jerome Morio

from the ONERA. Publications and collaborations with Fabrice Guet were based on probabilistic timing analysis and statistical approaches to worst-case models to task execution. Fabrice Guet is actually an Embedded System Engineer at SII group.

3. Jasdeep Singh, ONERA Toulouse. Thesis subject on "Probabilistic schedulability analysis of safety critical embedded systems". The thesis is ongoing with a possible defense in 2019. For Jasdeep Singh thesis, director (with derogation) from the EDMITT PhD school, co-supervisors are PhD Guillaume Infantes and PhD Jean-Loup Farges from the ONERA. The collaborations and publications with Jasdeep Singh are on formal methods applied to schedulability analysis of probabilistic real-time systems.

The Master students supervised at the ONERA for internships of at least 4 months are the following.

1. 03/2013 – 08/2013 - Alessandra Melani, Master student at Scuola Superiore Sant'Anna. Stage with subject: "Accuracy in Worst-Case Execution time"; During the stage, there has been a publication [180] and the contributions made to SchedMcore (https://forge.onera.fr/projects/schedmcore).

2. 03/2016 – 07/2016 - Corentin Damman, Master M2AE student at the ISAE – Ingegnerie aerospatiale, aeronautique et astronautique. PIR project with subject: "Architectural performance analysis". During the stage, there has been a publication [157] and the contributions made to SchedMcore as well and FPGA implementation tools.

3. 03/2016 – 07/2016 - Gregory Edison, Master M2AE student at ISAE Ingegnerie aerospatiale, aeronautique et astronautique. PIR project with subject: "Architectural performance analysis". During the stage, there has been a publication [157] and the contributions made to SchedMcore as well and FPGA implementation tools.

4. 03/2017 – 07/2017 - Gregor Vindry, Master M2AE student at the ISAE – Ingegnerie aerospatiale, aeronautique et astronautique. PIR project with the subject on: Subject: "Timing analysis for multi-core architectures". Contribution made to multi-core embedded systems running under Linux.

5. 05/2016 – 09/2016 - Jasdeep Singh, Master International student at ISAE – Ingegnerie aerospatiale, aeronautique et astronautique. Stage with subject: "Schedulability analysis for probabilities and with probabilities". The stage has been about studying the applicability of formal methods to real-time modeling and analysis.

6. 5/2018 – 10/2018 - Julien Durand, Master M2AE student at ISAE – Ingegnerie aerospatiale, aronautique et astronautique. Stage with subject: "Toward predictable many-core real-time systems". The stage studied the predictability of multi-core and many-core platforms while running real-time applications. It has been developed a statistical analysis for average and worst-case task modeling.

Since 12/2017, supervision of Postdoc Phavorin Guillaume with subject: "Probabilistic timing analysis of multi-core platforms". The 18 months Postdoc contract comes from the IRT Saint Exupery in Toulouse and the ANR project CAPHCA. During the Postdoc, there are investigated the predictability and the deterministic levels of multi-core architectures with probabilistic/statistical approaches to timing analysis.

The research activity has been accompanied with the teaching activity:

- 2008 – 2009: Assistant Professor for the "Sistemi Real-Time" course of Prof. Dragoni, Polytechnic University of Marche Ancona. The course was about real-time systems and networks for real-time systems.

- 2012: Assistant Professor for the course "Real-Time Operating Systems" of Prof. Jerome Hugues at the ISAE Toulouse.

- 2012-today: Assistant professor at the ISAE Toulouse. Courses are on:

    1. Embedded systems and control systems (Master first year), 20 hours classes;

2. Real-Time Operating Systems (International Master and Master third year), 10 hours classes;
3. Worst-Case Execution Time (Master third year), 3 hours classes as part of the real-time embedded system course;
4. Object oriented programming - Java theory and practice - (Master second year), 40 hours classes;
5. Multi-core scheduling (Master third year), 6 hours classes as part of the real-time embedded system course;
6. C language (International Master), 12 hours classes.

All the ISAE teaching are in English and French; the teaching in Italy were in Italian.

1.3 Serving within the research community

Participation to PhD and Master thesis juries as a reviewer or examiner:

- In 2013 the Master M2R Mathematique appliquées thesis of Alexandra de Cecco at the ONERA Toulouse;
- In 2013 the PhD of Francesco Prosperi and PhD Christian Nastasi at the Scuola Superiore Sant'Anna Pisa;
- In 2014 the Master research thesis of Bestien Pasdelopup at IRISA Renne;
- In 2016 the PhD thesis of Dario Socci at VERIMAG Grenoble;
- In 2017 the PhD thesis of Ignacio Sanudo Olmedo at the University of Ferrara;

The activity made for the research community is exploited, among others, with review efforts and participating to technical committees.

Regularly reviewer for journals such as ACM Embedded Letters, ACM Transactions on Embedded Computing Systems, Springer Real- Time Systems, Transactions on Computers, Future Generation Computer Systems, as well as JCSC and JSA Elsevier. Reviewer for conference and workshops like ETFA, SIES, ECRTS (sub-reviewer), RTNS, RTSS (sub-reviewer), ARCS, CRTS, WCET, and WMC.

Serving in the technical committee of SIES, ETFA, RTNS, and ARCS conferences, at the Workshops WMC and CRTS at RTSS conference, WCET at ECRTS, and WFCS. Program chair of the Junior Workshop at RNTS 2011, program chair for the IEEE SIES WiP session 2013, program chair for Brief Presentation session (ex. WiP) at RTSS 2017. Serving as session chair at RTNS, ETFA, and SIES.

Since 2014, review of ANR projects; also, registered as external project reviewer for the EU on real-time embedded systems. IEEE fellow from 2009.

Since the beginning of the PhD in 2003, there have been proposed different technical seminars on the research topics approached. Keynote and seminars given are:

- Seminar at ETH Zurich $2008-2009$. Department seminars about real-time calculus achievements and applicability to real-time. In particular, they were about adaptive resource reservation mechanisms validated with real-time calculus;
- Seminar at LORIA Nancy, $2010-2012$. Seminars on projects and research activity results mostly on probabilistic real-time systems;
- Keynote on mixed critical embedded systems in Grenoble 2016. Seminar about probabilistic approaches for real-time task modeling and schedulability conditions;
- Seminar at the University of Missouri Science and Technology in Rolla 2016. Seminar on probabilistic approaches to real-time;

- Seminar at ISAE Toulouse in 2017. Seminar on the mathematics behind probabilistic time models and probabilistic approaches to real-time;
- Seminars ONERA from 2012, two every year. Periodic seminars about the research activity made and the results obtained on topics such as deterministic real-time and probabilistic real-time;
- Seminars for projects like EU FP7 PROARTIS, EU ITEA3 TIMMO-2-USE, DGA IREHDO2, and ANR CAPACITES.

In the last years, together with the collaborations and the students supervised, there have been developed modeling and analysis tools:

- diagXtrm – `https://forge.onera.fr/projects/diagxtrm2`. Measurement-based probabilistic timing analysis tools,
- RTprob – `https://forge.onera.fr/projects/probscheduling`. Analysis tool for the probabilistic schedulability with Markov chain models. The tool proposed models tasks and jobs with continuous time Markov chains and interfaces with the PRISM model checker.

The tools have been realized for projects that have been carried out together with Ph.D students Fabrice Guet and Jasdeep Singh. They are for probabilistic analysis and have been published with papers; they are open source and made available for artifact evaluation and for project use.

1.4 Collaborations and international mobility

Since the PhD thesis, there have been collaborations together with: colleagues at the ONERA Toulouse within the former DTIM department and now DTIS, the IRIT Toulouse with Christine Rochange and Hugues Casse, INRIA Nancy and Paris with Liliana Cucu-Grosjean, the Scuola Superiore Sant'Anna at the RETIS lab with the Giorgio Buttazzo group, the ISAE Touloue, The Barcelone Supercomputing Center with Francisco Cazorla, Jaume Abella and Eduardo Quinones, the VERIMAG in Grenoble Claire Maize, the University of Sciences et Technologies of Lille with Giuseppe Lipari, the School of Engineering of the Polytechnic Institute of Porto and CISTER Porto with Konstantinos Bletsas, the University of North Carolina at Chapell Hill, the Malardalen University, the Ecole Polytechnique of Montreal, the University of Missouri Science and Technology Rolla (now at University of Central Florida in Orlando) with Zhishan Guo, the University of Paris Est with Laurent George, the University of Luxembourg, the University of British Columbia in Vancouver, the Texas State University, the ESIEE Paris, and the IRT Saint Exupery in Toulouse.

There exist also collaborations with companies like AIRBUS, Thales, Honeywell, Real-Time at Work (RT@W), Rapita UK, MBDA, SAFRAN, Continental, Cobham Gaisler, and Kalray.

With some PhD students and research groups there have been strong collaborations on specific scientific topics.

- Francesco Prosperi at the Scuola Superiore Sant'Anna Pisa. Thesis on "Energy management in embedded systems under timing and resource constraints"; defense in 2013, thesis supervisor was Prof. Giorgio Buttazzo. The collaboration with Francesco Propsperi focused on adaptive behaviors for real-time systems and timing guarantees during mode transitions; publications have been made as a result of such collaboration. The collaboration continued as a Postdoc; participation to the thesis jury.
- Dorin Maxim at INRIA Nancy (LORIA). Thesis on "Probabilistic analysis of real-time systems"; defense in 2013, supervisor was PhD Liliana Cucu-Grosjean. Collaborations and publications on "probabilistic real-time systems" i.e., probabilistic schedulability analysis, priority assignment, and distribution re-samplings with safety/pessimism guarantees. They all have been made with Dorin Maxim, Liliana Cucu-Grosjean, and other members of the TRIO team.

- Chao Chen at Polytechnic University of Montreal. Thesis on "Static probabilistic timing analysis for real-time systems in presence of faults"; defense in 2017, supervisor Prof. Giovanni Beltrame. With Chao Chen, the collaboration and publications were about Markov chains applied to static probabilistic timing analysis for random replacement caches and space applications. In particular, it has been developed and applied a Markov chain approach for probabilistic timing analysis. Chao Chen visited the ONERA for three months in 2016.

All the collaborations brought and are bringing to publications, project proposals, and research projects. The scientific production so far composes of 7 journal papers, 2 book chapters on wireless sensor networks and artificial intelligence, and 64 conference papers.

Hereby the exhaustive list of projects (EU project, Italian and French national projects, e.g. ANR projects, and ONERA internal projects) on which there have been contributions and that served as inspiration for the research made or to be made. To all of them, the contribution exploits with deliverables, publications, and technical reports. With the projects, it is also listed the role covered.

1. 2007 – 2009: Scuola Superiore SantAnna and ERICSSON-MARCONI project about data logging and monitoring for real-time networked applications - participant with the Retis Lab in Pisa.

2. 2009 – 2010: Italian project IPERMOB on traffic control and mobility. The project was about the technology transfer from research activity on wireless sensor networks and vision applications - participation with the Retis Lab in Pisa.

3. 2010 – 2012: EU FP7 project PROARTIS "Probabilistically Analyze Real-Time Systems". In the project, the development of real-time embedded systems that can be easily and reliably analyzable with probabilities, `https://www.proartis-project.eu/` - within WP3 coordinated by PhD Liliana Cucu-Grosjean at INRIA Nancy.

4. 2011 – 2012: EU ITEA3 projet TIMMO-2-USE as continuation of the EU TIMMO project. Model temporal, tools, algorithms, languages, methodology, and use cases; `https://itea3.org/project/timmo-2-use.html`. The project applies to automotive and other embedded system applications; contribution made are with modeling and analysis tools for real-time - participant with the TRIO team at INRIA Nancy.

5. 2013: DGAC French project ADCN+. The project was about avionic network performance analysis - participant with the ONERA at the former department DTIM.

6. 2014–2017: BGLE French project CAPACITE. The project was about software and hardware development for embedded real-time reviews on "many-core" architectures; `http://capacites.minalogic.net/en/`. Contributions for timing analysis and certification arguments - coordinator ONERA.

7. 2013 – 2017: ONERA PRF project R2D2. The project relates to creating a deterministic mid-layer for robotic applications. In R2D2, the MAUVE tool-chain developed at the ONERA is applied; `https://forge.onera.fr/projects/mauve` - participant with ONERA former DTIM and DCSD departments.

8. 2015 – 2018: DGAC French project IREHDO2. Performance analysis on AFDX networks. Network calculus and probabilities applied to evaluate and compare performance. - WP1.2.4 participant with the ONERA former DTIM department.

9. Since 2015: ONERA PRF WIRELESS. External collaboration with AIRBUS and ISAE to study applicability of wireless technology within avionic applications - WP2 participant with the multiple ONERA departments, included the new DTIS department.

10. Since 2015: ONERA PRF project MACSIMA. ONERA project to investigate predictability of multi-core platforms. Vision and control algorithms are considered as applications - participant within the ONERA DTIM, now DTIS, department.

11. Since 2018: ANR PRCI project CORTEVA about the development of a correct methodology by construction for support the variability of execution time in real-time systems. Schedulability analysis and sensitivity analysis applied to single- and multi-core platforms - coordinator ONERA, other partners are the University of Lille and RT@W.

12. Since 2018: ANR IRT Saint Exupery project CAPHCA about predictability of systems embedded real-time multi-core `http://www.irt-saintexupery.com/ourprojects/caphca-project/` - within WP2, coordinator ONERA.

13. Since 2018: ONERA PRF project MUSIK. Contributions to big data and statistical analysis of real-time system performance - participant with the ONERA DTIS department.

14. Since 2018: Industrial collaboration project MAMA with MBDA UK and France, and Rapita York. Study of predictability for multi-core architecture with statistical analysis and partitioning techniques - participant within the ONERA DTIS department.

In 2014 and 2015 two H2020 ICT project proposals with French and International partners have been coordinated. In 2014, coordinator of the H2020 ICT-1 Cyber Physical Systems project proposal named TASCOS with 10 EU partners and 4 Millions budget asked. The proposal has not been accepted, but it has been classed first in the waiting list. Noted 14.1/15. In 2015, a similar consortium has been coordinated for second H2020 ICT-4 Customized and Low Power Computing project proposal named TASCOS with 10 EU partners and 4 Millions budget asked. The proposal has not been accepted. Both TASCOS proposal were about the study of trade-off performance/predictability within multi-core platforms.

For all the projects, there has been active participation with technical reports, redacting deliverables, and developing analytical tools for the analysis, the validation, and the simulation of [networked] real-time embedded systems.

Figure 1.2 recaps the professional and academic path listing the main steps from the Master graduation up until today. There are also represented the main projects, the main achievements, and the main collaborations per step.

1.5 Organization of the manuscript

This manuscript is a research oriented document which presents the research project proposed for mixed critical real-time embedded systems. Both, manuscript and research project are also for preparing and defending the "Habilitation Diriger des Recherches" (HDR).

This first chapter, Chapter 1, describes the path made from the Master thesis up until today. It describes the professional and academic experiences, the contributions to the real-time community, the collaborations in the academic and industry communities, and the projects worked on so far.

In Chapter 2 is presented the mixed criticality research project as a journey from past research activity, present contributions on mixed criticality, and future works to mixed criticality open problems. The first session of the chapter presents past research made on real-time during the PhD, during the Postdoc, and during the first years at ONERA. In the past research, the mixed criticality has not been considered, but it is presented in this Chapter in order to outline that I consider the mixed criticality approach as the evolution of previous understanding of embedded systems. The second section introduces the mixed criticality problem with a brief state of the art which lists methods already established. Last session presents the research project around the mixed criticality problem with an introduction to the research activity on mixed criticality since 2015. Also, research perspectives, some inspired from the past and present research, are briefly discussed.

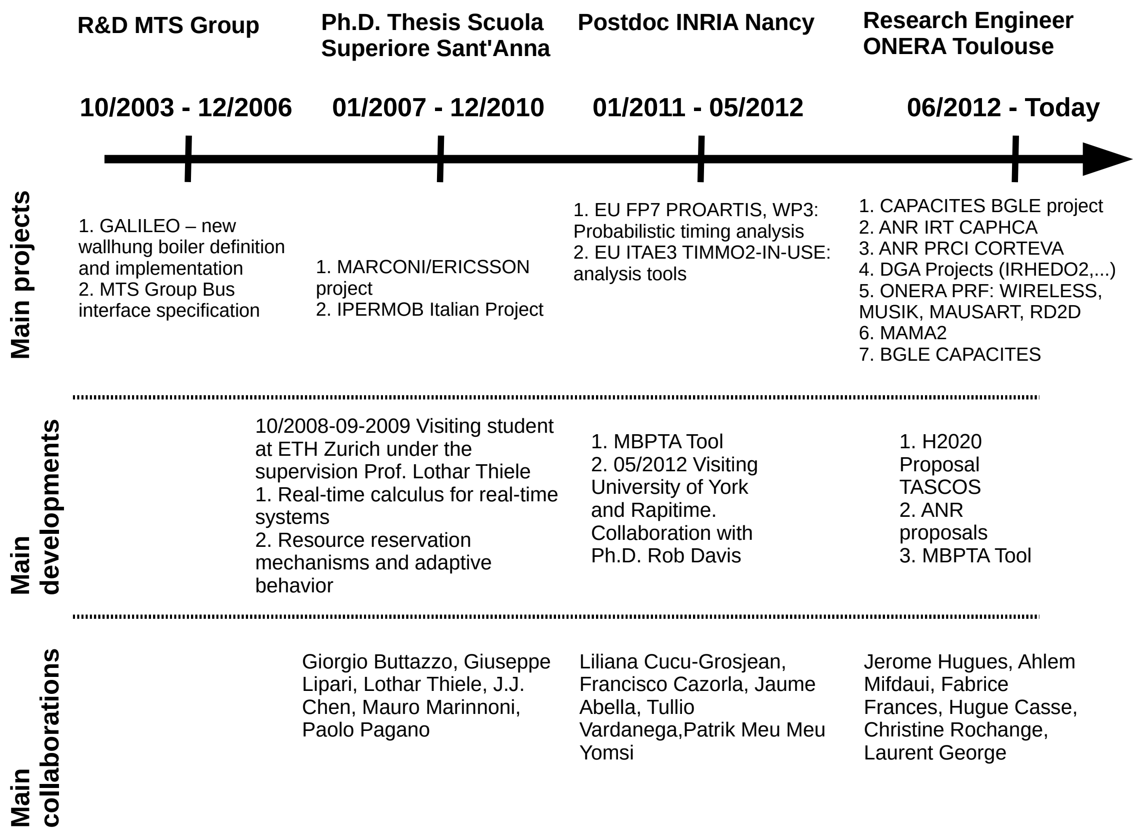

R&D MTS Group
Ph.D. Thesis Scuola Superiore Sant'Anna
Postdoc INRIA Nancy
Research Engineer ONERA Toulouse
10/2003 - 12/2006
01/2007 - 12/2010
01/2011 - 05/2012
06/2012 - Today
Main projects
1. GALILEO – new wallhung boiler definition and implementation
2. MTS Group Bus interface specification
1. MARCONI/ERICSSON project
2. IPERMOB Italian Project
1. EU FP7 PROARTIS, WP3: Probabilistic timing analysis
2. EU ITAE3 TIMMO2-IN-USE: analysis tools
1. CAPACITES BGLE project
2. ANR IRT CAPHCA
3. ANR PRCI CORTEVA
4. DGA Projects (IRHEDO2,...)
5. ONERA PRF: WIRELESS, MUSIK, MAUSART, RD2D
6. MAMA2
7. BGLE CAPACITES
Main developments
10/2008-09-2009 Visiting student at ETH Zurich under the supervision Prof. Lothar Thiele
1. Real-time calculus for real-time systems
2. Resource reservation mechanisms and adaptive behavior
1. MBPTA Tool
2. 05/2012 Visiting University of York and Rapitime. Collaboration with Ph.D. Rob Davis
1. H2020 Proposal TASCOS
2. ANR proposals
3. MBPTA Tool
Main collaborations
Giorgio Buttazzo, Giuseppe Lipari, Lothar Thiele, J.J. Chen, Mauro Marinnoni, Paolo Pagano
Liliana Cucu-Grosjean, Francisco Cazorla, Jaume Abella, Tullio Vardanega,Patrik Meu Meu Yomsi
Jerome Hugues, Ahlem Mifdaui, Fabrice Frances, Hugue Casse, Christine Rochange, Laurent George

Figure 1.2: Time-line of the research steps since the Master thesis graduation.

In Chapter 3 are presented some research results made before electing mixed criticality real-time embedded systems as research project i.e., *the past*. They are works made during the PhD thesis, during the Postdoc, and also during the first period at the ONERA. Those results define the background for timing analysis and schedulability analysis. Among them, the deterministic approaches to task schedulability analysis are defined during the PhD; there is the formalization of probabilistic approaches to task timing analysis made during the Postdoc; there are the probabilistic approaches to task schedulability analysis elaborated during the first years at the ONERA. The past research does not relate to mixed criticality, but it is applied to solve mixed criticality problems. All the works made have the twofold motivation from professional as well as academic experience made on embedded systems.

In Chapter 4 are detailed some works already published on mixed criticality i.e., *the present*; they are the beginning of exploration of the research problem envisioned. Those works apply the background presented in the previous chapter, and allow to address the mixed criticality problem letting it to evolve over its actual state of the art. With respect to timing analysis, the research project considers probabilistic approaches to define more flexible and less pessimistic task models for mixed criticality. With respect to schedulability analysis, the project makes use of both deterministic and probabilistic approaches in order to reduce the pessimism of already existing solutions, to cope with more realistic behavior of mixed critical systems/tasks, and to define more efficient resource usage with multi- and many-core platforms. Such solutions are motivated from both practical and theoretical needs from industry and research projects that ONERA can access.

Chapter 5 concludes the manuscript with remarks as lessons learned from the works already done on mixed criticality, including the potential and the limitations of the solutions proposed. Future work will be proposed to overcome the limitations of already existing solutions and push forward the development of mixed criticality real-time embedded systems for multi- and many-core platforms. Also, are presented future perspectives in order to continue approaching the mixed criticality problem.

Chapter 2

From safety critical real-time to mixed criticality real-time

In this chapter, the mixed criticality problem for real-time embedded systems is introduced as a journey from past "understanding" of real-time systems which was not considering mixed criticality i.e., safety critical systems with only one criticality level, to present and future research which consider mixed criticality. Next chapters detail such journey with past contributions, present contributions, and research perspectives.

The chapter begins introducing the research topics approached since the Master thesis. This section defines the past research which did not considered mixed criticality. The contributions come from the PhD thesis on "adaptive real-time systems", and works on classical real-time modeling as well as analysis. Also, they are from the Postdoc, in which probabilistic approaches have been developed and applied to modeling and analysis of real-time embedded systems. Lastly, in this part there is the initial research at the ONERA that have considered real-time modeling and analysis for multi-core and many-core implementations.

In the middle section, the chapter presents the mixed criticality problem with solutions existing in the literature. The state of the art is in terms of task models and schedulability analyses; there are depicted the academic and industry perspectives to mixed criticality with the differences between academic and industry task models, and academic and industry scheduling approaches. As of today, the two perspectives are still conflicting.

The chapter ends detailing the research project about mixed criticality for real-time embedded systems. In the concluding section, present research and research perspectives are introduced. The project is conceived in order to approach both academic and industry perspectives to mixed criticality. *Timing analysis* and *schedulability analysis* are investigated within the research project; *multi-core* and *many-core* systems are tackled with since the represent the present and the future platforms for real-time embedded systems.

With the mixed criticality research project there are applied *probabilistic representations*. Also, it is discussed the need for less pessimistic *deterministic scheduling algorithms* which apply more realistic hypothesis on the behavior of the tasks. Finally, the *probabilistic scheduling* and its potential for mixed criticality are presented: probabilities will actively take part into scheduling decisions.

Mixed criticality for real-time embedded systems is considered as one of the key elements for developing today's and future real-time embedded systems.

2.1 Past contributions on real-time

In this section there are introduced the works made since the master thesis graduation in 2003. Since then, research interests have always included real-time topics, and among them, real-time operating systems, schedulability analysis and scheduling algorithms, energy aware scheduling, dynamic and adaptive resource reservation/allocation policies, probabilistic real-time modeling and analysis, real-time controls, and formal methods.

We name the past research as safety critical since no mixed criticality problems have been approached in that period: only application with one criticality level i.e., safety critical, have been considered. Although not related to mixed criticality, the past research is presented in this chapter in order to build the link between past approaches to real-time and a new vision of real-time based on mixed criticality. All the past experience is bringing toward the mixed criticality problem, and is providing me tools and knowledge with which effectively approach mixed criticality.

The past research focused on meeting theoretical and industrial needs related to modeling and analysis of real-time embedded systems. This comes from the professional and academic experiences. The underlying theme such past works is to guarantee the performance of real-time embedded systems, critical and non-critical, that run on single-core, multi-core and many-core platforms. With that, five main phases can be identified in order to characterize the past research activity. The phases refer respectively to the professional experience, the PhD. period, the Postdoc period, and the research engineer period at the ONERA.

- **Phase 1:** When the initiation to embedded system took place. The Master thesis, written in $2002 - 2003$ under the supervision of Prof. Tascini at the Computer Science department of Polytechnic University of Marche Italy, has supported research activity for *fingerprint recognition with artificial intelligence approaches*. The solutions proposed applied neural networks and reactive agents: machine learning has been proposed for fingerprint recognition devices, and the algorithms have been implemented into embedded systems [181]. The Master degree in electronic engineering with specialization in computer science has allowed to make the first steps with embedded systems.

- **Phase 2:** During the professional experience in the private sector $(2003 - 2006)$, there has been research and development activities with safety critical embedded systems. Boilers and wall-hung boilers systems have been designed and developed, functional and non-functional requirements have been studied, and real product challenges have been approached. There have been produced certification arguments for single-core platforms, and for the software applications of such products. The collaborations with TUV Netherlands engineers has results into product certifications. The participation to the GALILEO industrial project for defining new boiling systems; in there, the definition of requirements and the implementation of safety critical embedded systems. Participation to industrial projects for introducing RTOSs and RTOS principles with single-core boards on the wall hung boilers. Also, participation to projects for defining and implementing communication protocols for distributed embedded systems. There have been collaborations with ELCO and RINNAI engineers in defining embedded system requirements and implementing time constrained embedded systems.

- **Phase 3:** After the professional experience, it has been chosen in 2007 the PhD program on Innovative Technologies and focused on Embedded Systems from the Scuola Superiore Sant'Anna of Pisa. During the PhD $(2007-2010)$, real-time theory has been applied to embedded systems; there has been the study of the scheduling theory applied to single-core implementations for real-time embedded systems. Component-based design paradigms as well as resource reservation mechanisms and policies have been developed and guaranteed [155, 186, 187, 195, 196, 199, 205, 206]. Deterministic approaches have been applied, and sensitivity analysis has been proposed for adaptive real-time as well as for distributed embedded systems. As modeling and analysis tools applied to investigate real-time embedded systems, there are the analytical approaches like the processor demand criterion, the response time analysis, and the real-time calculus [167, 168, 169, 170, 171]. The real-time calculus has been studied with with the one year visit at the ETH Zurich. Initially, it was for modeling embedded systems and defined component-based feasibility theory for real-time embedded systems. During the visit, the real-time calculus become also a tool for modeling and analyzing adaptive and dynamic systems.

- **Phase 4:** During the Postdoc $(2011 - 2012)$, there has been developed probabilistic approaches for timing analysis (worst-case execution time modeling) and schedulability analysis. The Postdoc has been financed by the EU project PROARTIS and contributions have been made on the timing analysis work package WP3. Measurement-based probabilistic timing analysis approaches have been formalized in order to derive reliable and safe probabilistic estimates of task worst case execution time [153, 156, 177]; measurement-based probabilistic timing analysis is the intended contribution for PROARTIS project. During the postdoc, there has been developed research on static probabilistic timing analysis for random replacement cache memories [158]. Moreover, the probabilistic timing models have been applied into probabilistic schedulability analysis [172, 173, 178, 179, 189, 197]. Those works, have been inspired by the real-time calculus or other analytical approaches to schedulability analysis, and have been designed, validated, and applied to both academic and industrial case studies. The whole contribution of probabilistic approaches to real-time systems is the result of collaborations within the PROARTIS project, with the support of PhD Liliana Cucu-Grosjean and PhD Rob Davis.

- **Phase 5:** During the ONERA fellowship (2012 – today), there has been approached the study of real-time embedded systems, mostly implemented with multi-core and many-core platforms. Both deterministic and probabilistic approaches have been applied to adaptivity problems and mixed criticality under the different conditions that system can experience [151, 160, 161, 165, 166, 174, 175, 190, 193]. Deterministic and probabilistic guarantees have been proposed for those topics. The works on measurement-based probabilistic timing analysis have been extended to multi-core, many-core, and GPU architectures [148, 149, 159, 162, 163, 164, 180, 182, 192, 193]. Also, sensitivity analysis and formal methods applied to scheduling guarantees have been developed. The schedulability analysis has been approached with both tools for less complex approaches and a correct construction of the systems [152, 191, 194, 201, 202, 203]. Formal methods have been applied also to the static probabilistic timing analysis in order to improve previous results on random replacement caches [154]. Distributed embedded systems and embedded networks have been studied with approaches similar to the real-time calculus for performance analysis [150, 183]. Problems with performance guarantees and performance comparisons have been considered with both deterministic and probabilistic approaches for real-time [distributed] embedded systems. The whole contribution of this phase is the result of collaborations within the ONERA, French and International collaborations with both academic and industry, and from the supervision of PhD and Master students.

Some minor contributions are not included into the list above presented; the complete list of works made can be found in `https://www.onera.fr/staff/luca-santinelli`.

The whole work made relates to developing tools for *modeling* and *analysis* of real-time embedded systems. What has been changed across the phases are the tools applied and the perspective adopted to tackle with predictability guarantees; all the works had the twofold motivation from professional as well as academic experience made on embedded systems.

Phase 1 and Phase 2 have been important for understanding the requirements of embedded systems. In particular, Phase 2 gave the industrial perspective with more practical motivations for developing embedded systems. No notion of real-time has been maturated during those, since the experience was more oriented to performance – Phase 1, and efficient resource usage as well as safety criticality by HW and SW construction – Phase 2.

The PhD (Phase 3) served the purpose was for introducing real-time and understanding its benefit for embedded systems. It is the first structured research experience in which the deterministic schedulability analysis has been studied. Single-core and distributed real-time applications have been considered. Also, the problem of adaptivity for real-time embedded systems has been approached and solutions for schedulability analysis have been proposed.

The second research experience, Phase 4, has been for investigating alternative timing models for real-time systems i.e., the probabilistic timing models for tasks parameters. During the Postdoc, it has been developed a measurement-based approach to timing analysis. Probabilistic models have been developed with single-core platforms, but measurement-based approaches will mostly benefit multi-core platforms. Also, the probabilistic models have been applied into specifically developed probabilistic schedulability analysis. The coupling between probabilistic models and probabilistic schedulability analysis allows for more flexibility and more efficient resource usage. The motivation for investigating probabilistic approaches, is to have accurate models for describing system natural variability, a to reduce pessimism with respect to the classical deterministic analysis.

The development of probabilistic models and probabilistic schedulability analyses have continued with the third research experience, Phase 5. In it, deterministic/probabilistic sensitivity analysis and formal methods have been added to the set of tools developed for investigating real-time embedded systems. Since the objective was, and still is, guaranteeing predictability with more efficient resource usage, reduced complexity, and reduced pessimism, both sensitivity and formal methods allow to explore those.

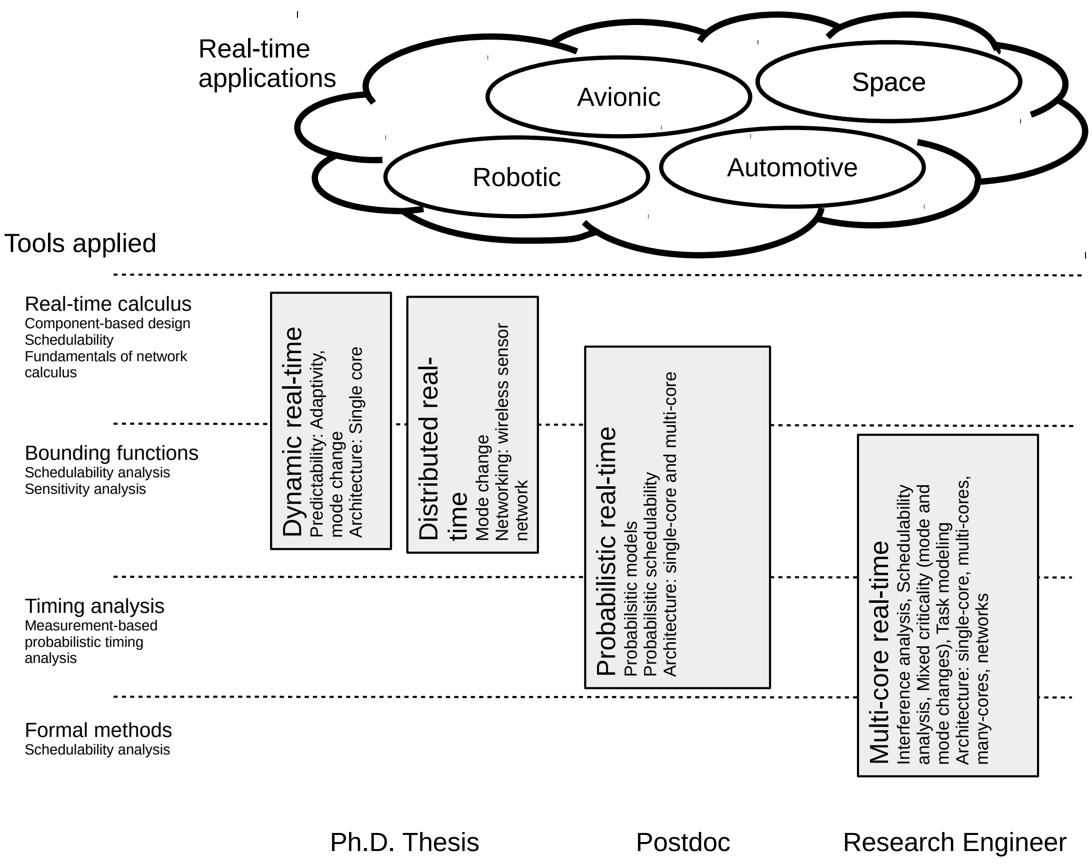

Real-time applications
Avionic
Space
Robotic
Automotive
Tools applied
Real-time calculus
Component-based design
Schedulability
Fundamentals of network calculus
Bounding functions
Schedulability analysis
Sensitivity analysis
Timing analysis
Measurement-based probabilistic timing analysis
Formal methods
Schedulability analysis
Dynamic real-time
Predictability: Adaptivity, mode change
Architecture: Single core
Distributed real-time
Mode change
Networking: wireless sensor network
Probabilistic real-time
Probabilsitic models
Probabilsitic schedulability
Architecture: single-core and multi-core
Multi-core real-time
Interference analysis, Schedulability analysis, Mixed criticality (mode and mode changes), Task modeling
Architecture: single-core, multi-cores, many-cores, networks
Ph.D. Thesis
Postdoc
Research Engineer

Figure 2.1: Research topics (in the middle), theoretical tools (on the left) and applications (at the top) approached during the PhD Thesis, the Postdoc, and as Research Engineer.

The research topics since the PhD are represented in Figure 2.1. There are represented also the theoretical tools applied: real-time calculus, bounding functions, timing analysis approaches, and formal methods for schedulability analysis. In particular, are represented the topics approached during the PhD i.e., dynamic and distributed real-time approached with deterministic techniques; those at the Postdoc i.e., probabilistic real-time for modeling and analysis As researcher at the ONERA, the focus is on multi-core real-time approached with both deterministic and probabilistic methods. Avionic, space, robotic, and automotive applications have be considered all along the path in both publications and research projects.

All the references in this section are my publications since the PhD; they are also not related to mixed criticality but they represent the background to approach mixed criticality research.

2.2 The mixed criticality problem

In the following an introduction to mixed criticality with its state of the art. None of my contributions are cited in this section.

Nowadays real-time systems are mostly implemented with multi-core and many-core commercial-off-the-shelf platforms. Cache memories, branch predictors, communication buses/networks and other features present in such implementations allow increasing average performance.

With multi-core and many-core implementations, it comes the opportunity to combine different applications on the same platform which may have different criticality levels, e.g. safety critical, mission-critical, non-critical. The trend of combining different criticality levels within the same system is driven by the demand for a more efficient platform resource usage.

All of the above contribute making predictability much harder to guarantee.

Criticality is a designation of the level of assurance against failure needed for a system component. A Mixed Criticality (MC) system is one that has two or more distinct levels, e.g. safety critical, mission critical and low-critical. It can have two or more distinct levels, perhaps up to five levels may be identified, see for example the IEC 61508 [81], DO-178B [59] and DO-178C [60], DO-254 and ISO 26262 [82] standards.

The fundamental research question underlying the mixed criticality problem is how to reconcile the conflicting requirements of 'partitioning' for safety assurance and 'sharing' for efficient resource usage. This question gives rise to theoretical problems in modeling and verification, and systems problems relating to the design and implementation of the necessary hardware and software runtime controls [35].

The mixed criticality view

For mixed criticality systems there are distinct issues to be considered. Among them, the *survivability* in terms of graceful degradation, the *static verification*, and the *schedulability analysis*.

In the form of fault tolerance, survivability allows for graceful degradation to occur. Graceful degradation abstractly describes the behavior in which in the event that all components cannot be serviced satisfactorily; the goal is to ensure that lower-criticality components are denied their requested levels of service before higher-criticality components are [35].

Static verification is closely related to the certification problem of safety critical systems. In safety critical systems, typically only a relatively small fraction of the overall system is actually of the highest criticality. In order to certify a system as being correct (or acceptably correct), the designers are guided by the appropriate certification standards, and must make certain assumptions about the runtime worst-case behavior of the system. The certification methods tend to be very conservative, and thus the analysis assumptions result in more pessimism than those that would typically be used if certification was not required [1].

A full review of the certification standards on the subjects of mixed criticality is beyond the scope of this manuscript. For that, the reader is referred to [68, 69]. As a synopsis, few words on standard and mixed criticality. The main points from ISO 26262 [82] for automotive systems are that software safety requirements should include any necessary timing constraints. The software architecture should describe temporal constraints on the software components, including tasks [69], and software testing must include resource usage tests to confirm that the execution time allocated to each task is sufficient [82].

The standards other than the ISO 26262 say very little about timing issues related to mixed criticality. DO-178B and DO-178C (Software Considerations in Airborne Systems and Equipment Certification) are guideline dealing with the safety of safety critical software used in certain airborne systems. Although technically guidelines, they are a de-facto standard for developing avionics software systems. The software level, also known as the Development Assurance Level (DAL) is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. The failure conditions are categorized by their effects on the aircraft, crew, and passengers. DO-178B alone is not intended to guarantee software safety aspects. Safety attributes in the design and implemented as functionality must receive additional mandatory system safety tasks to drive and show objective evidence of meeting explicit safety requirements.

In automotive, the risk classification defined by the ISO 26262 - functional safety for road vehicles standard [82] - is from the Automotive Safety Integrity Level (ASIL). This is an adaptation of the safety integrity

[1] More details on mode degradation and possible notions of mixed criticality can be taken from the contributions of PhD Robert Davis at University of York, Prof. Alan Burns at University of York, Prof. Sanjoy Baruah at Washington University St. Louis, and Prof. Zhishan Guo at University of Central Florida.

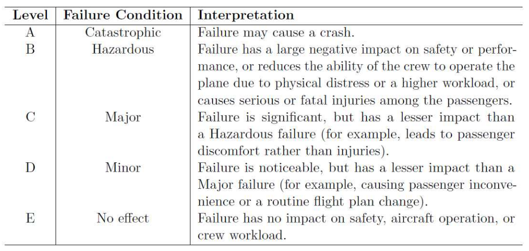
level used in the standard IEC 61508 [1, 81]. This classification helps defining the safety requirements necessary to be in line with the ISO 26262 standard. The ASIL is established by performing a risk analysis of a potential hazard by looking at the *severity*, *exposure* and *controllability* of the vehicle operating scenario.

Figure 2.2 and Figure 2.3 give some details about criticality levels according to different standards. The levels are ordered from less constraining to most constraining. The determination of criticality comes from DAL and SIL notions that are assigned to system level functionalities and to all the software elements that contribute to them [55].

For static verification and mixed criticality, it has to be shown that in the worst-case scenario i.e., all high-criticality functions simultaneously require their maximum pessimistic resource quotas, all of these functions will meet their timing requirements. For graceful degradation and mixed criticality, it has to be measured responses to varying levels of overrun: to not disproportionate the response to a small effect. Indeed for minor overruns it is possible to expect full system functionality to continue to be delivered (a requirement that is termed fail operational or fail passive).

Mixed criticality and mode changes

MC systems are typically defined to execute in a number of criticality modes. The schedulability analysis for the MC problem focuses on providing multiple assurance levels to the possible execution conditions or modes. Today's MC schedulers tend to efficiently schedule functionalities of different criticality so that the less critical tasks may execute in the resource gaps left from the execution of high criticality tasks under normal circumstances. Instead, they may be dropped in a occasional situation where tasks of higher importance level execute beyond their estimated common case running time.



Level	Failure Condition	Interpretation
A	Catastrophic	Failure may cause a crash.
B	Hazardous	Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the plane due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers.
C	Major	Failure is significant, but has a lesser impact than a Hazardous failure (for example, leads to passenger discomfort rather than injuries).
D	Minor	Failure is noticeable, but has a lesser impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change).
E	No effect	Failure has no impact on safety, aircraft operation, or crew workload.

Figure 2.2: DAL criticality levels with failure conditions and their interpretations [59, 60].

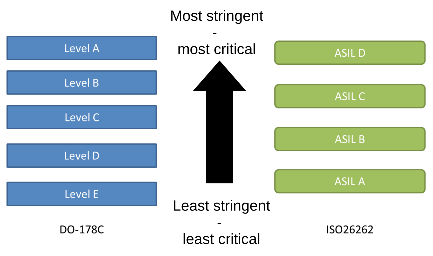


Figure 2.3: ASIL and DAL comparisons [60, 82] with criticality levels ordered from most stringent to the least.

Complex real-time systems are dynamic, which consists in having several behaviors and is described by a set of functionalities that are carried out by different parameter settings. Those systems are characterized by different operational modes, designed to achieve different functionalities or to respond to changes in the environment, [114, 117, 124, 130]. Each mode defines functional and not functional characteristics and consists of specific computational demands, resource requirements and resource availability. The overall computational load and the allocated resources may change over time depending on the operational mode selected for the system. For example, adding a new task into the system at runtime may result in a reduction of the computational resources allocated to the other tasks.

A typical example is that of an aircraft control system where it can be distinguished *landing*, *take off* and *normal cruise* modes, each with a different general objective. The current operating mode of the aircraft depends on the particular phase it is executing. A change in the system state, e.g. from start-up to normal, or from normal to energy saving, or from normal to shut-down, may also require re-allocating the computational resources among the tasks composing the application. Changing one or more parameters in the system components at runtime must also be considered a mode change, because it affects the system load and hence modifies the timing behavior of the application and the system itself.

The mode change is required to cope with the changing conditions so that the system can work properly with the new requirements. in order to change the operating mode it is necessary to switch form a set of parameters to another and this circumstance introduces a transient stage where it is not well defined what are the condition and the parameters of the system along that phase. Such an uncertainty has to be investigated to make the system predictable even during the mode changes. A mode change is initiated whenever the system detects a change either in the environment or in the internal state.

The transformation of real-time systems into MC real-time systems demands for timing analysis which is able to cope with multiple criticality levels for tasks, and schedulability analysis which can guarantee the different possible execution modes.

2.2.1 Mixed critical task models

Vestal's work [140] launched the mixed criticality scheduling theory. In there, a criticality level is assigned to each task derived from the system function that the task is implementing or contributing to. It is the Worst-Case Execution Time (WCET) parameter of task that depends on the criticality level that has to be assured for each task.

In its most general representation, the task model of a period task τ_i is:

$$\tau_i = ([C_{1,A}, C_{1,B}, C_{1,C}, C_{1,E}], T_i, D_i, \chi_i). \tag{2.1}$$

T_i and D_i are respectively the task period and deadline. With X $= \{A, B, C, E\}$ an ordered set of design assurance levels, where A is the highest and E is the lowest. χ_i specifies the assurance level for τ_i. $C_i = \{c_{i,\chi_i}\}$ with $\chi_i \in$ X, specifies a set of execution times for τ_i, and χ_i is the degree of assurance that the execution time of τ_i will not exceed C_{i,χ_i}. It is assumed that $D_i \leq T_i$ and $C_{i,A} \geq C_{i,B} \geq C_{i,C} \geq C_{i,E}$ for all i. χ is the assigned task criticality level [140]. To note that in Equation (2.1), and in the rest of the manuscript, classical criticality level "D" has been replaced with "E" to resolve the ambiguity with task deadline D_i.

A piece of code could have a higher WCET if it is defined to be safety critical with higher level of assurance required ($C_{i,A}$ larger than the others). Instead, if the task is just considered to be mission critical its WCET requirement would be smaller, as the task demands less in terms of resource request and guarantees ($C_{i,B}$ or less). Finally, if the task is considered to be non-critical, it demands even less resource request and guarantees. Its WCET requirements would be the smallest ($C_{i,E}$) [54, 140], [165].

Restricting the modeling to two criticality levels, it is possible to name the high criticality level as HI and the low criticality LO. In the following there is detailed the interpretation for $C(\text{HI})$ and $C(\text{LO})$.

- $C_i(\text{HI})$ (equivalently $C_{i,A}$) that models the HI-criticality behavior of the system, thus the worst possible conditions τ_i can suffer. $C_i(\text{HI})$ would be derived from the most safe timing analysis in order to guarantee the worst model and the most assurance to the execution behavior, [165]; $C_i(\text{HI})$ has to be the task model with maximum confidence of being the worst-case, [140]. This would be the case of safety critical applications that have to account for the worst-case behaviors that can possibly happen. The 'best' modeling of task parameters has to assure the coverage of any of the execution conditions, including the worst-cases [55, 140].

- $C_i(\text{LO})$ (equivalently $C_{i,E}$) that models the LO-criticality conditions for τ_i; $C_i(\text{LO})$ is for mission critical or less-critical conditions; it does not assure against faults, at least it does not against all of them; $C_i(\text{LO})$ could come from less safe timing analysis and encode a certain confidence to be the worst-case model under specific execution conditions [140]. This would be the case of mission critical or low critical applications rely on less constrained/demanding models and the guarantees on them are not as strict as those for safety critical applications [55, 140].

By MC constraints, it has to be $C_i(\text{HI}) \geq C_i(\text{LO})$, [140].

In a more common two-critcality-level model, the MC paradigm it is possible to distinguish between HI-criticality tasks from LO-criticality tasks such that: HI-*criticality tasks* can have two execution modes, HI-criticality mode and LO-criticality mode, represented by $[C_i(\text{HI}), C_i(\text{LO})]$. The HI-criticality task τ_i is represented with multiple parameters:

$$\tau_i \overset{def}{=} ([C_i(\text{LO}), C_i(\text{HI})], T_i, D_i, \chi_i), \tag{2.2}$$

where T_i is the period and D_i is the deadline of the task; $\chi_i \in \{\text{LO}, \text{HI}\}$ is the task's criticality level; and $[c_i(\text{LO}), c_i(\text{HI})]$ is the tuple of WCETs, where $c_i(\text{LO}) \leq c_i(\text{HI})$. The LO-criticality task τ_j is represented with parameters:

$$\tau_j \overset{def}{=} (C_j(\text{LO}), T_j, D_j), \tag{2.3}$$

where T_j is the period and D_j is the deadline of the task and $C_j(\text{LO})$ characterizes the LO-criticality mode worst-case execution time. For LO-criticality tasks only the LO-criticality behavior is possible, as they will be discarded upon a mode switch into HI-criticality level.

The system mode will be switched from LO to HI if any HI-criticality task exhausted its LO-WCET and did not signal finishing. Form the Vestal task model it is notable the increased flexibility to represent task behaviors: with multiple WCETs per task could represent more task behavior, thus reduce pessimism. Current discussions on enhancing the classical (Vestal) MC model could make interesting future contribution to mixed criticality [35, 75].

The MC modeling resolves into a timing analysis problem. Timing analysis seeks upper bounds to the task execution time (with WCET estimates), and so the predictability required by real-time systems can be granted. Classically, the bounds are deterministic WCET which are single values able to upper bound the time needed to finish execution. Refer to [143] for definitions and a survey of approaches for WCET estimates. In order to be safe, WCETs have to account for any case/execution condition possible, including the highly improbable pathological cases such as faults. Deterministic WCETs could be very pessimistic with respect to task actual execution times, and could lead to resource under-utilization. Nonetheless, it is safe.

Traditional rigorous WCET analysis may lead to not affordable pessimism (for safety), and the occurrence of such WCET is extremely unlikely, unless under highly pathological circumstances. The gap between the actual running time and the WCET may be significantly large, which makes a strong argument for MC task modeling according to Equation (2.2).

Much prior research on mixed criticality scheduling has focused upon the phenomenon that different tools for determining WCET bounds may be more or less conservative than one another, which results in multiple WCET estimations for each individual task or piece of code. Typically in the two-criticality-level case, each task is designated as being of either higher (HI) or lower (LO) criticality, and two WCETs are specified for each HI-criticality task: a LO-WCET determined by a less pessimistic tool, and a larger HI-WCET determined by a more conservative one, which is sometimes larger than the LO-WCET by several orders of magnitude.
Safety critical systems are failure prone as any other system, and today's system certification approaches recognize this with permitted system failure probabilities. The underlying idea is to certify considering more realistic system models which account for any possible behavior, included faulty conditions, and the probability of these behaviors occurring. The gap that still exists is between such enhanced models and the current conservative deterministic analyses.
Task models have to embed fault manifestations and fault effects in order to provide upper bounds to every possible condition the real-time system can experience at runtime. In the MC framework, and with the highest abstraction model, faults can be seen as task passing $C(\text{LO})$ thresholds. Timing analysis approaches have to take fault effects into account. System faults have a non negligible impact on worst-case models; although as pathological and improbable cases, they have to be considered with timing analysis.

2.2.2 Mixed criticality scheduling

The MC scheduling extends with multiple conditions the classical scheduling conditions. Restraining to two-criticality level, MC schedulability analysis objective is to determine a runtime scheduling strategy which ensures that: (i) all jobs of all tasks complete by their deadlines if each job completes upon executing for no more than its LO-WCET; and (ii) all jobs of tasks designated as being of HI criticality continue to complete by their deadlines (although the LO-criticality jobs may not) if any job requires execution for more than its LO-WCET (but no larger than its HI-WCET) to complete.

MC systems are typically defined to execute in a number of criticality modes. All of them have to be guaranteed at runtime. According to Vestal's definition [140], mode switch can be defined as follows: if any task attempts to execute for a longer time or more frequently like in case of faults, then a mode change occurs imposing high-criticality behavior to the tasks and the system [21, 46]. Under classical MC model, all low-critical tasks could be dropped from the system upon a mode switch, which may be a result of one single high-criticality task overrunning for 1 ms, or a 1 ms speed drop of one of the many processors. Obviously huge pessimism is involved under such modeling, even with the recent developments in providing multiple assurance levels to the possible execution conditions [22, 24, 25, 57].

In the real-time systems community, some existing work in MC scheduling consider the concept of graceful degradation and proposes effective scheduling techniques, such as fluid-based scheduling [17], utilization-based Earliest Deadline First with Virtual Deadlines (EDF-VD) [103], putting restriction on the asymptotic rate-based correctness notation [20, 125], and involving mixed criticality weakly hard constraints [67]. [110] applies probability thresholds into MC schedulability conditions; different schedulability conditions are defined from LO-criticality to HI-criticality.

The Own-Criticality-Based-Priority (OCBP) algorithm is proposed to generate a correct priority list for mixed-criticality jobs in polynomial time [23, 27]. This is an efficient approximation algorithm which recursively searches a lowest-priority job by simulating the behavior of all other jobs according to the candidates own criticality. OCBP following the idea of Audsleys priority assignment mechanism [12].

In the traditional real-time scheduling, Earliest Deadline First (EDF) is proved to be an optimal exact algorithm on preemptive single-processor platform. Its philosophy of favoring urgent jobs, in the form of always choosing the job with the earliest deadline to execute, provably maximizes the use of processor capacity in the traditional real-time systems. In order to promote high-criticality tasks in MC systems, these tasks will be given earlier virtual deadlines to reflect their importance over low-criticality tasks. This is how the modified EDF algorithm named EDF-VD [23], where VD stands for virtual deadlines, works for MC systems. EDF-VD algorithm shrinks the high-criticality deadlines proportionally by a certain factor so that the high-criticality tasks will be promoted by the EDF scheduler.

The OCBP algorithm is aimed at discrete one-shot tasks with an arbitrary number of criticality levels. The EDF-VD algorithm is aimed at recurrent tasks with two criticality levels (safety critical and non-critical). The two algorithms differentiate also because they consider respectively jobs and tasks.

Another MC scheduler is the Mixed-Criticality Federated Scheduler (MCFS) [100] which can reduce the pessimism in resource allocation by allowing HI-criticality tasks designating a worst-case overload utilization as well as a common case nominal utilization, and allowing lower-criticality tasks to meet their deadline as long as HI-criticality tasks requires their nominal utilization. MCFS still ensures that HI-criticality tasks will meet their deadlines if they end up requiring their overload utilization.

Under MC, the resource is utilized in the manner such that all tasks are allowed to execute under low-critical modes in a more fairly manner, while priorities will be given to more critical tasks in a dedicated manner upon a mode switch. Such mode based correctness definition is welcomed by the industry, yet providing many research challenges [19].

It is known [8] that MC scheduling under Vestal model is highly intractable, such that polynomial-time optimal solution is impossible unless $P = NP$. As a result, speedup bound is widely used in MC scheduling for measuring how close to optimal is a given schedulability analysis. A schedulability test has speedup factor of s

($s \geq 1$), if any task set that is schedulable by any algorithm on a given platform with processing speed of 1, it will be deemed schedulable by this test upon a platform that is s times as fast.

Of course when deriving MC schedulers and associated schedulability tests, one of the goals is to identify/prove a relative small speedup bound (that is closer to 1). A minimum possible speedup is often presented as the optimal speedup bound of a given MC scheduling problem.

When deriving speedup bounds, in most of the existing works of the community, the proposed algorithm is compared with a clairvoyant optimal scheduler, and adapts the necessary conditions for MC schedulability. This may not be a very fair way of comparison, since the penalty for unawareness of the future is applied into the speedup bounds

For scheduling (dual-criticality) Vestal job set on a single-core platform, it has been shown [27] that OCBP algorithm has an optimal speedup bound of $\frac{\sqrt{(5)}-1}{2}$. However, several algorithms has been identified to strictly dominate OCBP, e.g. Lazy Priority Adjustment [70], LE-EDF [73, 74]. They have the same speedup, yet the latter has better schedulability at all time. Similar results can be observed when it is considered the scheduling of Vestal task as well. It has been shown that $\frac{4}{3}$ is the best speedup that any non-clairvoyant scheduler can achieve. Upon proposing a speedup-optimal single-processor EDF-VD [27], improvements on the schedulability can still be made, e.g. [50, 52]. As for the multiprocessor case, it is proved [18] that both MC-Fluid [123] and MCF [18] achieve the optimal speedup of $\frac{4}{3}$. However, MCF is a simplified version of (and is dominated by) MC-Fluid. Moreover, improvements on schedulability can be further made to MC-Fluid [123].

So far, the academic perspective to the MC problem which focuses on merging tasks within the same core. The objective is to improve resource usage by mixing tasks with different criticality levels. It is left to the schedulability analysis guarantees task timing constraints depending on the criticality level and the available resource. Have a look at [8] for recent observations on the complexity of the MC schedulability problem.

2.2.3 The industry perspective to mixed criticality: partitioning and isolating

The industry perspective to the MC problem focuses more on partitioning and isolating applications by their criticality level, and then guaranteeing separation mechanisms. Safety critical applications would be isolated in time and/or in space from mission critical applications, [55, 85]: both safety critical and mission critical MC models and MC schedulability are guaranteed within the partition with approaches that resembles more to classical schedulability analysis. For industrial MC applications, a MC system is not a system where to sacrifice lower criticality applications. Instead the applications are partitioned by their criticality level and execute in isolated environment to avoid that HI-criticality tasks suffers interference from LO-criticality tasks.

With the industry, the core concept in realizing MC systems is the demonstration of "sufficient independence" and guaranteeing that the mechanisms that should provide sufficient independence actually provide it. The mechanisms are kernels and schedulers that guarantees resource management to provide independence in the functional and time domain, and mechanisms that detects faults and control them, e.g. monitors, scheduling.

A clear example of the industry perspective to the MC problem is the IMA paradigm for avionic real-time embedded systems. The IMA concept proposes an integrated architecture with application software portable across common hardware modules. An IMA architecture imposes multiple requirements on the underlying operating system and consists of a number of computing modules capable of supporting numerous applications of differing criticality levels. It is clear how the industry goals from mixing criticality are more about partitioning applications into modules and easy the certification jobs of each module and module composition.

Figure 2.4 represents an example of distributed IMA avionic embedded systems where modules communicate via the safety critical embedded network like AFDX [1]. Other examples of IMA applications can be find, including comparisons between IMA and the former federated architecture design paradigm.

Partitioning is a concept for spatial and temporal isolation/segregation of functionally independent components in order to prevent interference between two components. Also, partitioning is key in incremental development as pushed by IMA. There exists time partitioning where temporal aspects are involved; there exists

[1]The picture is taken from the AIRBUS presentation at the ARTIST2 Workshop on Integrated Modular Avionics, PARADES, Rome 2007.

space partitioning which focused on memory aspects; in the I/O partitioning, both time and space partitioning for I/O are involved.

The partitioning is achieved through independent segregated environments. The use of a separation kernel or memory management unit to control execution instances. Instead, temporal partitioning does not need to be implemented in time slots [33].

The complexity of the "industrial MC" shifts toward partitioning problems and guaranteeing temporal and spatially each partition. In that, the most significant effort is devoted to develop and certifying real-time operating systems and/or hypervisors which guarantees that. Few research works have approached the MC scheduling with industry perspective. [85] is an example where the solution consist on executing the most critical tasks in isolation and then the execution of the remaining tasks follows.

Moreover, in "industrial MC" it does not exist the notion of criticality mode, mode change, nor LO-criticality WCET overrun. Instead, the task model remains the classical Liu and Layland task model [102], with only one WCET per task, and the criticality classification is applied to tasks according to their level of assurance.

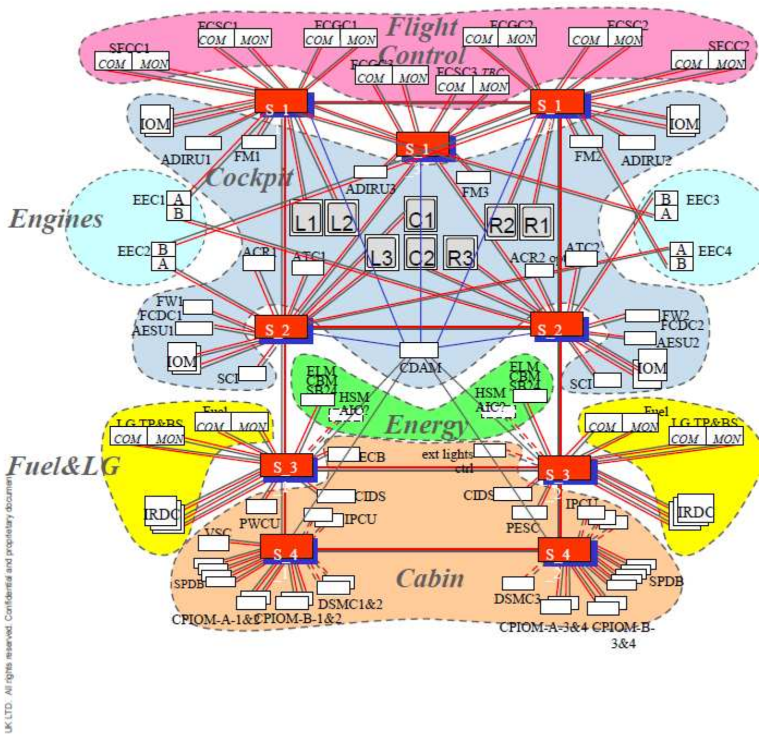


Figure 2.4: An example of IMA for avionic distributed application with different functionalities and criticality levels associated.

2.2.4 Dynamic systems: toward mixed criticality

The problem of schedulability analysis across mode changes, similarly to mode MC, has been addressed in the real-time literature under different assumptions and system models [117, 127, 133, 138]. For instance, Fohler [58] investigated the problem of mode changes in the context of pre-runtime scheduled hard real-time systems, where a table-driven schedule is constructed for each operational mode and an appropriate time must be selected to start a new mode and avoid deadline misses. Crespo et al. [124] presented a survey of mode change protocols for single-processor systems under fixed-priority scheduling and proposed a new protocol along with their own schedulability analysis. Guangming [71] computed the earliest time at which a new task can be safely added to the system under the EDF scheduling, without jeopardizing the feasibility of the task set. The underline idea behind such solutions is to wait for a certain amount time before changing the schedule, identifying a safe time instant where the new mode can be activated without causing deadline misses.

All of these results address the problem of performing mode transitions in applications without violating their schedulability. System that can changes are called *dynamic systems*, and guaranteeing timing constraints during steady states and during mode transitions is a research topic often named *adaptivity*.

Together with the applications, any other element of real-time system may change at runtime. The parameters of servers, schedulers (hence the scheduling policy) and more other can change as a consequence of either external or internal changes, [119]. Furthermore, whereas a server manages an application by supplying the resource it requires, adaptive applications must rely on adaptive servers to meet their changing resource requirements.

Fixed reservations paradigms are not appropriate to achieve the desired performance with applications in which the computational demand is highly variable. To cope with such dynamic systems, Buttazzo et al. [36] proposed an elastic scheduling methodology for adapting the rates of a periodic task set to All of these frameworks are only suitable for soft real-time systems. Abeni and Buttazzo [7] introduced a bandwidth reservation mechanism (the constant bandwidth server) that allows real-time tasks to execute in dynamic environments under a temporal protection mechanism, so that the server never exceeds a predefined bandwidth, independently on the actual requests of the server tasks.

Despite the amount of works done so far, the classical server paradigms and models do not allow adaptations to changing conditions. To the best of our knowledge, however, none of the proposed reservation mechanisms has been analyzed to predict the timing behavior of the served application during a reconfiguration process. Clearly, a safe approach could be to delay the mode change at the next idle time in the system, as done in the FRESCOR framework [41]. However, the delay cold be too long and it is highly unlikely that the idle time occurs at the same time for all applications.

Recently, some mechanisms have been proposed to dynamically change the server models at runtime. For instance, de Olivera et al. [47] addressed the problem of dynamically reconfiguring reservation parameters, offering support for multi-mode and adaptive real-time applications. Valls et al. [139] presented an adaptation protocol based on the definition of a contract model for filtering peaks in resource demands. However, in both frameworks no schedulability guarantee is provided during reconfigurations. The FRESCOR project [79] has proposed mode change protocols for sporadic servers, but they are not as general as the results presented in this paper, which can cope with arbitrary activation patterns.

To stress the fact that I see mode change analysis and adaptivity for real-time systems as the precursor to MC analysis. They both aim at defining safe mode change and guaranteeing determinism during steady states as well as during mode transitions. With MC the notion of mode change is applied to describe different possible behaviors for the task in relationship with the criticality level to be guaranteed during execution; the MC schedulability analysis guarantees task schedulability once the mode change has taken place. The techniques applied in adaptive real-time and the MC problem are slightly different, but both relates to the common need of guaranteeing predictability during possible changing conditions fro real-time systems. This is the reason for which adaptive real-time works are in this state of the art, and are also among the research topics I have approached in the past.

There exist a difference between the notion of mode change with MC and the notion of mode change for adaptivity. In MC, mode changes happens for different reasons, and the scheduling conditions after the mode change have to be guaranteed; *the guarantees are for higher criticality level after the mode change.* In adaptivity, it is the mode transition that is taken into account; *the guarantees here are for making a predictable transition.*

The investigation of dynamic real-time systems consists of proposing schedulability analysis that accounts for modes as steady states modes and mode transitions [5, 6, 7, 36, 41, 112, 117]. Dynamic real-time systems, with their adaptive problem could be considered the precursor of MC; the research serves as inspiration for the MC.

2.3 Mixed criticality as research perspective

The research project chosen to be investigated in the present research activity and mostly in the future, is **mixed criticality for real-time embedded systems**. With it, the intent is to define *accurate mixed critical task models* as well as *efficient mixed criticality scheduling algorithms*.

The research project collects the past research as background, presents some early works to timing analysis and schedulability analysis, and outlines research perspectives. Future contributions from it will continue investigating the mixed criticality problem tackling with timing models and schedulability analysis for improving already existing solutions or proposing new ones.

In my past research, before 2015, and for the professional experience, the mixed criticality problem never appeared. This is because past activity was more focused on understanding and modeling traditional real-time embedded systems. The MC research project is the *evolution* of my perception of real-time systems and of the past research topics. It is *interesting* and it is *challenging* for the real-time community, especially for the development of today's and tomorrow real-time embedded systems.

The MC is becoming one of the key perspectives for studying real-time embedded systems and for guaranteeing their predictability. The manuscript and the research project chosen represents the occasion for formalizing future perspective around the mixed criticality problem, for defining the tools to be applied to approach MC problems, and for proposing research direction to take. In both, industry and academic requirements and motivations for mixed criticality are taken into account.

MC in real-time means different level of guarantees combined in the same system: approaches for studying MC systems have to apply techniques for hard real-time as well as techniques for soft-real-time. MC means also complexity issues for assuring timing constraints from the different requirements needed: MC schedulers have to manage the different requirements guaranteeing first safety critical tasks. Moreover, MC means also efficient resource usage challenges: once guaranteed HI-criticality tasks, less critical tasks can be executed without jeopardizing the rest.

Mixing criticality within the same system is becoming the trend that both academics and industry tackle with when implementing and guaranteeing real-time embedded systems.

Evolution of real-time embedded systems. Multi-core and many-core allow for enhancing performance due to their features, e.g. the large amount of shared resources, the parallelism and the inter-core communication infrastructure. But in order to be effectively applied with real-time systems, their main challenge remains the reconciliation of performance and predictability.
With that amount of resource available, real-time systems begin packing applications with different criticality and different functionalities within the same platform. Thus, it exist the need to model different behavior depending on the runtime conditions, and to guarantee different timing requirements depending on criticality requirements.
The MC problem is the evolution of classical real-time and it very actual due to multi-core and many-core implementations: many open problems exist and need to be taken into account. Among them, and what will be studied with the research project, there are: 1) the *complexity* of scheduling algorithms, 2) the *pessimism* of task models and of schedulability results, and 3) the *trade-offs* between partitioning and resource sharing.
MC demands for enhancing already existing approaches. In the past, the problem of adaptivity and mode change guarantees has been approached; the research project naturally looks at MC as continuation of that. Past contributions on deterministic scheduling such as sensitivity analysis will be applied in order to reduce complexity of already existing MC approaches. Also, previous research has been made with probabilistic models which proved to reduce the pessimism of classical deterministic models. It is necessary a research project that applies such models with MC tasks in order to reduce pessimism and in order to evaluate effects of resource partitioning or complete sharing. Past research has also considered probabilistic schedulability analysis. With the use of probabilistic models, it will be approached both complexity and pessimism problems for MC.

Interests and challenges with MC. The MC problem is interesting since it concentrates the attention of academics and industry from the embedded system community. They offer different perspective to the MC problem, but both concord that it is the actuality and the future of real-time embedded system development.
Multi-core implementation and MC problems, combined together give rise to theoretical problems in modeling and verification, and systems problems relating to the design and implementation of the necessary hardware and software runtime controls. The MC problem is challenging and in large part remains an open problem. Recent attention is dedicated to certification issues and its challenges with multi-core and many-core. All those aspects have to be considered.

Industry and academic perspectives to MC. There still exist the dichotomy between the research community and the industry, in coping with the MC problems. Either the two communities do not speak the same language with respect to MC, or they propose completely different models and algorithms. This represent a problem that keeps the two communities separated with limited communication and sharing. The research community should make an effort in interpreting industry needs with today's and future real-time systems, and proposing solutions that can cope with those.
With multi-core implementation, mixing criticality problem is not just ordering (scheduling) task execution anymore. There are problems which have to be taken into account for timing analysis and schedulability analysis. All of them deserve an important role in developing MC task models and MC scheduling algorithms.

- There is the problem of partitioning tasks and placing them into different cores and guaranteeing their predictability according to different possible criticality levels. In case of partitions, what are the isolation properties from the RTOS or the hypervisor? What are the guarantees that every shared resource is well partitioned among the applications? What is the interference level from other tasks and partitions?

- There is the concurrency problem and the interference that shared resources create with or without partitioning. The interference have to be embedded into efficient task models and only then they can be taken into account by the schedulability analysis. This is even more important without partitioning, with more task competing for shared resources: the interference effects can be important and affect schedulability conditions. Timing analysis has to take into account interference on shared resources and the impact they have on worst-case models; timing models have to be parametrized by the criticality levels and the guarantees that different criticality levels deserves.

- There is the problem of efficient resource utilization among different criticality tasks and the development of less pessimistic or more realistic scheduling algorithms.

MC has to involve timing analysis and schedulability analysis problems. With multi-core and many-core implementation, this brings in plenty of elements to be investigated and to be guaranteed. With respect to timing analysis, the study of interference, the evaluation of resource partitioning, and more, have to be taken into account. Probabilistic models and statistical approaches allow for better task models and reduce the pessimism in MC representations. With respect to schedulability analysis, mode change and mode combinations have to be efficiently included into scheduling decisions. The challenge here is on developing better (more efficient and less complex) scheduling decisions able to account for all the possibilities that can happen at runtime. Probabilistic approaches and sensitivity analysis will help with that.

The research project plan consists on approaching both timing analysis problems and schedulability problems which are related to MC. Both are necessary for effective modeling and analysis of mixed criticality; both contributes to efficient resource usage with multi-core and many-core applications.

MC timing models can be improved, and also newly developed, especially the probabilistic ones. Probabilistic model to task behavior can be applied to the MC problem. The enhanced flexibility of such models will allow to represent tasks with different criticality with less pessimism but still safely. Interference analysis between tasks and on shared resources will allow characterizing interference effects and quantify those with respect to the criticality levels. Statistical analysis will allow for guaranteeing space and time partitioning between applications. Partitioning mechanisms can be analyzed studying the impact that partitions have on MC task models. Moreover, interference, contentions, and their effects have to be studied in order to improve the knowledge of multi-core dynamics, and evaluate the quality of partitioning.

Scheduling algorithms for MC need to be improved, and efficient analyses analyses need to be developed. The extension of existing deterministic approaches based on a flexible mode change and a smarter priority assignment is possible. In addition, new analyzes of probabilistic schedulability are possible by applying the flexibility of the probabilistic model. The new deterministic and/or probabilistic solutions must all guarantee the level of criticality and should improve the use of resources and the predictability of these systems, primary needs for the industry as well.

The research project will exclusively tackle with multi-core and many-core applications. All the contributions will have the twofold motivation from professional as well as academic experience made on embedded systems.

The research project will account for academic problems as well as industry problems to MC. A complete research project cannot neglect that: i) theoretical problems are important to push forward research and contributions to the development of today's and future real-time embedded systems; ii) practical problems are necessary to face needs that realistic MC systems have. The attempt in this project is to develop similar approaches for academic and industry challenges, and also develop adhoc solutions for issues that are peculiar to the specific perspective considered. The professional experience and academic experience made so far make approaching the problem from both theoretical and practical requirements.

The academic perspective is important also for arguing classical research approaches to MC and the assumptions made to obtain them. Among the academic assumptions that will be studied or revised, there are:

- The multiple execution time model proposed i.e., Vestal's, which will be studied and extended with probabilities;
- The criticality level that apply to tasks and to system level functions. Differences and commonalities to both will be investigated;
- The definition of criticality mode change. Classically, it happens when there is violation of timing assumption; it will also be studied other possible definition of mode change which could better cope with realistic conditions;
- Resource usage improvement is obtained leveraging the difference between $C(\textsc{hi})$ and $C(\textsc{lo})$. The research project will study that as well as other solutions for more efficient resource utilization for the MC paradigm.

The industry perspective focuses more on guaranteeing safe resource partitioning and isolation. In there, partitioning mechanisms are tested, interference analysis are proposed, and rigorous guarantees are provided. Also, the attention to certification requirements proper of industry approaches would benefit academic results with rigor and formalism.

Figure 2.5 illustrates how the theoretical tools prepared since the master thesis (in white background), and applied to non-MC topics (in gray background), have and will be applied to MC. Previous contribution, as in Figure 2.1 can be compared with the research perspective on MC. In Figure 2.5 are outlined the challenges to be approached and how the research topics as well as the tools will apply for that. Also, it is listed a brief list of motivations for the use of such into the MC problem which embed research and professional experience: reduced pessimism, flexibility, and trade-offs which will be detailed in the following.

In the following, some MC problems are discussed listing past and present contributions as well as the research perspective for them.

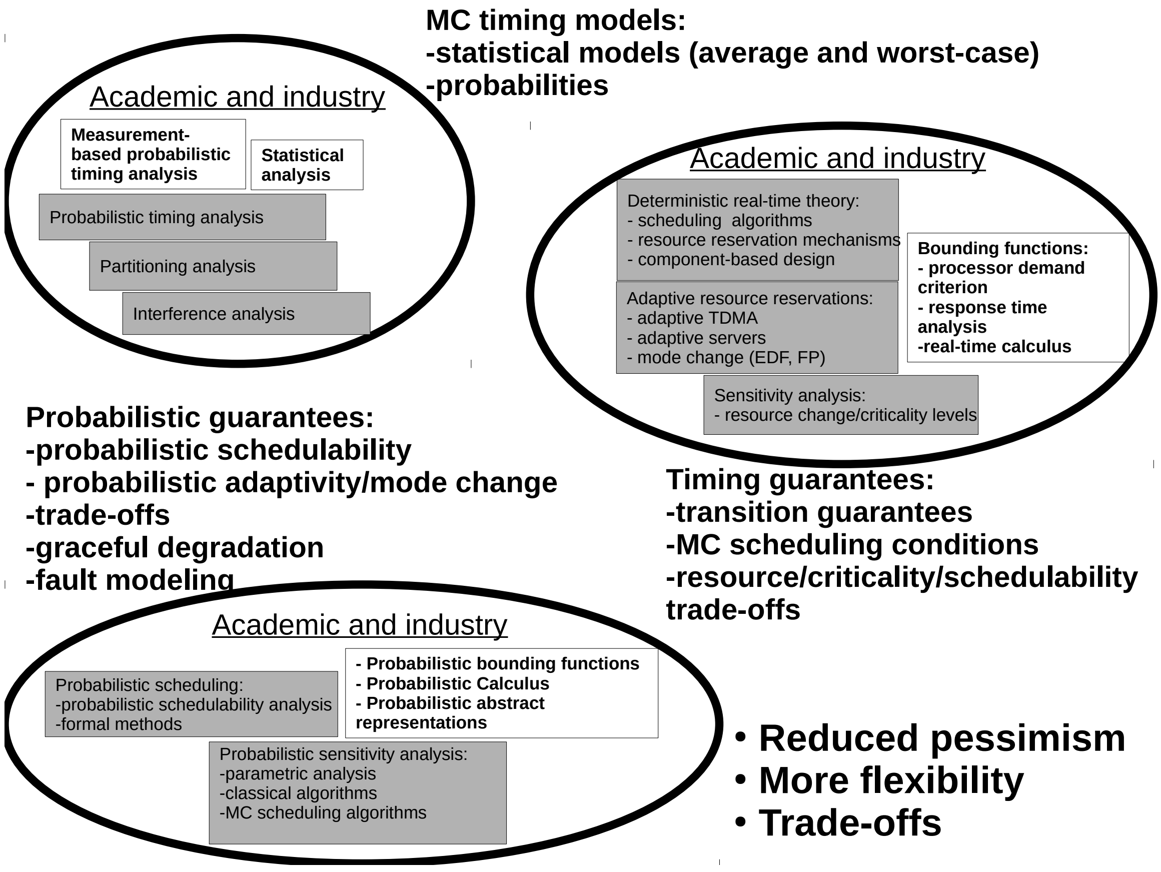

MC timing models:
-statistical models (average and worst-case)
-probabilities
Academic and industry
Measurement-based probabilistic timing analysis
Statistical analysis
Probabilistic timing analysis
Partitioning analysis
Interference analysis
Academic and industry
Deterministic real-time theory:
- scheduling algorithms
- resource reservation mechanisms
- component-based design
Bounding functions:
- processor demand criterion
- response time analysis
-real-time calculus
Adaptive resource reservations:
- adaptive TDMA
- adaptive servers
- mode change (EDF, FP)
Sensitivity analysis:
- resource change/criticality levels
Probabilistic guarantees:
-probabilistic schedulability
- probabilistic adaptivity/mode change
-trade-offs
-graceful degradation
-fault modeling
Timing guarantees:
-transition guarantees
-MC scheduling conditions
-resource/criticality/schedulability trade-offs
Academic and industry
- Probabilistic bounding functions
- Probabilistic Calculus
- Probabilistic abstract representations
Probabilistic scheduling:
-probabilistic schedulability analysis
-formal methods
Probabilistic sensitivity analysis:
-parametric analysis
-classical algorithms
-MC scheduling algorithms
• Reduced pessimism
• More flexibility
• Trade-offs

Figure 2.5: Research topics since 2003 and theoretical tools applied to MC problem: possible achievements and motivations.

2.3.1 Some thoughts about timing analysis for mixed criticality

With regard to timing analysis and mixed critical task models, the research project will focus on probabilistic models.

Recently, probabilistic timing models are emerging as alternative to the classical deterministic ones. For example, what the real-time community is discussing nowadays is the notion of probabilistic Worst-Case Execution Time (pWCET) and the confidence that can be offered to probabilistic bounds for task behavior.

The pWCET model generalizes the WCET with a worst-case distribution that upper bounds any possible task execution behavior. Hence, pWCET representations focus on probability of occurrence of worst-case conditions and abstract them into multiple worst-case values with their correspondent probability of happening [158].

The task execution worst-case representation using random variables is richer and less pessimistic than with classical deterministic WCET. A deterministic WCET can be far from the real behavior since it has to upper bound any possible task behavior, including the extremely rare worst-cases. WCET representations using random variables are emerging as alternative to deterministic WCETs. In that regard, a pWCET models what is behind the WCET bound with possible WCET values each with an associated probability/confidence of being passed.

The task nominal behavior is made of multiple execution times depending on the different phenomena that can apply at runtime: on average, the task behavior resemble more to a random variable than a deterministic variable. The notion of pWCET is a generalization of that, applied as a probabilistic worst-case bound to the task execution behavior.

From pWCET estimates, it is possible extract WCET thresholds, each with an associated probability of being overcome. Multiple thresholds or different distributions, each with a confidence associated depending on the conditions considered, will cope with the MC task modeling; they will aim at providing less pessimistic models able to face the different runtime behavior that MC task exhibit.

Probabilistic models with MC are for improving the quality of the models: reduce pessimism, have more flexible representations with multiple possible values, and maps probabilistic values with criticality levels. Also, statistical approaches and probabilistic models are for representing interference effects, partitioning properties and shared resource usage with MC multi-core and many-core implementations. Furthermore, fault effects will be included into MC probabilistic task models to parametrize criticality levels with different fault models. In Chapter 4 and Chapter 5, are presented respectively few works and future ideas to begin explaining probabilistic timing analysis and its potential for MC.

During the Postdoc period, with the support of the EU project PROARTIS, I made works on probabilistic timing models.

Few have been my contributions on static probabilistic timing analysis which apply to randomized real-time systems [154, 158]. Exact and estimated pWCET are computed from the knowledge of the task instruction sequence (task model) and from the possible interactions with concurrent tasks i.e., preemptions. More have been my contributions on measurement-based probabilistic timing analysis [148, 149, 159, 162, 163, 164, 182, 192]. In those, the pWCET are estimated from task execution time measurement: the pWCET are from nominal models of task executions. In particular, the PhD thesis of Fabrice Guet $2014 - 2017$ have addressed confidence and robustness problems of the measurement-based methodology.

The research project is for applying all those results with probabilistic timing models to the MC problem. *Modes for timing analysis are execution scenario, or subset of execution scenarios, or system system hypotheses which define specific behaviors of the task to be modeled.*

So far, I have not maturated the competence to approach deterministic timing analysis. Instead, in collaboration with the IRIT Toulouse, it will be developed a hybrid timing analysis approach which combines static timing analysis and Measurement-Based Probabilistic Timing Analysis (MBPTA). This will apply to MC since different tools applied for tasks depending to their criticality levels. The idea of hybrid timing analysis is briefly presented in Chapter 5.

2.3.2 Some thoughts about schedulability for mixed criticality

Deterministic schedulability. The deterministic schedulability analysis has already provided quite contributions to MC. Complexity issues and overlay pessimism are critical for continuing investigating MC with multi-core and many-core implementations. The research project proposed will tackle with those aspects.
The research project will continue to apply deterministic approaches to schedulability analysis with realistic representations on the task behavior i.e., if a HI-criticality task or job goes in HI-criticality mode, it assumed that all the HI-criticality tasks or jobs go in HI-criticality mode. The work in [166] is a the result of an existing collaboration which aiming at reducing pessimism by applying more realistic representations. With that, new scheduling algorithms will be developed in order to guarantee timing constraints for the different criticality and the possible mode combinations at runtime.
Also, the plan is to continue exploring trade-offs between resource reservation, schedulability, and criticality level guarantees, as done in [190, 193, 198]. The trade-offs are for defining which criticality can be scheduled, and at which cost (resource utilization), as well as for defining beforehand the system conditions that can be guaranteed and those that cannot. There will be applied already existing deterministic scheduling, and the notion of deterministic sensitivity analysis with deterministic abstract representations.

Probabilistic schedulability. In terms of probabilistic schedulability analysis, the contributions to the MC scheduling problem will come from newly developed scheduling algorithms that can effectively leveraging probability information into scheduling decisions. The potential for probabilistic scheduling is to develop more efficient ("smarter") scheduling decisions based on the actual execution conditions and the probabilities that represent those.

Based upon pWCETs, execution time distributions, and/or other possible parameters, probabilistic schedulability analyses have been developed. Here a brief state of the art to explain already existing probabilistic timing analysis.
Probabilistic schedulability analysis approaches compute the probability of missing a deadline an check if it is small enough as safety requirements. Tia et al. [136] focus on unbalanced heavy loaded system, with maximum utilization larger than 1 and much smaller average utilization, and provide two methods for probabilistic schedulability guarantees. Lehoczky [93] proposes the first schedulability analysis of task systems with probabilistic execution times. This work is further extended to specific schedulers, such as EDF in [147] and under fixed priority policy in [63].
[48] provides a very general analysis for probabilistic systems with pWCET estimations for tasks. In addition to WCET estimations, statistical guarantees are performed upon the minimum inter-arrival time (MIT) estimation as well [3, 109]. Schedulability analysis based on probabilistic Execution Time[1] (pET) (instead of pWCETs) is also done in [78] for limited priority level case (quantized EDF), and in [107] where an associated schedulability analysis on multiprocessors is presented. Statistical response-time analysis, e.g. [105], can be further done to real-time embedded systems based upon those probabilistic schedulability analysis.
The works applying discrete distributions to schedulability analysis [48, 109] mainly rely on the convolution operation between distributions to find tasks discrete probabilistic response time.
As demonstrated by [110], MC problems can be approached with probabilities for quantifying and managing the unlikely events such as high criticality modes. There is a strong belief that a tighter coupling between MC and probabilistic frameworks can end up into 'smart' schedulers for more efficient utilization of the computational resource. The strategy here would be to continue researching and pushing for more efficient probabilistic scheduling and more formalized and effective probabilistic sensitivity analysis. This will allow to take full advantage of the flexibility from the probabilistic task models with scheduling decisions from probabilities.
It is worthy to mention recent works on probabilities applied to the MC scheduling problem which will inspire the research project. Probabilistic analysis for the static mixed criticality and adaptive mixed criticality schemes is derived in [110, 111]. This valid research activity can be extended with the use of probabilistic calculus or probabilistic bounding functions.
The flexibility from probabilistic models should be used into system design, especially with scheduling policies that actively apply probabilities into both off-line and runtime decisions. The aim of those algorithm is to improve overall performance, especially with soft real-time systems. Actual works have not yet explored such flexibility. This is due to the fact that it is extremely complex to do schedulability analysis with probabilities – all the values from every task have to be combined for reliable results. Existing probabilistic schedulability analyses are empirical solutions and not structured as formal methods. So far, they rely on appropriate mathematical analysis to ease the obtaining and the guaranteeing of the results.
With the MC research project, the modes for schedulability analysis are in the MC sense as well as in the adaptivity sense: conditions after the mode change and during the mode change are guaranteed. In the past, I made some works about probabilistic schedulability analysis are [189, 197]. Also, I made some early works on probabilistic schedulability analysis specific to MC [165, 191]. Those contributions will continue to inspire the exploration of the MC scheduling problem and will serve to actively apply probability into scheduling decisions.

Complexity and formalism for MC probabilistic scheduling. The complexity of probabilistic schedulability approaches has driven recent studies that tends to apply formal methods for developing probabilistic schedulers. Among the formal methods, it is possible to list the use of stochastic Markov chain [43], or the use of stochastic Petri nets [107]. In particular, in the latter case, it is possible to derive exact result of probability of deadline miss along with the trace representing the probability of task being executed at a certain time.
The idea of applying formal methods and model checking is to benefit of the formal mathematical support of such approach, in order to develop models which can be validated/checked. The models are used to describe

[1]Probabilistic execution times are execution time profiles as empirical distribution made with all the possible execution times of a task; the probability associated to each execution time is the probability of happening for that execution time.

task and scheduling behaviors. *Formally and correctly representing the system dynamics allows also for more efficient scheduling decisions.*

Also, Markov decision processes can be used in [9, 10] to model job releases in MC systems. Probabilistic analysis is also used to investigate the safety of each criticality level [49]. Current works assumes that task execution times are independent. This is an unrealistic assumption, but one that will be weakened in future work.

Faults and fault effects have to be accounted into MC schedulability analysis as they contribute defining the criticality level of tasks. Various methods applies to perform fault diagnosis [15, 76]; they can be combined with adaptivity and reliability analysis for safety critical systems [16, 64, 88].

Some attempts with formal methods and real-time systems have been made in collaboration with Jasdeep Singh, during his thesis at the ONERA [201, 203]. Also, among Jasdeep Singh works there are results obtained applying formal methods to probabilistic MC problems [200, 202]. Such works tackle with the extreme complexity of the MC schedulability with the use of easier to validate formal methods and model checkers. Models of the MC system are built and formally validated in order to increase the confidence on the correctness of the models as well as of the analysis developed on top of them. With the PhD thesis of Jasdeep Singh, it is started such investigation of formal methods and their application for real-time scheduling problems. Future work is planned to continue such investigation with probabilities, formal methods and model checkers.

An early contribution with formal methods involved stochastic Petri Nets for developing probabilistic schedulability conditions [152]; stochastic Petri Nets will also be applied to the MC scheduling problem.

In [194], a first attempt of developing schedulability analysis that is able to consider some fault effects into task executions. With this work, abstract fault effects are embedded into both timing analysis and schedulability analysis. Future contribution will be devoted to proficiently make use of probabilistic MC task models with different fault effects into probabilistic schedulability analysis. This will contribute building the MC schedulability analysis in presence of faults.

The details about all those contributions and further developments with probabilistic MC schedulability analysis are given in Chapter 4 and Chapter 5.

Chapter 3

Contributions to timing analysis and schedulability analysis

In this chapter are listed some of my research papers and projects made from 2003 until 2015 i.e., the past research, equivalently safety critical research. They all are tools and background, that are used or will be used in order to approach MC. Such background composes of timing analysis approaches and of schedulability analysis approaches for real-time embedded systems. In it, it has been approached both theoretical and industrial needs related to modeling, analysis, and verification of real-time embedded systems.

The works on timing analysis are mainly about probabilistic timing analysis applied for deriving pWCET estimates. They have started during the Postdoc thanks to EU project PROARTIS, and have continued at the ONERA. Probabilistic timing models are from the need for understanding where worst-case models are from, and what is the complexity for deriving them, especially with multi-core platforms. Thus, instead of regarding classical deterministic approaches like static timing analysis, probabilistic approaches have been considered. Classical deterministic task models can be overlay pessimistic when system models are not accurate; the probabilities can mitigate such pessimism thanks to multiple WCETs available, each with a probability associated. Also, distributions for representing worst-case behaviors can cope better with execution conditions that can happen at runtime, since they are directly inferred from nominal behaviors of task or systems.

Guaranteeing that probabilistic representations are worst-case modeling, and quality modeling of system dynamics are among main problems. There are ongoing collaborations with static timing analysis groups for coupling measurement-based probabilistic timing analysis with deterministic static timing analysis. Future contributions will improve the model quality as well as the understanding of system dynamics from measurements information. Moreover, measurement-based probabilistic timing analysis will benefit from system models to enhance quality and confidence of pWCET estimates.

Works on schedulability analysis apply deterministic approaches made since the PhD thesis, and probabilistic approaches made since the Postdoc.

The deterministic schedulability analysis was the subject of my PhD thesis. It marked the first steps within real-time and allows defining predictability for single-core and distributed real-time embedded systems. Those works tackled with the adaptivity problem for dynamic systems and the challenge of guaranteeing timing constraints with changing execution modes. Hard real-time predictability has been enforced for safety critical systems. Deterministic schedulability analysis has been the subject of Francesco Propseri PhD thesis at the Scuola Superiore Sant'Anna, and of Tomasz Kloda thesis at the ONERA.

There is also the contribution on probabilistic schedulability analysis which is first motivated by the need for schedulability analysis that can apply probabilistic task models. Its purpose is for providing more flexible schedulability conditions which can cope with the different safety levels that real-time tasks can have. Probabilistic schedulability conditions generalizes deterministic ones adding to the deterministic condition many other conditions, each with associated a probability; the probability defines the confidence on respecting timing constraints; with probabilities, both soft and hard real-time can be studied. Probabilistic schedulability analysis has been the subject of Dorin Maxim PhD thesis at INRIA Nancy, and is the subject of Jasdeep Singh thesis at the ONERA.

Past research is grouped as:

- *Deterministic and probabilistic timing analysis.* Deterministic modeling is presented as background and used for the deterministic schedulability analysis proposed. So far there are not made contribution on deterministic static timing analysis. The only research produced is on probabilistic timing analysis, in the form of static probabilistic timing analysis [154, 158] and measurement-based probabilistic timing analysis [148, 149, 153, 156, 159, 162, 192, 192]. only some of those works are detailed in the following.
- *Deterministic and probabilistic schedulability analysis.* With respect to deterministic schedulability, are presented the basics for deterministic bounding curves, and mostly for the sensitivity analysis applied to the abstract representations with the C-space and the (α, Δ)-space. The deterministic sensitivity analysis has been used for studying dynamic real-time systems and guarantee their adaptivity [174, 175, 186]. With respect to probabilistic approaches to schedulability, are presented works on probabilistic bounding curves, and how the notion of probabilistic schedulability is defined [151, 152, 172, 173, 188, 189, 197, 201, 203]. Also, I made some works that formalize the sensitivity analysis with probabilistic models for defining probabilistic versions of the C-space and of the (α, Δ)-space; those are presented in the next chapter because they are applied directly into the MC problem.

The first notions are about timing analysis: deterministic task models and probabilistic extensions are detailed; timing analysis contributions are about the estimation of pWCETs with static probabilistic timing analysis and measurement-based probabilistic timing analysis approaches. Next, are proposed notions of deterministic and probabilistic schedulability analysis together with the abstract representations which are applied for defining and developing sensitivity analysis. Both versions of deterministic and probabilistic versions of sensitivity analysis are presented. A contribution on deterministic schedulability analysis for dynamic real-time systems is detailed. In it, deterministic sensitivity analysis is applied to evaluate changes and trade-offs between schedulability conditions and resource changes. Then, are described the probabilistic bounding functions as extension of deterministic bounds. It is described their definition and how they apply to represent task executions, and then to do schedulability analysis. The chapter ends with two works on probabilistic schedulability analysis: the first applies probabilistic bounds, while the second makes use of formal methods to reduce complexity of previous probabilistic schedulability analysis. Remarks and future developments are outlined for each contribution.

All the works in this chapter are the results of collaborations and supervision of PhD and Master thesis; they are applied to research projects or have been inspired by those. As a reminder, the works here presented does not consider the MC; instead, they are used and will be used as tools for approaching MC.

3.1 Computational models

In this section there are presented the basis for deterministic and probabilistic computational models for real-time.

3.1.1 Deterministic models

Hereby it is presented what is a real-time application with its deterministic models and the notions of deterministic schedulability.

A real-time application Γ, each composed by n tasks: $\Gamma \overset{\text{def}}{=} \{\tau_1, \tau_2, \dots, \tau_n\}$. Each τ_i can be a periodic task, such that it is characterized deterministic parameters such as:

$$\tau_i \overset{def}{=} (O_i, C_i, T_i, D_i). \tag{3.1}$$

O_i is the task offset; C_i is the task WCET upper bounding any possible execution time; T_i is the task period or inter arrival time, which denotes the periodicity of task executions. D_i is the task relative deadline. All these parameters are given with the interpretation that a periodic task τ_i generates an infinite number of successive jobs $\tau_{i,j}$, with $j = 1, \ldots, \infty$. Each such job has an execution requirement described by C_i; the arrival of the jobs is described by T_i. All the jobs are assumed to be independent of other jobs of the same task and those of other tasks. This assumption is possible to to the WCET modeling which does not change from instance to instance.

A τ_i can also be an aperiodic task. An aperiodic task, also called sporadic, is a task where consecutive jobs are separated by the minimum inter-arrival time. The parameters describing aperiodic tasks $\tau_i = (O_i, C_i, T_i, D_i)$ are such that: O_i is the task offset; C_i is the task WCET upper bounding any possible execution time; T_i is the task period or minimum inter arrival time; D_i is the task relative deadline.

In this manuscript, if otherwise specified, it is considered that the task offset O_i equals to zero; in the previous chapter, to present the MC task model it has implicitly assumed $T_i = D_i$; in then following, if otherwise specified it is also assumed $T_i = D_i$.

The environment and system models describe how the system is being used by the environment: how often will system functions be called, how much data is provided as input to the system, and how much data is generated by the system back to its environment. Real-time systems have functional and mainly temporal requirements that have to be represented.

The real-time theory aims at modeling elements of a real-time system and their requirements. It also applies the abstraction of those elements to the scheduling and feasibility analysis frameworks in order to guarantee hard real-time or simply quality of service requirements in complex systems. The schedulability of a system depends on the scheduling policy and most of all on the available resource.

Hereby, the focus on Fixed-Priority (FP) and dynamic priority scheduling paradigms such as EDF. In those cases, the resource required to execute applications and the resource provided by the system are compared to derive feasibility conditions. The task models like Equation (3.1) are the classical Liu and Layland models [37, 102].

Fixed-priority scheduling. In fixed-priority scheduling any task has assigned a priority that the scheduler applies to order the task set and let at any time the highest priority runnable task actually run. There is a unique priority associated with each task, and all the jobs generated by a task are assigned this priority.
The absolute priority can be assigned in many ways i.e., the rate monotonic policy where the deadline is assigned according to the period of the tasks [94], or deadline monotonic scheduling algorithm [99], which assigns priorities to tasks in inverse order of their relative deadline parameters. Without loss of generality, it is assumed that the tasks are indexed in decreasing order of priority: task τ_is jobs have priority over task τ_js jobs for all i, j such that $1 \leq i < j \leq n$.

Definition 3.1.1 (Response time) *The response time R is the time (measured from the release time) at which the task instance is completed.*

Audisley et al. have developed a necessary and sufficient method in order to compute the interference received by the task τ_i from its higher priority tasks, [13, 14]. The basic idea is that in oder to compute the largest response time of τ_i, R_i, then the interference I_i has to be computed in the interval $[0, R_i]$. The interference is the $I_i = \sum_{j=1}^{i-1} \lceil \frac{R_i}{T_j} \rceil$, and the response time results from $R_i = C_i + \sum_{j=1}^{i-1} \lceil \frac{R_i}{T_j} \rceil C_j$. In order to get a result from the last equation and provide the maximum response time for a task, an iterative solution is obtained by:

$$R_i^s = C_i + \sum_{j=1}^{i-1} \lceil \frac{R_i^{s-1}}{T_j} \rceil C_j, \tag{3.2}$$

and the iteration ends with two consecutive R_i, (R_i^{s-1}, R_i^s) with the same value or $R_i^s > D_i$. Index s define the iteration in the fixed point formula.

From Lehoczky et al. [94], it comes the schedulability criteria for rate monotonic scheduling algorithms. It compares the workload ad the total available resource task by task. The set of all schedulability point as been refined in order to reduce the complexity of such schedulability condition [30, 94].
With the tasks ordered by priority, from higher to lower priority, where $hp(i) = \{\tau_1, \tau_2, \ldots, \tau_i\}$ denotes the sub-set of all tasks with a priority higher than or equal to τ_i.

A task resource request can be represented with the resource bound function (rbf) $\mathsf{rbf}_i(t) \overset{def}{=} \max\left\{0, \left(\left\lceil \frac{t}{T_i} \right\rceil \cdot C_i\right)\right\}$; $\mathsf{rbf}_i(t)$ upper bounds any resource request.

The level-i workload wbf_i is the resource request from τ_i and it includes all the contributions of all higher priority tasks than τ_i:

$$\mathsf{wbf}_i(t) \overset{\text{def}}{=} \sum_{i}^{hp(i)} \mathsf{rbf}_k. \tag{3.3}$$

The workload function is used into FP scheduling conditions.

Dynamic-priority scheduling. EDF is among the dynamic-priority scheduling. Under EDF, the analysis of periodic tasks with deadline less than the period can be carried out using the processor demand criterion as introduced by Baruah et al. [28]. In general, the processor demand of a task τ_i in an interval $[t_1, t_2]$ is the amount of processing time $g_i(t_1, t_2)$ requested by the instances of τ_i activated and that must be completed in that interval: $g_i(t_1, t_2) = \sum_{r_{i,k} \geq t_1, d_{i,k} \leq t_2} C_i$. For the whole task set Γ, the processor demand in $[t_1, t_2]$ $g(t_1, t_2)$ is given by the sum of the processing time of the tasks composing the task set. The feasibility of the task set is guaranteed if and only if in any interval of time the processor demand does not exceed the available time, that is if and only if

$$\forall t_1, t_2 \;\; g(t_1, t_2) = \sum_{i \in \Gamma} g_i(t_1, t_2) \leq (t_2 - t_1). \tag{3.4}$$

Theorem 3.1.2 (Processor demand criterion) *A set of synchronous periodic tasks with relative deadline less than or equal to periods can be scheduled under EDF iff:*

$$\forall t \in \mathcal{D} \;\; \sum_{i=1}^{n} \lfloor \frac{L + T_i - D_i}{T_i} \rfloor \leq t, \tag{3.5}$$

where $\mathcal{D} = \{d_k \mid d_k \leq min(t^*, H)\}$*, and* $t^* = \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1-U}$.

As a reminder, U_i is the task τ_i utilization, $U_i = \frac{C_i}{T_i}$.
The computational demand of a task set can be precisely described by the demand bound function (dbf), introduced by Baruah et al. [26]. It expresses the total computation that must be executed by the processor in each interval of time when tasks are scheduled by EDF. For any given periodic task τ_i activated at time $t = 0$, its demand bound function $\mathsf{dbf}_{\tau_i}(t)$ in any interval [0, t] is given by:

$$\mathsf{dbf}_{\tau_i}(t) = \max\left\{0, \left(\left\lfloor \frac{t - D_i}{T_i} + 1 \right\rfloor C_i\right)\right\}.$$

The task computational demand dbf_i is equivalent to g_i in $[0, t]$.

The computational demand of a task set Γ of periodic tasks synchronously activated at time $t = 0$ can be computed as the sum the individual demand bound functions of each task, that is

$$\mathsf{dbf}_{\Gamma}(t) = \sum_{\tau_i \in \Gamma} \mathsf{dbf}_{\tau_i}(t).$$

Resource provisioning. The computational resources are provided by reservation servers. A class of server algorithms that can be described by a periodic server abstraction. A periodic server S is characterized by two parameters (Q, P) where Q is the maximum budget (or server capacity), and P is the server period. A server must guarantee that Q units of time are allocated in each period P to the served application, with $Q \leq P$.

Given server S, its *supply bound function* $\mathsf{sbf}_S(t)$ is the minimum amount of time provided by S in any interval of length $t \geq 0$ [56, 61, 128].

The resource provided by a reservation server can also be described by the bounded delay function (bdf) [56, 61, 128] characterized by the pair (α, Δ), where α is the resource provisioning rate of the server and Δ is the longest interval with no resource provisioning itself. The bdf is defined in the interval domain as

$$\mathsf{bdf}(t) = \max\{0, \alpha(t - \Delta)\} \tag{3.6}$$

with

$$\alpha \overset{def}{=} \lim_{t\to\infty} \frac{\mathsf{sbf}(t)}{t} \qquad \Delta \overset{def}{=} \inf\{q \mid \alpha(t-q) \leq \mathsf{sbf}(t)\ \forall t\}.$$

The bounded delay function bdf_S of a server S is defined as a linear approximation of the resource provisioning that lower bounds the resource provisioning, $\forall t \quad \mathsf{bdf}_S(t) \leq \mathsf{sbf}_S(t)$. It is used for schedulability abstractions applied into sensitivity analysis, as described in the following.

Feasibility analysis. Using the former abstractions, the EDF schedulability of a task set Γ within a server S can be guaranteed if:

$$\forall t \quad \mathsf{dbf}_\Gamma(t) \leq \mathsf{sbf}_S(t). \tag{3.7}$$

Note that the schedulability can also be tested using the linear bounded delay function bdf, which however provides a more pessimistic condition, because of the approximation applied. An example of demand bound function, supply bound function and its bounded delay approximation is illustrated in

The FP schedulability is guaranteed if each task in Γ, with static priority ordering, has enough resource to execute within its deadline. A task set Γ executing within a server S can be guaranteed under FP iff:

$$\forall\ i\ \exists\ t\ \in SchedP\ :\ \mathsf{wbf}_i(t) \leq \mathsf{sbf}_S(t). \tag{3.8}$$

SchedP defines the set of time instances where FP schedulability has to be verified [30, 44].

So far, I did not make any research activity around the definition or the to develop deterministic task models. The static timing analysis, [143] to derive WCET estimates is assumed as reference: it has not been considered during my research activity. Also, the classical scheduling algorithms such as FP and EDF, have been considered in the works on schedulability analysis; no alternative deterministic scheduling algorithms have been developed alternative algorithms. Nonetheless, such deterministic background has been applied to various works and projects listed in the following.

3.1.2 Probabilistic models

In the following, more details about the motivations for probabilistic models and how they apply to real-time tasks. Such models are formalized, and there are presented the research for deriving those.

Before presenting the probabilistic task model, some fundamentals about probabilities are needed.

Probability fundamentals. Given a continuous random variable $\mathcal{X}$ defined in $[0, +\infty)$, the Probability Density Function (PDF) $f_\mathcal{X}(x)$ of $\mathcal{X}$ gives the probability that a value extracted from $\mathcal{X}$ lies between a and b as: $P(a \leq \mathcal{X} \leq b) \overset{def}{=} \int_a^b f_\mathcal{X}(x)dx$. In $[0, +\infty)$, $\int_0^\infty f_\mathcal{X}(x)dx = 1$. In case of $\mathcal{X}$ being a discrete random variable, the PDF (alternatively the probability mass function) gives the probability of an event x, $f_\mathcal{X}(x) \overset{def}{=} P(\mathcal{X} = x)$. The Cumulative Distribution Function (CDF) $F_\mathcal{X}(x)$ of $\mathcal{X}$ gives the cumulative probability for $\mathcal{X} \leq x$. With a continuous random variable it is $F_\mathcal{X}(x) \overset{def}{=} \int_0^x f_\mathcal{X}(y)dy$, while with a discrete random variable it is $F_\mathcal{X}(x) \overset{def}{=} \sum_0^x f_\mathcal{X}(y)$.

The Inverse Cumulative Distribution Function (ICDF) $F'_\mathcal{X}(x)$ of $\mathcal{X}$ gives the exceeding threshold probability a x as the probability that $\mathcal{X} > x$. $F'_\mathcal{X}(x) \overset{def}{=} 1 - \int_0^x f_\mathcal{X}(y)dy$. With $\mathcal{X}$ a discrete random variable, it is $F'_\mathcal{X}(x) \overset{def}{=} 1 - \sum_0^x f_\mathcal{X}(y)$.

The convolution of two continuous PDFs $f_{\mathcal{X}}(x)$ and $g_{\mathcal{Y}}(y)$, denoted by $\otimes$, refers to the summation of the random variables $\mathcal{X}$ and $\mathcal{Y}$, given as: $f \otimes g(z) = \int_{-\infty}^{\infty} f(z)g(x-z)dz$. With discrete random variables, the convolution is such that: $f \otimes g(z) = \sum_{-\infty}^{\infty} f(z)g(x-z)$. The convolution of more than two PDFs is represented as $\underset{i}{\otimes}\mathcal{C}_i$.
Figure 3.1 illustrates an example of a continuous random variable represented respectively with the PDF, the CDF and the ICDF. With the PDF it is the probability in certain interval, while with the CDF and ICDF there are the cumulative probabilities starting from O or ending at ∞ in the respective cases. Figure 3.2 illustrates an example of a discrete random variable.

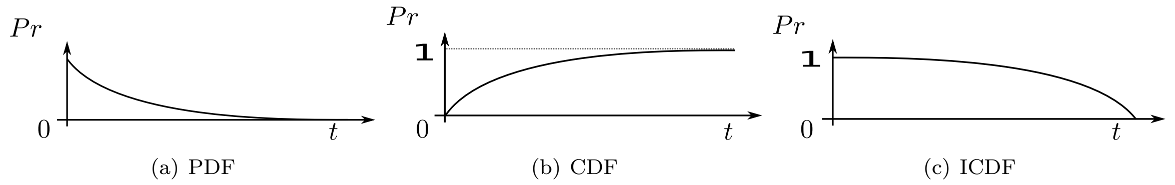


Figure 3.1: PDF, CDF, and ICDF representation of pWCET continuous and finite random variable.

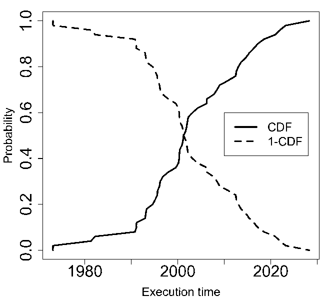


Figure 3.2: CDF, and ICDF representations of pWCET discrete and finite random variable.

Probabilistic representations such as the pWCETs attempt to describe the [epistemic] uncertainty within a system with random variables [45]. The aim is to overcome the pessimism of classical deterministic models by characterizing with probabilities, what is behind single value representations. A real-time system with probabilistic parameters described with random variables, e.g. pWCETs, is called probabilistic Real-Time System (pRTS).

Probabilistic worst-case execution time. In here, the task probabilistic parameter considered here of pRTSs is the worst-case execution time. The task execution behavior is characterized with a pWCET instead of a deterministic WCET. It is such that $\tau_i = (\mathcal{C}_i, T_i, D_i)$. T_i and D_i are deterministic values (single value parameters) and $\mathcal{C}_i$ is the worst-case execution time distribution pWCET.
Task execution behavior can change from one instance to another due to multiple interference conditions that can happen at runtime. The actual execution time of a task can be better represented with an empirical random variable to capture the variability of the runtime task behavior. The Execution Time Profile (ETP) $\overline{\mathcal{C}}$ is the discrete empirical random variable[1] defined on the finite support $\Omega_{\overline{\mathcal{C}}}$ of possible execution time values $C_{(k)}$, $\Omega_{\overline{\mathcal{C}}} = (C_{(k)})_{k \in [\![1;N]\!]}$. $\overline{\mathcal{C}}$ represents the variability of the task under specific execution conditions imposed or considered while evaluating the task execution times. Probabilistic timing analysis approaches look for pWCET distribution estimates $\overline{\mathcal{C}}$ that are able to upper bound any possible task execution behavior, thus ETPs under every possible execution condition.
Jobs of tasks can exhibit multiple duration (execution times) at run time, due to interference and changing conditions from the system elements and the environment. It is then reasonable to describe task execution time with random processes[2]. The ETP is the execution time distribution of the task obtained under specific execution conditions.

[1]The ETP is a discrete random variable since execution time C_j can only assume values multiple of the system tick. Calligraphic letters are for both random variables, discrete or continuous, and traces; non-calligraphic letters are for single value variables.

[2]Note that a random process is a sequence of random variables describing a process whose outcomes do not follow a deterministic pattern, but follow probability distributions.

A trace $\overline{\mathcal{T}}$ is a collection of execution time measurements C_j, $\overline{\mathcal{T}} = \{C_j \mid j \in [\![1;n]\!]\}$ and $\forall j \in [\![1;n]\!]$ $\overline{\mathcal{T}}(j) = C_j$. Given $\overline{\mathcal{T}}$, the ETP $\overline{\mathcal{C}}$ is the discrete empirical random variable. It is discrete since execution time C_j can only assume values multiple of the system tick. $\overline{\mathcal{C}}$ defined on the finite support $\Omega_{\overline{\mathcal{C}}}$ of possible execution time values $C_{(k)}$, $\Omega_{\overline{\mathcal{C}}} = (C_{(k)})_{k\in[\![1;N]\!]}$ and $C_{(k)} \in \mathcal{T}$.
Probabilistic timing analysis approaches look for pWCET distribution estimates $\mathcal{C}$ that are able to upper bound any possible task execution behavior. Representations for $\mathcal{C}$ are the PDF $f_{\mathcal{C}}(c)$ either continuous or discrete, the CDF $F_{\mathcal{C}}(c)$ and the ICDF $F'_{\mathcal{C}}(c)$.

$\mathcal{C}$ has to be pessimistic: it has to be larger than or equal to the exact pWCET distribution $\mathcal{C}^*$, $F'_{\mathcal{C}}(c) \geq F'_{\mathcal{C}^*}(c)$ for every c. The partial ordering between random variables is defined as in [48]. $\mathcal{C}^*$ is the unknown exact pWCET and it is definable as the tightest upper bound distribution to any possible ETP, [158]. The probabilistic worst-case execution time $\mathcal{C}_i$ of a task τ_i generalizes the deterministic WCET It is defined as the worst-case distribution that upper bounds any possible task execution time the task can exhibit, [158]. In its abstract interpretation, $\mathcal{C}_i$ would includes multiple values, each with the probability of being the worst-case of the task execution time.

With the ETP $\overline{\mathcal{C}}$ obtained under a measurement scenario, $\mathcal{C}$ has to be larger than or equal to $\overline{\mathcal{C}}$. $F'_{\mathcal{C}}(c) \geq F'_{\mathcal{C}^*}(c) \geq F'_{\overline{\mathcal{C}}}(c)$ for every c. $\mathcal{C}$ is safe if it is larger than or equal to $\mathcal{C}^*$ and any $\mathcal{C}$, $F'_{\mathcal{C}}(c) \geq F'_{\mathcal{C}^*}(c) \geq F'_{\overline{\mathcal{C}}}(c)$ for every measurement scenario.

$\mathcal{C}$ has also to be a tight upper bound to $\mathcal{C}^*$ and to the ETPs; the tightness is for the quality of the pWCET estimates.

WCET thresholds $\langle WCET; p\rangle$ can be extracted from $\mathcal{C}$. Given a risk probability p such as $p = P(\mathcal{C} > WCET)$, $WCET$ is the worst-case value which has probability p of not being overcame at runtime.

From $\mathcal{C}_i$ of a task τ_i, it is possible to define WCET thresholds $\langle C_{i,j}, p_{i,j}\rangle$. The value $C_{i,j}$ is associated to the probability $p_{i,j}$ of being the WCET for τ_i; $p_{i,j} \overset{def}{=} F'_{\mathcal{C}_i}(C_{i,j})$ quantifies the probability that τ_i executes for more than $C_{i,j}$. $1 - p_{i,j}$ is the probability of respecting $C_{i,j}$, thus the confidence on $C_{i,j}$ of being the task worst-case execution time. Depending on the granularity of the pWCET, it would be possible to define WCET thresholds at probability of 10^{-3}, 10^{-6}, 10^{-9} and beyond. With finite support pWCETs, it exist a WCET threshold C_i^* such that $F_{\mathcal{C}_i}(C_i^*) = 1$. C_i^* would be the deterministic WCET estimation upper bounding any pWCET[1].

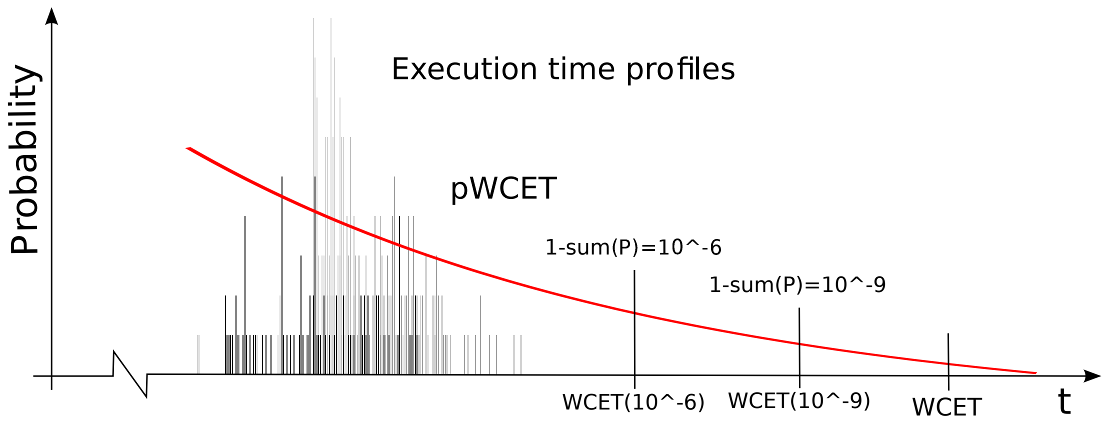


Figure 3.3: WCET and pWCET representations to the task execution behavior; pWCET and execution time profiles profiles compared.

In Figure 3.3, the pWCET is represented in its ICDF function together with thresholds at probability of 10^{-6} and 10^{-9}. In particular, it is shown how both WCET and pWCET are estimated in order to upper bound execution time profiles (as actual task execution time – normal task behavior) [201].

Worst-case independence. In order to be a worst-case distribution for τ_i, $\mathcal{C}_i$ has to include all the possible interference that τ_i can suffer [42]. This would include interference from concurrent task, environmental interference, etc..

[1]In the rest of the document, WCET thresholds, distribution values, and deterministic variables are represented with upper- or lower- case non calligraphic letters; distributions are represented with calligraphic upper-case letters.

If an interference $\mathcal{I}$ occurs during task execution, its pWCET does not change because of it: the conditional distribution $\mathcal{C}_i|\mathcal{I} = \mathcal{C}_i$ because $\mathcal{I}$ has already been accounted for by $\mathcal{C}_i$. This is the definition of statistical independence between probabilistic tasks, since the task execution distribution does not change in presence or not of interference. By considering worst-case distributions $\mathcal{C}_i$ to model task execution behavior, tasks and jobs become independent between each other. To note that this reasoning is exactly the same the is implicitly made with deterministic WCETs. Since they are worst-cases, they are able to upper bound any interference of dependence effect. Hence for, tasks and jobs are independent.

Assuming the pWCET as continuous distribution, the PDF representation of $\mathcal{C}_i$, $f_{\mathcal{C}_i}$ described the probability that a value extracted from $\mathcal{C}_i$ lies between C_1 and C_2: $P(C_1 \leq \mathcal{C} \leq C_2) = \int_{C_1}^{C_2} f_{\mathcal{C}_i}(C)dC\ \mathcal{C}_i$. The CDF $F_{\mathcal{C}_i}(C)$ gives the cumulative probability for $\mathcal{C}_i \leq C$, $F_{\mathcal{C}_i}(C) = \int_0^C f_{\mathcal{C}_i}(c)dc$. The ICDF $F'_{\mathcal{C}_i}(C)$ gives the exceeding threshold probability as the probability that $\mathcal{C}_i > C$ (C is the threshold), $F'_{\mathcal{C}_i}(x) = 1 - \int_0^C f_{\mathcal{C}_i}(c)dc$. Continuous pWCET distributions can be defined on a finite support of an infinite one.

Assuming the pWCET as a discrete distribution, the PDF representation of $\mathcal{C}_i$ $f_{\mathcal{C}_i}$ describes the probability of happening of a certain event C from $\mathcal{C}_i$, $f_{\mathcal{C}_i}(C) = P(\mathcal{C}_i = C)$, with $\sum_0^\infty f_{\mathcal{C}_i}(C) = 1$. The CDF is $F_{\mathcal{C}_i}(C) = \sum_0^C f_{\mathcal{C}_i}(c)$; the ICDF $F'_{\mathcal{C}_i}(C)$ is such that $F'_{\mathcal{C}_i}(C) = 1 - \sum_0^C f_{\mathcal{C}_i}(c)$. The values of the pWCET for a task τ_i belong to $[C_i^{min}, C_i^{max}]$ and it is assumed that $C_i^{max} \leq D_i$.

$$f_{\mathcal{C}_i} = \begin{pmatrix} C_{i,1} \equiv C_i^{min} & C_{i,1} & \dots & C_{i,k_i} \equiv C_i^{max} \\ P(\mathcal{C}_i = C_{i,1}) & P(\mathcal{C}_i = C_{i,2}) & \dots & P(\mathcal{C}_i = C_{i,k_i}) \end{pmatrix}, \tag{3.9}$$

with $f_{\mathcal{C}_i}(C_{i,r}) = P(\mathcal{C}_i = C_{i,r})$, $\sum_{r=1}^{k_i} f_{\mathcal{C}_i}(C_{i,r}) = 1$, and k_i is the number of elements composing the pWCET distribution.

Probabilistic minimum inter arrival time. There exist other task parameters which can be represented with probabilities instead of using deterministic bounds.
An alternative probabilistic task parameter could be the period or arrival time. In this case, it is $\tau_i = (C_i, \mathcal{T}_i, D_i)$. C_i and D_i are deterministic values (single value parameters) and $\mathcal{T}_i$ is the worst-case inter arrival time distribution.
$\mathcal{T}_i$ is the probabilistic minimal inter-arrival time (pMIT) with a known pdf denoted by $f_{\mathcal{T}_i}(\cdot)$. D_i is the task relative deadline.
In case of a discrete distribution setting, the values of $\mathcal{T}_i$ belong to $[T_i^{min}, T_i^{max}]$. It is assumed that $D_i \leq T_i^{min}$.

$$f_{\mathcal{T}_i} = \begin{pmatrix} T_i^0 = T_i^{max} & T_i^1 & \cdots & T_i^{l_i} = T_i^{min} \\ P(\mathcal{T}_i)(T_i^{max}) & P(\mathcal{T}_i)(T_i^1) & \cdots & P(\mathcal{T}_i)(T_i^{min}) \end{pmatrix} \tag{3.10}$$

where $\sum_{j=0}^{k_i} f_{\mathcal{T}_i}(C_i^j) = 1$ and $\sum_{j=0}^{l_i} f_{\mathcal{T}_i}(T_i^j) = 1$. Here, $(k_i + 1)$ and $(l_i + 1)$ are respectively the number of pWCET and pMIT representing task τ_i. The values of pMIT are ordered in an opposite manner than those of pWCET for sake of readability and ease of representation of the mathematical expressions.

Of course, probabilistic tasks can have both WCET and MIT probabilistically described, as well as other combinations of parameters. All those can be managed into probabilistic schedulability analysis, at the cost of an increased complexity.

The probabilistic worst-case models as random variable generalize deterministic worst-case models. They are defined as the worst-case distributions that upper bound any possible task execution time, respectively lower bound task inter-arrival time, the task can exhibit, [158]. Hence, worst-case execution time distributions represents a way to account for the system variability as the worst-case model to all of them. Probabilistic models tend to characterize what is behind the worst-case single value. They offer more flexibility in the representation, but makes schedulability analysis extremely complex.

$\mathcal{C}$, $\mathcal{T}$ and other probabilistic models have been proven to be effective in characterizing the natural variability of tasks. The predictability is achieved with probabilistic models applying them into probabilistic schedulability analysis.

The pWCET $\mathcal{C}_i$ would includes multiple WCET values, each with the probability of being the worst-case. For example, given a trace of task execution time which would be an empirical distribution due to the task

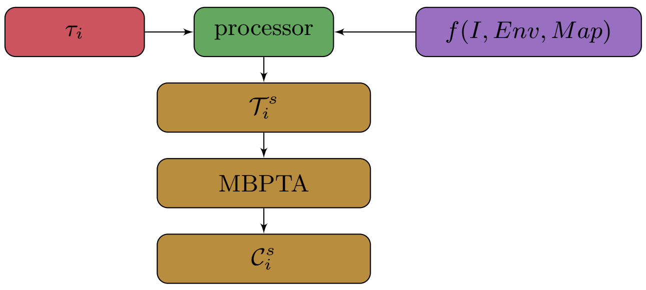


Figure 3.4: MBPTA work flow.

execution time variability, the worst-case execution time distribution could be the distribution made out of the maximum of blocks of execution times, each block representing a specific task execution condition. The pMIT $\mathcal{T}_i$ would includes multiple MIT values, each with the probability of being the worst-case.

3.2 Probabilistic timing analysis

Among probabilistic timing analysis approaches to estimate pWCETs, it is possible to distinguish between Static Probabilistic Timing Analysis (SPTA) and measurement-based probabilistic timing analysis MBPTA.

First papers on probabilistic timing modeling describe the worst-case execution time of tasks with random variables, using either discrete [3, 137] or continuous [96] distributions. Since Edgar and Burns [51], several papers have worked on obtaining safe and reliable pWCET estimates [77, 80], [156].

SPTA methods analyze the software and use a model of the hardware behavior to derive an estimate of pWCET; SPTA is applicable when some part of the system or the environment have been artificially time randomized, [158], [11]. MBPTA approaches rely on the Extreme Value Theory (EVT) for computing pWCET estimates out of measured behaviors [156]. Figure 3.4 depicts key elements for MBPTA which accepts input measurements of task execution times under specific execution conditions and applies the EVT for inferring pWCET estimates. The pWCET estimations from MBPTA are continuous distributions [156], [2], while those from STPA are discrete distributions [98].

Few works on SPTA have been developed [154, 158]. SPTA uses techniques similar to static timing analysis, applied on random replacement cache memories. As a result, they produces discrete pWCET estimates. In particular, [154] approaches SPTA with Markov chain for randomized architectures. The MBPTA instead, does only require measurements of the task execution time in different execution conditions, and from that it estimates worst-case execution time models [153, 156, 162], [101, 142]. The EVT guarantees that if certain hypotheses are verified, from the actual measured behavior it is possible to infer rare events, where the worst-case execution time lie. Whenever correctly applied, the EVT produces continuous distributions which are safe pWCET estimates $\mathcal{C}_i$.

diagXtrm [162] and `https://forge.onera.fr/projects/diagxtrm`, is a MBPTA tool developed at the ONERA by PhD Fabrice Guet in R. It proposes as an automatic tool for reliable and efficient pWCET estimations with the use of the EVT. diagXtrmapplies up-to-date statistical tests for evaluating the EVT applicability hypotheses. Moreover, it is able to provide confidence and reliability for the quality of the $\mathcal{C}_i$ worst-case probabilistic model. Finally, ii applies exhaustive search of the best distribution parameters such as the shape or the threshold to guarantee the best quality models as well as the best EVT applicability.

Figure 3.4 shows the basics of MBPTA, hence of diagXtrm, with the EVT applied to measurements of task execution time – average execution time – for inferring pWCET estimation.

Execution scenarios and worst-case profiling. With timing analysis, it is evident the dependence of the worst-case model with the execution scenario considered or the hypotheses made. In particular, this ”dependence” has a large impact with measurement-based timing analysis approaches, included MBPTA.

The guarantees that the EVT provides worst-case task models strongly depend on what has been measured, e.g. the execution conditions for the measurements, the confidence or representativity of the measurements.

A trace of execution time measurements accounts for some of the interfering conditions and inputs (to the system and tasks) which happen at runtime. The pWCET estimate $\mathcal{C}_i$ from the EVT embeds those system conditions and others which have not been measured (the so called rare events which are costly to observe by measurements) i.e., the EVT is able to infer some of the unknowns from the known measurements. Unfortunately, not all the unknowns can be estimated with the only use of the EVT.

An execution scenario $s^j = f(I, Env, Map, \ldots)$ abstracts the execution conditions the system (and the task) subdue to. s^j represents instances of the inputs (for tasks and system) I, of the environment Env, of the task mapping and scheduling policy Map, etc.; s^k is a function of I, Env, Map and more. For a real-time system, it exists a finite set Sc of all the possible execution scenarios, $Sc = \{s^1, s^2, \ldots, s^n\}$, since inputs, environment conditions, mapping, etc. are finite.

As any measurement-based approach, MBPTA with the EVT can guarantee the pWCET as worst-case estimate only for the scenario or the scenarios applied/considered by the measurements. The pWCET obtained with MBPTA is named *relative* pWCET as they are relative to only the scenarios applied, see Figure 3.4 for some details on the impact of scenarios on measurements and thus on pWCET estimates. The *absolute* pWCET has to be obtained exploring every possible scenario.
The scenarios in Sc can be ordered based on the pWCETs associated to them. There exist scenarios with worst and less worst execution time measurements and consequently worst and less pWCETs. The worst scenario s^{worst} in Sc would be the scenario that produces the worst interferences on the task and the worst pWCET. Analyzing Sc with respect to the *worst-case scenario* consists of seeking for s^{worst} and then estimating $\mathcal{C}^{s^{worst}}$ from execution time measurements under it; the worst pWCET $\mathcal{C}_i$ is $\mathcal{C}_i \overset{def}{=} \mathcal{C}^{s^{worst}}$.
Envelope: With Sc the finite set of possible measurement scenarios for the system, the worst worst-case estimate $\mathcal{C}_i$ could be defined as an envelope of all the possible probabilistic profiles: $\mathcal{C}_i \overset{def}{=} max_{s^j \in S}\{\mathcal{C}_i^{s^j}\}$. With the icdf it is:

$$F'_{\mathcal{C}_i}(C) \overset{def}{=} max_{s^j \in S}\{F'_{\mathcal{C}_i^{s^j}}(C)\}, \tag{3.11}$$

This approach to worst-case profiling is named the envelope [192]. The worst pWCET could results not necessarily from an actual scenario, but from multiple scenarios contributing to the worst-case.
Worst-case set: It is possible to define a task execution model that keeps all the scenarios estimates $\mathcal{C}_i^{s^j}$; the *Worst-case set* representation [192] collects all the pWCET from Sc as a set of pWCET estimates $\overline{\mathcal{C}}_i$:

$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^1}, \mathcal{C}_i^{s^2}, \ldots, \mathcal{C}_i^{s^{|S|}}). \tag{3.12}$$

Although with today's real-time systems it is reasonable to assume a finite number of measurement scenarios, enumerating all the scenarios remains a complex problem. What would effectively allow applying the *Worst-Case Set* representation, the envelope or the worst-case scenario is the existence of scenarios which dominate other scenarios.
Scenario dominance is in the sense that a dominant scenario s^k has larger pWCET than the pWCETs from dominated scenarios s^j; the partial ordering between pWCET can be defined according to [48].

With dominance between scenarios it would be possible neglecting the dominated scenarios and ease the task representation. For example, in case of Equation (3.12) fewer dominating scenarios could be considered to represent the whole S, $\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^k}, \mathcal{C}_i^{s^j}, \mathcal{C}_i^{s^r})$; in here, s^k dominates some scenarios in S, s^j dominates other scenarios as well as s^k and its dominated scenarios, s^r dominating all the scenarios.

The MBPTA is a relatively young approach to timing analysis. It does not have the rigor of static timing analysis [143], and it suffer the limitations of every measurement-based approach: the cost for exploring the whole space of inputs/execution scenarios.

Probabilistic representations. With probabilistic models, multiple are the possible representations for the task behavior. Hereby there are listed the *inter-scenario* and the *intra-scenario* representations, which accounts for probabilities and scenario information. Given a probability p and the WCET threshold $\langle C_i^{s^j}, p\rangle$ at p for the scenario s^j, for each scenario $s^j \in S$ the inter-scenario task set WCET threshold $\langle \overline{C}_i, p\rangle$ is such that:

$$\overline{C}_i \overset{def}{=} (C_i^{s^1}, C_i^{s^2}, \dots, C_i^{s^{|S|}}). \tag{3.13}$$

In particular, it is possible to pick $p = 10^{-9}$, $\langle C_i^{s^j}, 10^{-9}\rangle$ $\forall s^j \in S$, and have the task set WCET threshold $\langle \overline{C}_i, 10^{-9}\rangle$. The inter-scenario representation $\langle \overline{C}_i, p\rangle$ of Equation (3.13) makes use of all the scenarios (at least the dominating ones) for characterizing the task execution behavior.

For a given scenario $s^j \in S$ and a set of exceeding thresholds probabilities $(p_1, p_2, \dots p_m)$, the set of WCET thresholds $\hat{C}_i^{s^j}$ for s^j is such that:

$$\hat{C}_i^{s^j} \overset{def}{=} (\langle C_{1,i}^{s^j}, p_1\rangle, \langle C_{2,i}^{s^j}, p_2\rangle, \dots, \langle C_{m,i}^{s^j}, p_m\rangle). \tag{3.14}$$

The intra-scenario representation focuses on a specific scenario with all the meaningful WCET thresholds and probabilities for that scenario. For example, with $(10^{-6}, 10^{-9}, 10^{-12})$ it would be
$\hat{C}_i^{s^j} = (\langle C_{1,i}^{s^j}, 10^{-6}\rangle, \langle C_{2,i}^{s^j}, 10^{-9}\rangle, \langle C_{3,i}^{s^j}, 10^{-12}\rangle)$ representing the task execution behavior with s^j.

The notion of relative worst-case, representations specific to scenarios, and WCET thresholds parametrized with probabilities will be very handy in order to represent MC task models.

3.2.1 Static probabilistic timing analysis

In this section, some details of one of the two works made on SPTA with random replacement caches [154, 158]. It is only outlined [158] which is the result a collaboration with Robert I. Davis (University of York), Sebastian Altmeyer (formerly at Saarland University, now at University of Amsterdam), Claire Maiza (INP and VERIMAG Grenoble) and Liliana Cucu-Grosjean (INRIA Paris).

This paper integrates analysis of probabilistic Cache Related Preemption Delays (pCRPD) and SPTA for multi-path programs running on a hardware platform that uses an evict-on-miss random cache replacement policy. The SPTA computes an upper bound on the pWCET of the program, which is an exceedence function giving the probability that the execution time of the program will exceed any given value on any particular run. The pCRPD analysis determines the maximum effect of a preemption on the pWCET. The integration between SPTA and pCRPD updates the pWCET to account for the effects of one or more preemptions at any arbitrary points in the program. This integration is a necessary step enabling effective schedulability analysis for probabilistic hard real-time systems that use preemptive or co-operative scheduling.

Random cache replacement policy. The cache considered is a theoretical implementation [122] of a fully associative cache with an evict-on-miss random replacement policy. Here, if the requested instruction is not in the cache, then a cache line is randomly selected for eviction, and the memory block containing the instruction is fetched from main memory and loaded into the evicted location in cache. Each cache line thus has the same probability of being evicted on a miss i.e., for an N-way associative cache, the probability of each cache line being evicted is $\frac{1}{N}$ for each cache set. The aim of such a cache replacement policy is to reduce the dependency of execution times on execution history while leaving the functional behavior unchanged.

Instruction modeling

A task or program is a sequence of instructions stored in memory, thus a single path program may be represented by a sequence of symbols $a, b, c, \dots$ where each symbol represents the memory block containing the specific instruction.

Definition 3.2.1 (Re-use Distance) *Given an arbitrary sequence of instructions, then the re-use distance k of a particular instruction is defined by the maximum possible number of evictions*[1] *since the last use of the memory block containing that instruction.*

[1]Actually, the number of evictions in the same cache set; however, as it is assumed a fully associative cache, there is only one cache set.

Given the evict-on-miss policy, the second access to a in the sequence a, b, a has a re-use distance of 1. The re-use distance of an instruction dictates its overall probability of being a hit, with larger re-use distances being indicative of a higher probability of a cache miss. In the next, the calculation of these probabilities.

Any instruction is characterized by two discrete latency: i) one resulting from a cache hit $L\{hit\} = H$. H is the time to load the instruction from cache and execute it; ii) and one for a cache miss $L\{miss\} = M$. M is the time to check the cache, fetch the instruction from memory, load the instruction from cache and execute it.

For convenience, in [158], is assumed that each instruction takes the same time to execute, and hence H and M are the same for every instruction. In practice, instructions may take different times to execute, in which case this analysis can be applied with simple modifications provided that the cache miss penalty $(M - H)$ is consistent across all instructions.

Each instruction has a probability of being a cache hit $P\{hit\}$, and of being a cache miss $P\{miss\} = 1 - P\{hit\}$. At the instruction level, this results in a random variable $\mathcal{I}$ representing the execution time of instruction I based on the history of previous accesses. Formally the probability mass function of I is

$$f_{\mathcal{I}} = \begin{pmatrix} L\{hit\} = H & L\{miss\} = M \\ P\{hit\} & P\{miss\} = 1 - P\{hit\} \end{pmatrix} \tag{3.15}$$

Probabilistic real-time analysis focuses on random variables, the notion of independence among random variables, and the "summation" of random variables via the convolution operator. Also, the partial ordering among distributions can be defined according to [104]. The partial ordering among random variables (discrete and continue) enables to say that some pWCET distributions are greater than or equal to (i.e., worse than) others.

Since the execution time of a program can only take discrete values that are multiples of the processor clock cycle, the execution time of the program for a combination j of path and initial cache state is given by a discrete random variable $\mathcal{C}_j$.

Definition 3.2.2 (probabilistic Worst-case Execution Time) *The pWCET distribution $\mathcal{Z}$ of a program is defined as a tight upper bound on the execution time $\mathcal{C}_j$ of the program for all combinations j of initial cache states and possible paths. Hence, $\forall j,\ \mathcal{Z} \succeq \mathcal{C}_j$. To note that for this model, an empty cache is the worst-case initial cache state.*

The execution time $\mathcal{C}_i$ of a program path is a discrete random variable with a probability mass function, denoted by $f_{\mathcal{C}_i}(\cdot)$,

$$f_{\mathcal{C}_i} = \begin{pmatrix} C_i^0 = C_i^{min} & C_i^1 & \cdots & C_i^{n_i} = C_i^{max} \\ f_{\mathcal{C}_i}(C_i^{min}) & f_{\mathcal{C}_i}(C_i^1) & \cdots & f_{\mathcal{C}_i}(C_i^{max}) \end{pmatrix} \tag{3.16}$$

where $f_{\mathcal{C}_i}(c)$ is the probability that the execution time is c $f_{\mathcal{C}_i}(c) = P\{\mathcal{C}_i = c\}$ on any given run. Alternatively, the execution time can be specified using its Cumulative Distribution Function (CDF), denoted by $F_{\mathcal{C}_i}(\cdot)$, where $F_{\mathcal{C}_i}(x) = \sum_{c=0}^{x} f_{\mathcal{C}_i}(c)$, Or by the 1-CDF denoted by $F'_{\mathcal{C}_i}()$, where $F'_{\mathcal{C}_i}(x) = 1 - \sum_{c=0}^{x} f_{\mathcal{C}_i}(c) \equiv P\{\mathcal{C}_i \geq x\}$.

Now it is derived for each instruction in a single-path program, a lower bound on the probability of a cache hit. This lower bound is crucially independent of the previous sequences of cache hits and misses, and instead depends only on the re-use distance. Hence, it is shown how SPTA can be used to determine the pWCET distribution for single-path programs assuming an evict-on-miss random cache replacement policy. Extensions to SPTA for the multi-path case are detailed in [158].

Probabilities from cache behavior

With an evict-on-miss random cache replacement policy, the probability of evicting a given line is $1/N$ on each miss, where N is the number of cache lines in a set. (As it is assumed a fully associative cache, N equates to the total number of cache lines). In 2010, Zhou [146] gave the following formula for the *overall* probability of a hit on a particular access to such a cache

$$P^{hit} = \left(\frac{N-1}{N}\right)^k \tag{3.17}$$

where k is the re-use distance of the instruction.

Unfortunately, with the evict-on-miss policy, the probability that the second occurrence of an instruction is a hit is not independent of whether previous intervening instructions are hits or misses, neither does a sequence of all misses necessarily provide the worst-case scenario. This lack of independence is reflected in the conditional probability. If it is known that memory block a was not evicted because it is observed a hit, then the probability that b was evicted instead may be higher than if we observed a miss for a; effectively there is a dependency via the finite size of the cache.

For an evict-on-miss policy, consider the sequence of instructions represented by memory blocks b, c, b, a with a cache of size $N = 2$. If the second access to b is a hit, then both b and c must be residing in the cache at this point, and hence the conditional probability that the second access to a is also a hit is zero. This means that the joint probability that the second accesses to both a and b are hits is zero. This differs from the probability of 1/16 that would be obtained by assuming that all accesses were independent and could potentially be misses causing evictions.

Computing conditional probabilities is exponential in the re-use distance and so quickly becomes intractable. Instead, it is derived lower bounds on the probability of a hit that are a function of the re-use distance but independent of the pattern of hits and misses for intervening instructions. This is achieved considering the maximum amount of information that could be known due to the behavior of intervening instructions (e.g. by them being hits). An upper bound on this information is obtained by assuming that the intermediate addresses are all unique and remain in the cache for all of the re-use distance. This reduces the effective size of the cache available to the instruction of interest.

In the case of an evict-on-miss policy, for an instruction with a re-use distance of k (given by the number of intervening instructions located in different memory blocks), then the probability of a hit can be lower bounded for each value of $h = 0 \cdots k$, where h is the number of intervening instructions that are hits. Each such instruction reduces the effective cache size by 1, but also reduces the number of evictions by 1, hence a lower bound on the probability of a hit $P^{hit}(h)$ given h hits among the intervening instructions is given by:

$$P^{hit}(h) = \left(\frac{N-h-1}{N-h}\right)^{k-h} \tag{3.18}$$

provided that $h < N$, otherwise the effective cache size is zero, as is the lower bound probability of a hit. Thus a lower bound on the probability of a hit assuming that any arbitrary number of intervening instructions may be hits is given by:

$$P^{hit} = \begin{cases} min_{h\in\{0,\ldots,k\}} \left(P^{hit}(h)\right) & \text{if } k < N \\ 0 & \text{if } k \geq N, \end{cases} \tag{3.19}$$

It is possible to show that the series of values given by $P^{hit}(h)$ for $0 \leq h \leq k$ is monotonically increasing for $k < N$, and hence $h = 0$ gives the minimum value, and $h = k$ the maximum value of 1. (A proof is given in the appendix of the technical report associated to the paper and published by the University of York). Equation (3.19) can be simplified as follows:

$$P^{hit} = \begin{cases} \left(\frac{N-1}{N}\right)^{k} & \text{if } k < N \\ 0 & \text{if } k \geq N, \end{cases} \tag{3.20}$$

Observe that Equation (3.20) is monotonically non-increasing with respect to k.

P^{hit} delivers a lower bound on the probability of a hit $P\{hit\}$ that is independent of the previous pattern of hits or misses. Hence substituting $P\{hit\} = P^{hit}$ in Equation (3.15) delivers both an upper bound on the 1-CDF of the instruction, and a distribution that can be convolved.

It is interesting to compare the formula for P^{hit} given by Equation (3.20) for the evict-on-miss policy, with the formula for P^{hit} given by Equation (1) in [156] for the evict-on-access policy; reproduced below with an adjustment for the different definitions of re-use distance k.

$$P^{hit} = \begin{cases} \left(\frac{N-k-1}{N-k)}\right)^{k+1} & \text{if } k+1 < N \\ 0 & \text{if } k+1 \geq N, \end{cases} \tag{3.21}$$

Observe that evict-on-miss dominates evict-on-access as it provides a value of P^{hit} that is larger for all values of $k < N$. There are two reasons for this: evict-on-miss results in smaller re-use distances[1] and evict-on-access reduces the effective size of the cache by the number of intervening instructions k. As an example, with $N = 256$, $k = 104$, $P^{hit} = 0.5$ with evict-on-access and $P^{hit} = 0.66$ with evict-on-miss.

Static probabilistic timing analysis

For a single-path program, SPTA computes the pWCET distribution as the joint distribution of all of the composing instructions. As the lower bound probability of a hit for each instruction, given by Equation (3.20), is valid irrespective of the previous sequence of hits and misses, there is effectively independence and hence the pWCET $\mathcal{C}$ can be obtained via convolution. probability

$$\mathcal{C} = \mathcal{C}_{I_1} \otimes \mathcal{C}_{I_2} \otimes \dots, \tag{3.22}$$

where I_i are the instructions composing the program, with $\mathcal{I}_i \equiv \mathcal{C}_{I_i}$.

Assuming an evict-on-miss random cache replacement policy, the only information needed to characterize a memory access is its re-use distance. A sequence of n instructions, is thus representable by the corresponding sequence of re-use distances

$$\mathbb{Q} = \{k_1, k_2, \dots k_n\}. \tag{3.23}$$

Further, as convolution is commutative, such a sequence $\mathbb{Q}$ can be reordered without changing the final result i.e., the computed pWCET distribution. For ease of use later, $\mathbb{Q}$ is ordered by increasing re-use distance.

Probabilistic cache related preemption delay. Now the study the effect of preemption, referred to as the pCRPD, on the pWCET of single-path programs. Extensions to the multi-path case are given in [158]. First, the modeling of the effect of preemption on single instructions, from this it is derived analysis of the preemption effect on multiple instructions due to preemption at a specific point in the program. Via the concept of dominance, then it is computed an upper bound on the preemption effect at any arbitrary point in the program, and finally the effect of multiple preemptions at arbitrary points.

Assuming a sequence of instructions for a program, P_p refers to a preemption point after the p-th instruction in the sequence, hence P_1 refers to preemption after the first instruction. Preemption at point P_p changes the sequence of instructions executed by effectively inserting a sub-sequence of new instructions. These new instructions are executed prior to the program being resumed and its remaining instructions being executed. Instructions belonging to the program that are contained in memory blocks that are accessed both prior to and after point P_p have the re-use distance of their first occurrence after point P_p increased as a result of the preemption. $\mathbb{D}_p$ represents the set of instructions (memory blocks and their re-use distances) affected by preemption at point P_p. Instructions that are in memory blocks not accessed prior to P_p or not accessed after P_p do not suffer any change in their re-use distances. (To note that the sets of memory blocks whose re-use distances are affected by preemption have some similarities with the sets of Useful Cache Blocks used in the analysis of deterministic cache replacement policies [92]). The increase in re-use distances provides a way of bounding the effect of preemption on a per instruction basis, and hence the effect on the overall execution time of the program.

Single instruction bounding. For a single affected instruction I, preemption has the effect of changing its distribution from $\mathcal{I}$ to $\mathcal{I}'$. To note that the latency do not change but the probabilities do, and they change according to Equation (3.20) such that if k is the re-use distance of the instruction without preemption, then the re-use distance with preemption becomes $k' = k + d$, where d is the maximum number of evictions that could be caused by the preemption. Hence preemption decreases the probability of a cache hit and increases the probability of a cache miss.

[1]If the re-use distance for evict-on-miss is k, then it is at least $k + 1$ for evict-on-access

Modeling the increased re-use distance in this way gives a safe upper bound on the preemption effect, but requires precise information about the increase in the re-use distance caused by the preemption (i.e., knowledge of the task or nested tasks that are preempting).
A simpler upper bound which is obtained by assuming that preemption flushes the cache, i.e., evicts all of the cache contents. This can be modelled via the random variable $\mathcal{B}_I$ representing the bounding instruction with $P^{hit} = 0$ and $P^{miss} = 1$, which equates to an instruction with an infinite re-use distance, $k = \infty$.
At the instruction level, intuitively the preemption effect is the difference between $\mathcal{I}$ and $\mathcal{B}_I$. The larger this difference is, the bigger the preemption effect is on instruction I.

Preemption effects on a single instruction. To obtain effective static probabilistic timing analysis accounting for preemption at an arbitrary point in the program there is the need of the notion of dominance among preemptions and preemption effects. First it is considered the dominance of the preemption effect on a single instruction I_x over that of the preemption effect on another single instruction I_y. It is subsequently extend this concept to preemption effects on multiple instructions.

In [158] are defined and proved notions such as preemption dominance for instructions, convolution monotonicity, and instruction dominance.

Preemption effects on multiple instructions. Preemption effects affecting multiple instructions. These instructions may be described using the $\mathbb{D}$ notation introduced earlier, for example $\mathbb{D}_1 = \{a^1\}$ represents the set of instructions and memory blocks affected by a preemption at point P_1, similarly $\mathbb{D}_5 = \{b^3, c^2, d^2, a^5\}$ represents the set of instructions and memory blocks affected by preemption at point P_5. Preemption effects can be expressed in terms of the $\mathbb{Q}$ notation, giving only the re-use distances, e.g. $\mathbb{Q}_1 = \{1\}$, $\mathbb{Q}_5 = \{2, 2, 3, 5\}$. With this representation, it is assumed that the values are sorted, smallest first.
The aim is to determine an upper bound on the effect of preemption at any arbitrary point in the program. This requires the concept of dominance between different preemption points, and hence dominance between the corresponding sets of affected instructions and their re-use distances. For mathematical convenience and without loss of generality, it is assumed that the sets of re-use distances for each preemption point are padded with infinite re-use distance values so that they are all of the same length. For example, $\mathbb{Q}_1 = \{1, -, -, -\}$ is equivalent to $\mathbb{Q}_1 = \{1\}$. To note that this does not change the preemption effect represented, as replacing an infinite re-use distance instruction by $\mathcal{B}_I$ results in no change. In any case, such padded values will not appear in the final analysis.

Definition 3.2.3 (Dominance for preemption points) *The preemption effect due to preemption at point P_x is said to dominate that at point P_y if $\mathcal{X}_{P_y} \succeq \mathcal{X}_{P_x}$ where $\mathcal{X}_{P_y}$ ($\mathcal{X}_{P_x}$) is the convolution of the distributions of the extended (padded) set of instructions affected by preemption at point P_y (P_x).*

The binary operator min^+ applies to our extended (padded) sets of re-use distances. Assuming $\mathbb{Q}_i = \{k_{i,1}, k_{i,2}, \ldots, k_{i,n}\}$ where $k_{i,r}$ is the r-th smallest re-use distance in $\mathbb{Q}_i$, and similarly for $\mathbb{Q}_j$, and that $\mathbb{Q}_i$ and $\mathbb{Q}_j$ are sorted, smallest value first, and in their extended form, i.e., with the same cardinality, $|\mathbb{Q}_i| = |\mathbb{Q}_j|$.

$$min^+(\mathbb{Q}_i, \mathbb{Q}_j) = \{k_r = min(k_{i,r}, k_{j,r}) \ \forall \ r \leq |\mathbb{Q}_i|\} \tag{3.24}$$

Hence for the sets $\mathbb{Q}_1 = \{1\}$, $\mathbb{Q}_5 = \{2, 2, 3, 5\}$ referred to earlier, $min^+(\mathbb{Q}_i, \mathbb{Q}_j) = \{1, 2, 3, 5\}$.

Theorem 3.2.4 (Preemption point dominance) *The effect of preemption at point P_x* dominates *the effect of pre-emption at point P_y if $\mathbb{Q}_x = min^+(\mathbb{Q}_x, \mathbb{Q}_y)$, where $\mathbb{Q}_x$ and $\mathbb{Q}_y$ are the extended representations of the re-use distances of the instructions affected by preemptions at points P_x and P_y respectively.*

Theorem 3.2.4 and the min^+ operator allow us to construct the preemption effect of a virtual preemption point P^* that dominates the effect of preemption at any point, and hence upper bounds the effect of preemption at any arbitrary point in the program.

$$\mathbb{Q}^* = min^+(\mathbb{Q}_1, \mathbb{Q}_2, \ldots, \mathbb{Q}_n) \tag{3.25}$$

To note that $\mathbb{Q}^*$ does not include any infinite re-use distances, but may include re-use distances obtained from a number of real preemption points. Hence the preemption effect captured by this virtual preemption point is a safe upper bound, but may be pessimistic. The virtual preemption point depends on the program and all its possible preemption points.

Theorem 3.2.5 (Dominant preemption point) *The upper bound $\mathcal{C}^P_{i,P*}$ on the pWCET distribution of the program assuming the preemption effect represented by the virtual preemption point P^*, $\mathbb{Q}^*$ computed by Equation (3.25), is* greater than or equal to *the upper bound on the pWCET $\mathcal{C}^P_{i,P_x}$ assuming preemption at any arbitrary point P_x (i.e., $\mathcal{C}_{i,P^*} \succeq \mathcal{C}^P_{i,P_x}$), with $P_x \in \{P_1, P_2, \ldots, P_n\}$.*

An upper bound on the pWCET of a program assuming a single preemption at any arbitrary point can therefore be obtained by applying the effect $\mathbb{Q}^*$ of the virtual preemption point (obtained from the $\{P_1, P_2, \ldots, P_n\}$ possible preemption points via Equation (3.25)) to the complete sequence of instructions of the program and their re-use distances, as represented by $\mathbb{Q}^{PROG}$. This is achieved for each element in $\mathbb{Q}^*$ by checking for a matching element in $\mathbb{Q}^{PROG}$ and if found, replacing that element in $\mathbb{Q}^{PROG}$ with the bounding instruction $\mathcal{B}_I$. The set of instruction distributions represented by the modified $\mathbb{Q}^{PROG}$ may then be convolved to produce an upper bound pWCET for the program which is valid for a single preemption at any arbitrary point.

Demonstration, examples and test cases applied to validate the solution proposed as SPTA can be seen in [158].

3.2.2 Measurement-based probabilistic timing analysis

In this section, it is presented the foundation of MBPTA detail one of the first work done at the ONERA in collaboration with Fabrice Guet and Jerome Morio [162]. This paper is used to recap more works resulting from the collaboration with Samuel Jimenez Gil (University of York), Iain Bate (University of York), George Lima (Federal University of Bahia), Adriana Gogonel (INRIA Paris), Liliana Cucu-Grosjean (INRIA Paris), Konstantinos Bletasas (CISTER Porto), Kostiantyn Berezovskyi (CISTER Porto), Eduardo Tovar (CITER Porto), Fabrice Guet, Jerome Morio [148, 149, 159, 163, 192].

In [162], it is proposed the first structured and formal MBPTA approach that makes use of the EVT and other statistics for both the pWCET estimation problem and the average execution time behavior characterization problem. The proposed framework defines automatic tests for reducing uncertainties and unreliabilities that current MBPTA has due to non-optimal testing and to subjective decisions; it offers offers the most complete set of tests to verify EVT applicability to the pWCET estimation problem. With it: i) it is investigated for the first time the modeling of uncertainties due to parameter estimation and the choice of the probabilistic laws for characterizing the pWCET of tasks; ii) it is evaluated the error on the estimation, from applying the tool to real embedded systems execution traces; iii) it is analyzed the complexity in terms of mathematical modeling and model parameter characterization.

Reliable pWCET estimates

The EVT is a widely used theory for predicting the improbable i.e., giving probabilities of occurrence to extreme behaviors.

Under the hypothesis of independent and identically distributed (iid) execution time measurements $C_1, \ldots, C_n$ from an average discrete cumulative distribution function $F_\mathcal{C}$. The EVT ensures that the limit law of the maxima, i.e., the extreme execution times, denoted by $M_n = \max(C_1, \ldots, C_n)$ is a Generalized Extreme Value Distribution (GEV) H_ξ under norming constants such as the shape parameter $\xi \in \mathbb{R}$, the mean $\mu \in \mathbb{R} > 0$ and the variance $\sigma^2 \in \mathbb{R} > 0$ of the extreme execution times, with the Fisher-Tippett-Gnedenko theorem [53, 72].

This result implies that $F_\mathcal{C}$ belongs to the *Maximum Domain of Attraction* of the GEV H_ξ, denoted by $F_\mathcal{C} \in MDA(H_\xi)$. Given $\mathcal{C}$, whenever the iid hypothesis is respected and under good norming constants, the GEV is an appropriate distribution for the extreme execution times.

Depending on the value of ξ, the GEV can be either the Frechet ($\xi > 0$), the Gumbel ($\xi = 0$), or the Weibull ($\xi < 0$) distribution. In previous works the pWCET distribution has been assumed to be Gumbel, while here

no assumption is made about the resulting GEV distribution and so there is no restriction on the values that ξ can take. The objective of the study is to get reliable pWCET estimates so that the distribution has to best-fit the measurements: the Gumbel can result from the best-fit or it can be imposed afterwards.
Considering $\mathcal{C}$ and $F_{\mathcal{C}}$, the CDF of the peaks $\mathcal{C} - u$ above the threshold u knowing $\mathcal{C} > u$ is

$$F^u(c) = P\{\mathcal{C} \leq u + c \mid \mathcal{C} > u\} = 1 - \frac{1 - F_{\mathcal{C}}(u+c)}{1 - F_{\mathcal{C}}(u)}. \tag{3.26}$$

If $F_{\mathcal{C}} \in MDA(H_\xi)$ then the limit law of the peaks is given by the Pickands theorem [120]:

Theorem 3.2.6 (Pickands theorem) *$F_{\mathcal{C}} \in MDA(H_\xi)$ iff*

$$\lim_{u \to c_0} \sup_{0 \leq c \leq c_0 - u} |F^u(c) - GPD_\xi(c)| = 0, \tag{3.27}$$

where c_0 is the potential WCET of τ. GPD_ξ the Generalized Pareto Distribution with the same shape parameter ξ as H_ξ, and F^u from Equation (3.26).

The Pickands Theorem states that for values above a threshold, the nearest the threshold is to the actual WCET (which is the task execution time right end-point for increasing values) the more the distribution of execution times tends to a Generalized Pareto Distribution.

Definition 3.2.7 (Generalized Pareto Distribution) *The distribution function GPD_ξ is the Generalized Pareto Distribution (GPD) defined as:*

$$GPD_\xi(c) = \begin{cases} 1 - (1 + \xi \times (c-u)/\alpha_u)^{-1/\xi} & if\ \xi \neq 0 \\ 1 - \exp(-(c-u)/\alpha_u) & if\ \xi = 0, \end{cases} \tag{3.28}$$

with $\alpha_u = \mu - \xi(u - \sigma^2)$, and defined on $\{c, 1 + \xi(c-u)/\alpha_u > 0\}$.

This fixes the basis of the EVT POT approach which consists in extracting the execution time measurements from $\mathcal{T}$ above a threshold u and fitting the experimental CDF with the continuous distribution function P_ξ. By applying the POT approach to the trace of execution times, the pWCET estimate which is the distribution of extreme execution times $\mathcal{C}^\lambda$ is a GPD.

For applying the EVT, one needs i) independent and ii) identically distributed execution time measurements from iii) a distribution in the Maximum Domain of Attraction of a GEV of shape parameter ξ. Those three elements are the hypotheses to check for having reliable pWCET estimates.

In practice, the independence hypothesis is difficult to assume because of history dependence in memory components as explained in the following. Moreover, the true distribution of the execution times is unknown and prevents from proving that execution times are identically distributed from a distribution in the Maximum Domain of Attraction of a GEV.

Further researches in the EVT domain proved the convergence of the Fisher-Tippett-Gnedenko theorem for stationary execution time measurements under two conditions [90, 91], and so the applicability of the EVT in the more general stationary case. The conditions especially relax the strict independence of the measurements and it is not necessary to know precisely the probabilistic law of the execution times as soon as they are stationary. The strict hypotheses that prevented EVT applicability to non time-randomized embedded systems (todays systems), notably the independence, are so released allowing to apply the EVT to the pWCET estimate problem for any real-time system (both time-randomized and non time-randomized).

A DIAGnostic tool for estimating the pWCET with the eXTReMe value theory – diagXtrm

The main challenge of the MBPTA is the definition of a systematic approach that provides reliable pWCET estimates with the EVT. The reliability of a process comes from its definition: it is crucial to well identify the hypotheses and to choose both powerful tests and a proper parameter estimate process. A test is said to be powerful if it is able to reject a hypothesis when it is known to be false but also not reject it when it is known to be true. The reliability of the pWCET estimates holds if every hypothesis of the EVT is verified. Making use of the well defined tests and a proper estimate of the distribution parameters, here ξ and α_u, the reliability can be guaranteed.

The DIAGXTRM, by construction tends to reduce the sources of uncertainty that lie on the execution time measurements to fulfill the EVT hypotheses and the selection of the threshold [126]; moreover it quantify the estimates confidence. In that sense, the tool is a diagnostic of the statistical modeling with the EVT.

The tool is designed as a logical workflow which checks the applicability of the EVT with specific tests. For an input trace of execution times, DIAGXTRM provides a pWCET estimate $\mathcal{C}^\lambda$ and an associated confidence with regard to the EVT applicability hypotheses. The hypotheses to check on the trace of execution time measurements are: 1) stationarity, 2) short range dependence, 3) local independence of the peaks, 4) empirical peaks over the threshold follow a GPD. The four hypotheses define the hypothesis testing blocks includes in the main steps of the tool. They are carefully described in [162, 192]. In this section the DIAGXTRM is presented at a high level; the tests that compose it will be detailed in the following section.

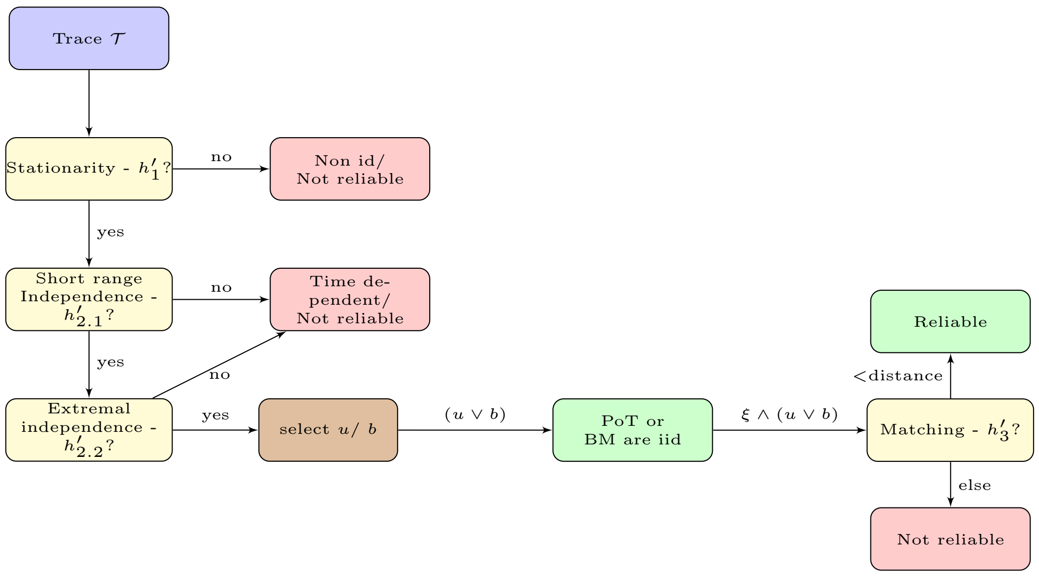


Figure 3.5: MBPTA decision diagram with tests and action applied.

Figure 3.5 describes the MBPTA decision diagram based upon the generalized EVT. This is an improved version of DIAGXTRM presented in [192] which extends the one in [162]. In here, the hypotheses to be verified compose the MBPTA control flow under peak over threshold or block maxima versions of the EVT.

Design of the tests. DIAGXTRM is mainly based on the hypothesis testing theory that studies the rejection of a null hypothesis H_0. If H_0 is not rejected it is a necessary but not sufficient condition to satisfy H_0. The first step is to select an appropriate metric that evaluates the possibility of rejecting H_0, then the metric is applied to the trace $\mathcal{T}$ of execution time measurements returning a *result* through which making a decision about H_0. In the design phase of the test, training sets are used to quantify the power of the metric for detecting H_0. The focus is on the conditional probability to reject H_0 knowing that H_0 is true $p-value = P(\overline{H_0}|H_0)$, which is the false positive rate of the test. The arbitrary threshold to reject H_0 is the value α defining the confidence interval for the test, hence for the hypothesis testing. A test may have a symmetric confidence interval, a two-sided test, otherwise this is a one-sided test. If the *result* of the applied metric to $\mathcal{T}$ is within the confidence interval, then H_0 is not rejected. Usually, α is chosen near 0, e.g. 0.01, 0.05 or 0.1, and corresponds to as many critical values *cv*s like in the two-sided test. If the *result* is near 0, then H_0 is rejected but one has $\alpha \times 100\%$ of rejecting wrongly H_0.
In DIAGXTRM, the possibility to fulfill H_0 [84] is considered, and the fuzzy logic approach to test hypotheses is applied. As the possibility to fulfill H_0 increases and so the confidence in H_0, the necessity to reject it decreases.

Fuzzy logic is widely used to build decision making processes and is called robust statistics [34, 62, 84] when applied to statistics by quantifying the uncertainties associated to classical statistical approaches.
For instance, instead of having or not a stationary trace of ETs, fuzzy logic quantifies whether the trace is near or far from the stationary model. The nearest it is the more confidence there is in the EVT applicability. Instead of one α level, 4 values are selected so that it is possible to either reject H_0 or accept H_0 with low, medium, high and full confidence level, corresponding to the $p-values$ 0.01, 0.025, 0.05 and 0.1. To resume, the approach hereby described formalizes the pWCET estimation with EVT defuzzifies the statistical test by associating fuzzy $p-values$ to human-understandable confidence levels.

Decision making process. Each hypothesis testing block, blocks 1), 2), 3) and 4) in Figure 3.5, provides a *result* about the trace of execution times and so a confidence levels. Those confidence levels aims at reliable pWCET estimates with the EVT with regard to its hypotheses applicability. One purpose of the fuzzy approach is to have a common scale for every test in order to aggregate each confidence level and to get a final confidence level on the pWCET estimate with the EVT. There exist many ways to aggregate the confidence levels, but one requirement is to have an aggregation in agreement with the tool specifications. In particular, the reliability is ensured when every hypothesis is guaranteed.
In the proposed process, four are the hypotheses to check: 1) the stationarity, 2) the short range dependence, 3) the local dependence of the peaks and 4) the matching with a GPD model. The final confidence level is denoted by $cl_{reliability}$ as a possibility metric to fulfill the whole process. Consequently, the confidence levels associated to each hypothesis are: cl_1, cl_2, cl_3 and cl_4. To satisfy the reliability requirement, if one confidence level is zero then the reliability has to be zero too. The confidence in the model is the barycenter of all the confidence levels so that it leads to the algorithm in [162]. Alternatively, if $\min(confidence\_levels) \geq 1$ it is:

$$cl_{reliability} = (cl_1 + cl_2 + cl_3 + cl_4)/4 \text{ else: } cl_{reliability} = 0.$$

Let H_0*: the EVT is applicable to* $\mathcal{T}$ be a null hypothesis, then $cl_{reliability}$ gives the confidence in fulfilling H_0. With regard to the algorithm in [162], either H_0 is rejected for a null $cl_{reliability}$, or H_0 is accepted and in this case the higher $cl_{reliability}$ is, the more confidence in the model there is and thus in the pWCET estimates. The power of the tool to fulfill H_0 and to provide reliable pWCET estimates, depends also on the selected tests for each hypothesis. diagXtrm represents the reliability with spider plots in which the cls are illustrated.
Work [162] concludes with multi-core test cases and experimental evaluation.

The diagXtrm is a high level methodology to provide reliable pWCET estimates, and one may easily replace a selected test in its respective hypothesis testing block by a better one thanks to new research works in time series (trace) analysis. As a reminder, the pWCET estimated with MBPTA approaches are relative pWCET depending on the conditions considered by the measurements. The EVT is able to infer some rare events from the measurements (the known) but only related to the scenario exercised by the measurements.

3.2.3 Some conclusions on probabilistic timing analysis

pWCET estimates aim at reducing the pessimist of deterministic WCET bounds with more flexible task models: there are considered multiple possible WCET thresholds each with the confidence associated. The pWCET gives more insight on what happens to execution time smaller than the deterministic WCET. From that, the better flexibility from probabilistic worst-case models.

The main problem with probabilistic timing analysis is the strong dependence with hypotheses and conditions assumed for the measurements or the system behavior. This reflects into the notion of worst-case specific to the hypothesis/condition considered, e.g. relative pWCET $\mathcal{C}^{s^j}$, absolute pWCET. To note that this is exactly the same problem happening to static timing analysis with multi-core processors [106]. With such platforms, the complexity is such that timing analysis is decomposed with hypotheses and WCETs are estimated according to the hypothesis considered. Then, it exists the WCET in isolation where the task executes in isolation within a core, and other WCETs can be estimated adding possible behaviors i.e., execution hypotheses, to the isolation case.

To note that the trend of separating hypotheses and execution conditions copes well with the MC problem and timing models which differentiates among criticality levels. This makes probabilistic models such as pWCET handy for representing tasks with multiple criticality and to characterize behaviors for multi- and many-core platforms.

SPTA sees the timing analysis from the same perspective as the deterministic static timing analysis: with system and task models, worst-case bounds are computed from worst-case hypotheses that simplify the behavior of the system. The risk with SPTA is to obtain overly pessimistic pWCET estimates from hypotheses that do not represent accurately the behavior of tasks i.e., in case of preemption. SPTA applies only with randomized replacement in cache memories. The papers on SPTA are [154, 158] and have been applied to real-time architectures with randomized memories. In the first work [158], it has tackled with convolution of distributions and preemption effects on cache memories. In the second paper [154], it has been applied the Markov chain formalization to represent randomized cache behaviors. With the Markov chain formal methods, into the pWCET estimates are embedded the costs for cache hit and cache misses.

MBPTA infers pWCET estimates from measurement of the system executions, and not from traditional models. MBPTA empirically build the model from the measurements; unfortunately, it suffers from the challenge of guaranteeing the worst-case conditions with just measurements. The works on measurement-based approaches are [148, 149, 153, 154, 156, 158, 159, 162, 177, 180, 182, 192]. The projects related are PROARTIS, IRHEDO2, R2D2, MAMA, CAPACITES, and CAPHCA. Fabrice Guet PhD Thesis has largely contributed to this subject, including the tool DIAGXTRM[1] that he has developed and the ONERA is using in projects. Moreover, the probabilistic models have been objects of works with Master students Alessandra Melani, Corentin Damman, Gregory Edison, Gregor Vindry, and Julien Durand that have faced multi-core platforms. Within project CAPHCA, the collaboration with Postdoc Guillaume Phavorin is to extend DIAGXTRM applicability to realistic multi-core real-time systems.

MBPTA and EVT in their classical formalization applies only if the measurements satisfy the independence and identical distribution hypotheses. This makes difficult to use with realistic real-time embedded systems, in which there manifest dependence effects from one execution to the next. In [148, 149, 159, 162, 180, 182, 192], it has been formalized the generic definition of EVT which applies in case of "weak dependence": with certain degree of dependence (local and extreme independence), the EVT can be applied and still provide safe worst-case representations. Under weak dependence, MBPTA apply can to realistic platforms as proved by the contributions made.

SPTA and MBPTA are relatively young approaches to task timing analysis and pWCET estimates, and require further development. With respect to SPTA, research activities are necessary to reduce the pessimism, and to extend its applicability to other randomized systems. For MBPTA, research is needed in order to increase the reliability of the MBPTA methodology in each of its composing steps, and it is also needed the definition of representativity of the measurements to be linked with pWCET estimates confidence and pWCET estimate safety.

An important future work will develop the "hybrid approach" to timing analysis which can take advantage of the strong points of [deterministic] static timing analysis and measurement-based approaches. At one end, the effectiveness of MBPTA requires only measurements of execution times, has to be applied to evaluate the quality of deterministic WCET bounds and the assumptions made to derive them. MBPTA can help static timing analysis to reduce his pessimism, especially with multi-core implementations. On the other end, an improved knowledge of the system will be obtained via static timing analysis. That can improve the safety of the MBPTA with more guarantees to have experienced the wort-case scenarios with measurements. Mixing the two approaches will allow reducing the pessimism of task worst-case execution time estimates. The development of the hybrid approach is in collaboration with IRIT Toulouse, Prof. Christine Rochange and Prof. Hugues Casse.

[1] `https://forge.onera.fr/projects/diagxtrm` as open source tool form probabilistic timing analysis.

3.3 Deterministic and probabilistic schedulability analysis

This section illustrates deterministic and probabilistic approaches to schedulability analysis. First are presented deterministic approaches, then are presented probabilistic approaches with probabilistic bounds and formal methods. Section starts with abstract representations which are used to define the notion of deterministic and probabilistic sensitivity analysis. The sensitivity analysis represents schedulability conditions.

3.3.1 Abstract models for sensitivity analysis

The sensitivity analysis to real-time scheduling aims at investigating the impact that task parameters have on the application schedulability. Classically, the sensitivity analysis applies to deterministic task models in order to study the impact that task parameters such as C and T have on schedulability, [186], [32]. With some enhancements, the sensitivity analysis can be applied to probabilistic schedulability analysis (the so called probabilistic sensitivity analysis), and also to both deterministic and probabilistic MC. The probabilistic sensitivity analysis, specific to pRTSs, could reduce the complexity of probabilistic schedulability analysis. it is applicable to probabilistic real-time frameworks and extend the impact evaluation to probabilities.

In the following are presented two deterministic abstract models for real-time systems schedulability under EDF or FP that are used to develop sensitivity analysis. Also, are presented their probabilistic extensions which are derived from probabilistic task models. All of them are applied into research works to define and apply sensitivity analysis with both deterministic and probabilistic task models; [186] is an example of deterministic sensitivity analysis, [194] is an example of probabilistic sensitivity analysis.

(α, Δ)-space

A resource reservation mechanisms i.e., servers, S_i can be described by the tuple (α_i, Δ_i), where α_i represents its bandwidth and Δ_i the worst-case delay in supplying the computational resource to the application. An application can also be mapped into the (α, Δ)-space, as its feasibility region, considered as the set of all service supply pairs that guarantee the application timing constraints. Note that such a region depends on the applied scheduling policy.

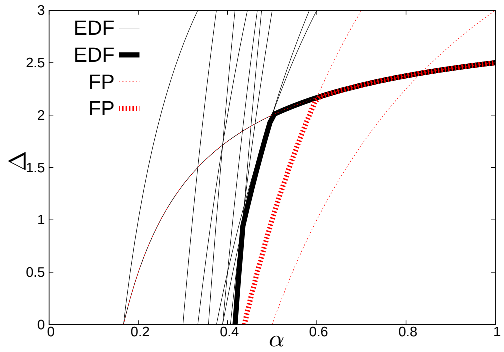


Figure 3.6: (α, Δ) space comparing EDF with FP.

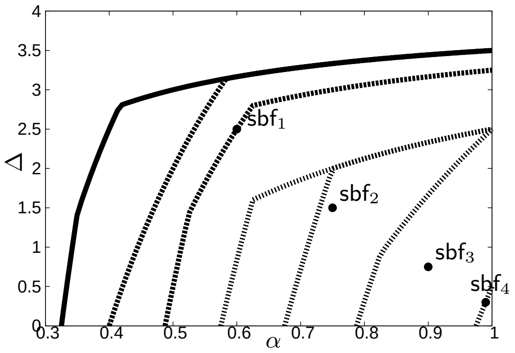


Figure 3.7: (α, Δ)-space and the sensitivity analysis that applies to it.

The feasibility region Φ_Γ in the (α, Δ)-space for Γ is defined from the FP schedulability condition and the definition of lsbf. Thus, $\forall i\ \exists t \in SchedP_i\ :\ \mathsf{wbf}_i(t) \le \alpha(t-\Delta)$ means that $\forall i$ it exists a t such that $\Delta \le t - \frac{\mathsf{wbf}_i(t)}{\alpha}$. For all i, $\Delta \le \max_{t \in SchedP_i} \left\{ t - \frac{\mathsf{wbf}_i(t)}{\alpha} \right\}$, and $\Delta \le \min_i \max_{t \in SchedP_i} \left\{ t - \frac{\mathsf{wbf}_i(t)}{\alpha} \right\}$.

The feasibility region Φ_Γ for Γ is defined from the EDF schedulability condition and lsbf. $\forall\ t \in \mathrm{D}\ :\ \mathsf{dbf}(t) \le \alpha \cdot (t - \Delta)$ means that $\forall\ t\ \in\ \mathrm{D}\ :\ \Delta \le t - \frac{\mathsf{dbf}(t)}{\alpha}$, and $\Delta \le \min_{t \in \mathrm{D}} \left\{ t - \frac{\mathsf{dbf}(t)}{\alpha} \right\}$.

Given the schedulability condition with demand bound function and lsbf, it exists the (α, Δ)-space where to represent resource provisioning and resource requests/demands, [186]: an application Γ can be mapped into the (α, Δ)-space with its feasibility region Φ_Γ as the set of all service supply pairs (α, Δ) that guarantee the application timing constraints (schedulability). The feasibility region Φ_Γ in case of EDF is such that $\forall\ t \in \mathrm{D}\ :\ \mathsf{dbf}(t) \leq \alpha \cdot (t - \Delta)$, which means that $\forall\ t\ \in\ \mathrm{D}\ :\ \Delta \leq t - \frac{\mathsf{dbf}(t)}{\alpha}$, and $\Delta \leq \min_{t \in \mathrm{D}} \left\{ t - \frac{\mathsf{dbf}(t)}{\alpha} \right\}$. Similarly, with fixed priority and level-i workload, it is possible defining (α, Δ)-space feasibility regions.

With probabilities there exist multiple probabilistic feasibility regions depending on the WCET thresholds $\overline{C}$ applied. Each $\langle \Phi_\Gamma(\overline{C}), P \rangle$ is a feasibility region $\Phi_\Gamma(\overline{C})$ from the WCET thresholds selected $\overline{C}$ and the probability P associated to it. P is the same as the one of $\langle \mathsf{dbf}(t, \overline{C}), P \rangle$ and is the probability of exceeding $\Phi_\Gamma(\overline{C})$; it can be also interpreted as the probability of verifying the condition $\mathsf{dbf}(\cdot, \cdot) \leq \mathsf{sbf}$, thus the 'schedulability probability'.

In the (α, Δ)-space, it is possible to define exists the Euclidean distance $(\delta\alpha = \alpha_2 - \alpha_1, \delta\Delta = \Delta_2 - \Delta_1)$ which defines the distance between two resource supply lsbf; it can be extended with probabilities and used for sensitivity analysis.

With the probabilistic (α, Δ)-space, the sensitivity analysis would:

- Evaluate the resource demand for probabilistic schedulability answering to: what is the resource that can be accommodated as function of the probabilistic performance that it is intended to achieve? For a specific probability, what are the resource provisioning necessary to guarantee the probabilistic real-time application? Resource evaluation can be applied to develop and guarantee strategies that change resource provisioning and obtain specific probabilistic schedulability levels.

- Define and explore the resource provisioning-probabilistic schedulability trade-offs. The resource provisioning and the probabilistic performance, with their mutual effects, are link together and applied into schedulability analysis. The trade-offs are to explore the effects that the WCET thresholds have on the schedulability and on the resource necessary to achieve certain probabilistic schedulability levels.

Figure 3.6 shows how to build the (α, Δ)-space for EDF and FP; also, it compares the feasibility region for EDF and for FP with the same task set. In Figure 3.7, an example of (α, Δ)-space for an probabilistic application scheduled under EDF. 7 feasibility regions are represented together with 4 possible resource modifications which can be applied in order to achieve the desired probabilistic schedulability level.

In the next chapter it is shown how the (α, Δ)-space abstract representations is used for sensitivity analysis applied to MC problems.

C-space

The C-space from [32, 66] is another abstract representation for real-time applications on which apply sensitivity analysis. The C-Space is a space built on tasks WCETs C_i, [32, 65]. Within it, it is possible to represent feasibility regions where each point $\overline{c} = (C_1, C_2, \ldots, C_n)$ inside the region represents a schedulable task set. A point $\overline{c}$ outside the feasibility region is a task set not schedulable. Each task τ_i assumes WCETs from $\overline{c}$, $C_i \in \overline{c}$.

Given the scheduling policy, in the probabilistic C-space there is the feasibility region where every point $\overline{c}$ within the region is a schedulable WCET thresholds configuration. The points outside the region do not represent schedulable WCET thresholds configurations.

From probabilistic models (pWCET and probabilistic dbf), it is possible to build the probabilistic version of the C-space (pC-space), such that each point $\overline{c}$ has a P probability associated. $\overline{c} = \{c_1, c_2, \ldots\}$ is a combination of task WCET thresholds (possible WCET thresholds, one per task, like $\overline{C}$), while the probability P is the probability of exceeding such thresholds $P = P_1(c_{1,j}) \times P_2(c_{2,k}) \times \ldots$. In the probabilistic the C-space, P cannot easily represent the schedulability probability; instead, it represents the probability of overcoming at least one of the WCET threshold.

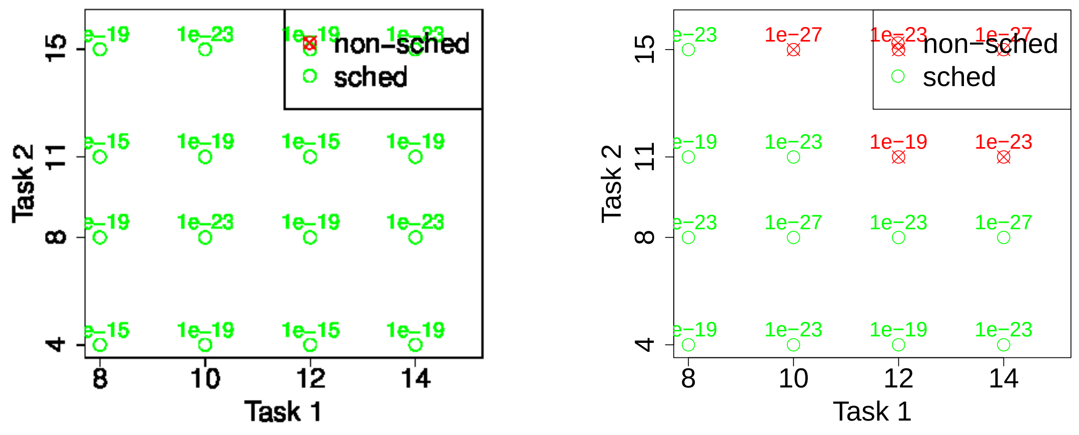
For simplicity reasons, it is presented the discrete random variable $\mathcal{C}_i$ as two arrays: $\overline{wcet} = \{c_{i1}, c_{i2}, \ldots\}$ for the WCET thresholds, and $\overline{p}_i = \{p_{i1}, p_{i2}, \ldots\}$ for the probabilities with $p_{ij} = f_{\mathcal{C}_i}(c_{ij})$; P_{ij} is the cumulative probability $P_{ij} = 1 - \sum_{k \geq j} p_{ik}$.

Assuming the EDF scheduling to schedule probabilistic real-time applications; similar reasoning can be done for fixed priority scheduling algorithms. The demand bound function dbf_i of task τ_i is the resource requested by τ_i to fully execute by its deadline. $\mathsf{dbf}_i(t) \overset{\mathrm{def}}{=} (\lfloor \frac{t - D_i}{T_i} \rfloor + 1)_0 \times c_{ij}$ and it represents the minimum resource request in order to execute the task by its deadline; c_{ij} is the task WCET. dbf_Γ is the resource demand of the whole task set Γ: $\mathsf{dbf}_\Gamma \overset{\text{def}}{=} \sum_\Gamma \mathsf{dbf}_i$.

With probabilities, it is possible defining multiple demand bound functions for τ_i, depending on the WCET threshold c_{ij} from $\mathcal{C}_i$. $\langle \mathsf{dbf}_i(t, c_{ij}, P_{ij}\rangle$ is a probabilistic demand bound function for τ_i where to the dbf_i there is associated the probability P_{ij} as the exceeding probability of the WCET threshold c_{ij} selected. P_{ij} is the probability for $\mathsf{dbf}_i(t, c_{ij})$ to be exceeded at runtime when the task executes for an execution time larger than c_{ij}, $P_{ij} = \overline{F}_{\mathcal{C}_i}(c_{ij})$. Γ is modeled with a set of probabilistic demand bound functions, each as $\langle \mathsf{dbf}(t, \overline{C}), P\rangle$ from the combination of tasks demand bound functions dbf_i. With $\overline{C} = (c_{1j}, c_{2k}, \dots)$ the array of WCET threshold values (one for each task), $\mathsf{dbf}(t, \overline{C})$ is computed as the summation of the tasks $\mathsf{dbf}_i(c_{ij})$, and P is the exceeding probability of $\mathsf{dbf}(t, \overline{C})$ such that $P = P_1(c_{1j}) \times P_2(c_{2k}) \times \dots$[1].

With the probabilistic C-space it is possible to apply Euclidean distance ($\delta c_1 = c_{1k} - c_{1j}, \delta c_2 = c_{2r} - c_{is}, \dots$) combined with probabilities, and evaluate the impact that WCET choices have on the schedulability. The sensitivity analysis for the probabilistic C-space can also work with parameter β. It is under discussion how to develop effective sensitivity analysis for the probabilistic C-space and with β. The goal is also to show a reduced complexity of the probabilistic schedulability analysis from the sensitivity analysis with the C-space.

Figure 3.8 give an example of the pC-space with points $\overline{c}$ and probabilities represented. The probabilities are the exceeding thresholds associated to the points, equivalently the confidences of not passing the execution times $\overline{c}$ at runtime. The example is 2D representations extracted from a 3D space of a real-time application of 3 tasks $\{\tau_1, \tau_2, \tau_3\}$.



(a) 2D plane (τ_1, τ_2), $C_3 = 15$ (b) 2D plane (τ_1, τ_2), $C_3 = 20$

Figure 3.8: 2D representations with safety levels HI and LO to distinguish effects on schedulability. Some possible transitions from non-schedulability to schedulability are represented.

In the next chapter, it is shown how the pC-space abstract representations is used for sensitivity analysis applied to MC problems.

3.3.2 Dynamic real-time systems

Real-time embedded systems are dynamic, in the sense that there exist changes between criticality modes at runtime. *Adaptivity means guaranteeing timing constraints during system or task changes.* Guaranteeing adaptivity of dynamic real-time systems has been part of past research activity [186, 187, 195, 196, 199, 206].

[1]The probability multiplication for P as joint probability is possible due to the worst-case distribution assumption [42]. As $\mathcal{C}_i$ are pWCETs they are independent, then the combination of dbf is independent. To remind that P_j are cumulative probabilities and their multiplication ends up into cumulative probability.

The research on dynamic systems and adaptivity for real-time represents a first step toward the guarantees for MC real-time systems.

Adaptivity for resource reservation mechanisms

In [186], in collaboration with Prof. Giorgio Buttazzo and Prof. Enrico Bini, it is addressed the problem of modifying reservation parameters in order to comply with different system requirements, specified through a set of operational modes. Provided that, the schedulability can be guaranteed within each mode in the steady state condition, the objective is to extend the schedulability analysis during mode transitions, under EDF scheduling. Thus, it is first derived resource reservation functions to bound the server resource provisioning before, after, and during mode transitions. Then, such functions are used to guarantee the resource provisioning in all the working conditions and to check the application feasibility.

In here, dynamic systems are considered and mode change analysis is developed to guarantee both steady states and mode changes that can happen in real-time systems. Also, in this work sensitivity analysis is applied for of (α, Δ)-space in order to quantify resource needs for changes in dynamic systems, and for developing mode change strategies.

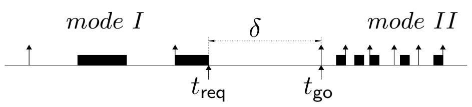
Mode change. Servers can change their parameters at runtime to cope with the system changing conditions. Whenever a server S switches from an *old mode* $S^I = (Q^I, P^I)$ to a *new mode* $S^{II} = (Q^{II}, P^{II})$, the equivalent supply bound function changes from $\mathsf{sbf}^I_S(t)$ to $\mathsf{sbf}^{II}_S(t)$. The supply bound function that describes the computational resource provided by the server across the mode transition is denoted by $\mathsf{sbf}^T_S(t)$, the transition supply bound function. The parameters of the corresponding bounded delay functions are denoted by the pairs (α^I, Δ^I), $(\alpha^{II}, \Delta^{II})$, and (α^T, Δ^T) respectively.
In the following, t_{req} denotes the time instant at which the mode change is requested. From this time on, all the required changes in the system are initiated, while the new mode begins at t_{go} after a delay $\delta \geq 0$ from the mode change initiation, hence $t_{\mathsf{go}} = t_{\mathsf{req}} + \delta$. The mode transition is the stage between the two steady modes, starting at time t_{req} and ending when the new mode becomes effective, at t_{go}.
A mode changing server can abort its resource provisioning any time during the mode transition $[t_{\mathsf{req}}, t_{\mathsf{go}}]$. The focus is on two extreme transition cases:

- *transitionA*: at t_{req} the old mode aborts, interrupting the resource provisioning until the new mode is effective, see Figure 3.9(a).

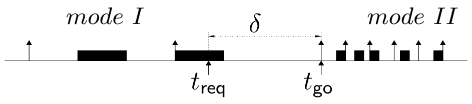
- *transitionB*: the old mode server keeps on providing its service until the new mode is effective, see Figure 3.9(b).

The two cases results in a different behavior of the server and then of the whole system. The differences will be explained in the next sections. All the intermediate cases, with abortion time t_{ab} such that $t_{\mathsf{req}} < t_{\mathsf{ab}} < t_{\mathsf{go}}$ can be easily inferred from the transitionB case.
It is assumed that the system is feasible in each mode, that is in both steady states conditions, and it has to be derived the condition in which feasibility can also be preserved during mode transitions.

Resource reservation. Although with different peculiarities, a lot of servers can be modeled as periodic servers because they guarantee to provide Q (and no more than Q) units of time in each period P. Examples of periodic servers include the polling server [37], the deferrable server [95] and the sporadic server [131], which are scheduled using fixed priority. Example of dynamic priority periodic servers include the constant bandwidth servers [4] or the dynamic versions of the Polling, Deferrable and Sporadic servers [37].
To achieve a more general result, it is performed the analysis in a worst-case scenario where the processor is allocated at the beginning of the first period and then at the end of all subsequent periods. Under this condition, all the servers mentioned above have the same supply bound function, thus they are referred as generic periodic servers. In the interval domain, the longest interval where no resource is provided is $2(P - Q)$, while after $2(P - Q)$ the server supplies the resource at a constant rate of $\frac{Q}{P}$.

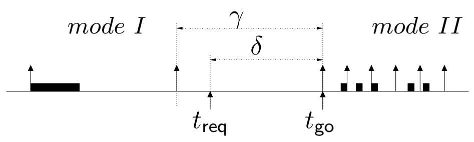


(a) TransitionA

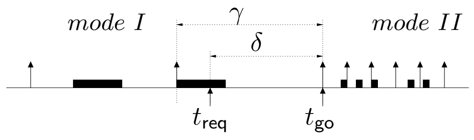


(b) TransitionB

Figure 3.9: TransitionA and transitionB; the service provisioning aborts or continues after the mode change t_{req}.



(a) TransitionA



(b) TransitionB

Figure 3.10: Server mode change and mode transition cases. Worst-case scenario resulting in the worst-case service provisioning during the mode transition stage.

The sbf of a periodic server, defined as the worst-case resource supply in time interval $[0, t)$, is

$$\mathsf{sbf}_S(t) = \max\{0, (k-1)Q, t-(k+1)(P-Q)\}, \tag{3.29}$$

with $k = \left\lceil \frac{t-(P-Q)}{P} \right\rceil$. Such a resource supply bound can be lower bounded by the bounded delay function of (3.6), with

$$\alpha = \frac{Q}{P}, \quad \Delta = 2(P-Q). \tag{3.30}$$

The computational resource the server provides is used by the application of the server.

Definition 3.3.1 (Server Schedulability) *A server is said to be schedulable if its budget is provided on time within each period.*

Definition 3.3.2 (Application Schedulability) *An application is said to be schedulable by a server if all its tasks are able to meet their deadlines.*

Definition 3.3.3 (System Schedulability) *A system consisting of multiple applications managed by a set of reservations is said to be schedulable if both servers and applications are schedulable.*

With multi-moded servers it is important to identify the minimum amount of resource provided in any stage of the server. This includes the transition between different modes, where the service provisioning is affected from both the old and the new mode. The transition supply bound function sbf^T is the resource provisioning during the transition stage, while the transition bounded delay

The following theorems provide the sbf^T functions for the two types of transitions transitionA and transitionB.

Theorem 3.3.4 (Mode change sbf - transitionA) *Let S be a periodic server with steady states parameters (Q^I, P^I) and (Q^{II}, P^{II}) and with corresponding supply bound functions sbf^I and sbf^{II}. Let t_{req} be the time at which the mode change is requested and let t_{go} be the time at which the new mode is started, after a delay δ, such that $t_{\text{go}} = t_{\text{req}} + \delta$. If the server aborts the old mode at time t_{req} (transitionA case), then the resource provisioning during the transition stage is lower bounded by*

$$\mathsf{sbf}^T(t) = \inf_{0 \le \lambda \le t} \{\mathsf{sbf}^I(t - \lambda - \gamma + P^I - Q^I) + \mathsf{sbf}^{II}(\lambda + P^{II} - Q^{II})\}, \tag{3.31}$$

being $\gamma = t_{\text{req}} - t^{\text{last}} + \delta$ and t^{last} the initial instant of the last period in the old mode.

Theorem proof can be found in [186].

Theorem 3.3.5 (Mode change sbf - transitionB) *Let S be a periodic server with steady states parameters (Q^I, P^I) and (Q^{II}, P^{II}) and with corresponding supply bound functions sbf^I and sbf^{II}. Let t_{req} be the time at which the mode change is requested and let t_{go} be the time at which the new mode is started, after a delay δ,* *such that $t_{\mathsf{go}} = t_{\mathsf{req}} + \delta$. If the server continues to provide its old mode service until time t_{go} (transitionB case), then the resource provisioning during the transition stage is lower bounded by*

$$\mathit{sbf}^T(t) = \inf_{0\le\lambda\le t}\{\mathit{sbf}^I(t - \lambda - \gamma + 2P^I - Q^I) + \mathit{sbf}^{II}(\lambda + P^{II} - Q^{II})\}, \quad (3.32)$$

with $\gamma = t_{\mathsf{req}} - t^{\mathsf{last}} + \delta$ and t^{last} the initial of the last period of the old mode.

Theorem proof can be found in [186].

The obtained sbf^Ts can be used to guarantee the service provisioning during the mode change transitions.

With the bounded delay modeling it is possible to derive the transition bounded delay functions. First, the same idea of Equation (3.31) and Equation (3.32) can be applied with bounded delay functions obtaining

$$\mathsf{bdf}^T_A(t) = \inf_{0\le\lambda\le t}\{\mathsf{bdf}^I(t - \lambda + 2P^I - \gamma - Q^I) + \mathsf{bdf}^{II}(\lambda + P^{II} - Q^{II})\},$$
$$\mathsf{bdf}^T_B(t) = \inf_{0\le\lambda\le t}\{\mathsf{bdf}^I(t - \lambda + 2P^I - \gamma - Q^I) + \mathsf{bdf}^{II}(\lambda + P^{II} - Q^{II})\},$$

for transitionA and transitionB, respectively.

The transitionA resource supply results in (α^T_A, Δ^T_A) where

$$\alpha^T_A = \min\{\alpha^I, \alpha^{II}\}$$
$$\Delta^T_A = \max\{P^I - Q^I + \gamma + P^{II} - Q^{II}, 0\}. \quad (3.33)$$

For transitionB:

$$\alpha^T_B = \min\{\alpha^I, \alpha^{II}\}$$
$$\Delta^T_B = \max\{\gamma - Q^I + P^{II} - Q^{II}, 0\}. \quad (3.34)$$

Both the transition supply bound function and the transition bounded delay functions depends on the transition delay δ, $\mathsf{sbf}^T(t,\delta)$ and $\mathsf{bdf}^T(t,\delta)$.

Equation (3.33) and (3.34) define the relationship between the transition delay δ and the transition service provisioning delay Δ^T.

Server schedulability. In multi-mode systems, an application can change its resource demand from one mode to another, thus dbf^I and dbf^{II} denote the resource demand of the application in the two modes, and dbf^T denotes its resource demand during the mode transition [118]. An application managed by a server is schedulable if and only if the resource demanded by the application does not exceed the resource provided by the server, in any possible stage of both the server and the application. Since it is assumed feasibility for each mode in a steady state condition, the following lemma states the condition for a guaranteed transition in the case both the application and the server change their mode at the same time.

Lemma 3.3.6 (Mode change EDF schedulability) *Given a server S handling an application Γ that is feasible in each in a steady state condition, if both S and Γ change their at the same time, then the application is feasible during the transition stage if*

$$\forall\ t \quad \mathit{dbf}^T_\Gamma(t) \le \mathit{sbf}^T_S(t). \quad (3.35)$$

Using the bounded delay linear approximation, the feasibility condition becomes:

$$\forall\ t \quad \mathsf{dbf}^T_\Gamma(t) \le \mathsf{bdf}^T_S(t), \quad (3.36)$$

Note that when the application increases its resource request, a short transition delay in the server adaptation is good for the application. However, a too short delay could result in a service over-provisioning that would steal bandwidth from the other servers, so jeopardizing the schedulability of the other applications during the transient. On the other hand, a large delay in the server adaptation would affect the schedulability of the application itself. In general, there is a trade-off between schedulability of the application and the schedulability of the servers.

Intra-Server Schedulability: By intra-server schedulability it is referred the schedulability of the application based on the resources provided by the server. From the analysis performed by applying Condition (3.35) or Condition (3.36), it can be derived a maximum delay $\delta^{\flat}$ after which the new mode can safely start. For all $\delta \leq \delta^{\flat}$ the server mode change is feasible for the application because it will result in a larger resource supply.

Inter-Server Schedulability: By inter-server schedulability it is referred the schedulability of the servers when they adapt their parameters, independently of the behavior of the served applications. Such an analysis can be performed using the results achieved for multi-mode task sets, which is well known in the real-time literature. The analysis can easily be applied to periodic servers by considering them as periodic tasks that must receive Q units of computational resource every period P. Therefore, the effect of a mode change in a server has to be investigated with respect to the entire set of servers composing the system.

Applying the results achieved in [71, 133] or [36], it is possible to derive a minimum mode change delay $\delta^{\sharp}$ that does not affect the schedulability of the other servers. Any change performed with a delay $\delta \geq \delta^{\sharp}$ keeps the system feasible because it reduces the amount of resource required by the server during its transition.

A real-time system with servers and applications is feasible during mode transitions if the delay is less than or equal to a threshold derived from the intra-server analysis and greater than or equal to a threshold $\delta^{\flat}$ derived from the inter-server analysis; that is, $\delta \leq \delta^{\flat} \wedge \delta \geq \delta^{\sharp}$. The resulting feasibility region for δ is then

$$\Phi = \begin{cases} 0 & \text{if } \delta^{\sharp} > \delta^{\flat} \\ \forall \delta \mid \delta^{\sharp} \leq \delta \leq \delta^{\flat} & \text{if } \delta^{\sharp} \leq \delta^{\flat}. \end{cases} \tag{3.37}$$

This result is stated in the following theorem.

Theorem 3.3.7 *Consider a multi-mode system with m servers S_i, each managing an application Γ_i, such that the system is feasible in any of its working mode. If server S_i performs a mode transition with a delay δ, the system remains feasible if $\delta \in \Phi_i$, where Φ_i is the feasibility region of server S_i from Equation (3.37). If Φ_i is empty, the transition is unfeasible under any condition.*

Theorem proof can be found in [186]. The algorithm in [186] describes how to compute a transition delay that keeps the system feasible.

Resource reservation analysis. The server and the application requirements during a mode transition are encoded into the feasibility region Φ in terms of transition delays that the set of servers and the applications can tolerate.

A server S_i can be described by the tuple (α_i, Δ_i), where α_i represents its bandwidth and Δ_i the worst-case delay in supplying the computational resource to the application. An application can also be mapped into the (α, Δ)-space, as its feasibility region, considered as the set of all service supply pairs that guarantee the application timing constraints. Note that such a region depends on the applied scheduling policy.

In the case of EDF, the application feasibility region is defined by

$$\forall t \in \mathrm{D} \; : \; \mathsf{dbf}(t) \leq \alpha(t - \Delta),$$

meaning that:

$$\forall t \; \in \; \mathrm{D} \; : \; \Delta \leq t - \frac{\mathsf{dbf}(t)}{\alpha} \quad \Delta \leq \min_{t \in \mathrm{D}} \left\{ t - \frac{\mathsf{dbf}(t)}{\alpha} \right\}, \tag{3.38}$$

where D is the set of deadlines in which the application schedulability has to checked.

Equation (3.38) describes the feasibility region $\Omega_{\Gamma,sched}$ of the application in the (α, Δ)-space, which depends on the application Γ and the scheduling algorithm *sched*, that in this case is EDF.

3.3.3 Real-time calculus and probabilities

The real-time calculus belongs to the class of so-called deterministic queuing theories. It is deterministic in the sense that hard upper and lower bounds to the performance indicators are always found. This distinguishes it from the class of probabilistic queuing theories for which this guarantee can not be provided in general.

Real-Time Calculus (RTC) [134, 141] is a direct extension of Network Calculus (NC) [89]. NC is used to deterministically reason about timing properties of data flows in queuing networks and RTC extends these concepts to the area of real-time embedded systems. In general, networked or distributed systems can be characterized by looking at the data flows over the network. It is referred to these abstracted data flows as event streams. In contrast to NC, RTC uses interval bound functions to characterize both event streams as well as available resources. The interval bound function to describe the event stream is called Arrival Curve and the interval bound function for the available resources is called service curve. The concept of arrival curve was first defined by [134].

The relation between the input arrival and service curves and the output arrival and service curves can be expressed using pure mathematical functions. Just like NC, RTC relies on max-plus and min-plus calculus, as described extensively in [38, 134, 135]. This mathematical structure is very well-suited for describing and manipulating arrival and service curves in Real-Time Calculus since these are functions that describe minimums and maximums. Convolution and deconvolution formulas, which are used in Real-Time Calculus , can be defined in min-plus and max-plus algebra.

In [150] are outlined main differences and commonalities between NC and RTC.

Probabilistic calculus

The probabilistic Calculus (pCalculus) is one of the research topic developed during the Postdoc in Nancy [172, 173, 189, 197]. With it, probabilistic bounding functions are formalized together with the algebra to compose bounds and define probabilistic interfaces.

Here some details about its pCalculus basic notions and representations. This part is also to enforce the link between RTC bounding and EDF or FP bounding with respectively demand bound function and workload function.

Tasks activations and executions time may vary from one instance to another. Each task τ_i may be associated with random traces of computation requests, depending on both the actual period and execution time of the task at each activation. One possible representation is throughout a stochastic process which counts the amount of resource requested in a time interval; then $X_i(t)$ results as the cumulative distribution function of the stochastic process describing its behavior in the interval $[0, t)$.

Considering the function $R_i(t)$ as the cumulative amount of execution that the probabilistic task τ_i has requested up to time t, it is possible to define probabilistic bounds of the task upper and lower constraining its actual behavior $\mathcal{X}_i$ with probabilities. A probability-based upper bounding function of the cumulative execution time $R_i(t)$ is defined as the "largest" $R_i(t)$ such that the probability of $X_i(t)$ being larger than or equal to $R_i(t)$ is lower than a probability value γ_i.

$$R_i^+(t) \overset{def}{=} \sup\{R_i(t) \;|\mathbb{P}\{X_i(t) \geq R_i(t)\} \leq \gamma_i\} \tag{3.39}$$

The tuple $R_i^+(t), \gamma_i$ represents a probabilistic bound to the execution trace of τ_i in the time domain and γ_i is a measure of the accuracy of such bound, being γ the probability that $\mathcal{X}_i(t)$ overtakes any of the $R_i(t)$ and consequently their superior $R_i^+(t)$. A value$\gamma_i = 0$ would be the deterministic case where $R_i^+(t)$ bounding $\mathcal{X}_i$ 100% of the time.

A probability-based lower bounding function (probabilistic bound) to the task cumulative execution is defined as the "smallest" $R_i(t)$ such that the probability of $X_i(t)$ being smaller than or equal to $R_i(t)$ is lower than a probability value ϕ_i. Formally, $R_i^-(t) = \inf\{R_i(t) \mid \mathbb{P}\{X_i(t) \leq R_i(t)\} \leq \phi_i\}$.

While any $\mathcal{X}_i$ and R_i always describe one concrete trace of an event stream, the tuple $\alpha_i(\Delta, \cdot) = [\alpha_i^u(\Delta, \cdot), \alpha_i^l(\Delta, \cdot)]$ of upper and lower request curves provides an abstract execution stream model, with bounds on all the admissible traces R_i of such execution stream, [134]: the request curve embeds information on both the arrivals and executions of tasks. The upper request curve $\alpha_i^u(\Delta, \cdot)$ is an upper bound on the number of events that are seen in any time interval of length Δ, $\alpha_i^u(t-s, \cdot) \; : \; R_i(t) - R_i(s) \leq \alpha_i^u(t-s, \cdot) \;\forall s < t$ and $\Delta = t - s$. The lower request curve $\bar{\alpha}^l(\Delta, \cdot)$ analogously provides a lower bound $\alpha_i^l(t-s) \; : \; R_i(t) - R_i(s) \geq \alpha_i^l(t-s, \cdot) \;\forall s < t$ and $\Delta = t - s$.

Parametrizing the request with a variable x, it follows that the upper bounding request curve $\alpha_i(\Delta, x) \overset{\text{def}}{=} \alpha_i(\Delta, \cdot) - x$ is such that $\alpha_i(t-s, \cdot) - [R_i(t) - R_i(s)] \geq x$ for all $0 \leq s \leq t$ and $\Delta = t - s$. Then it can be defined the probabilistic request curve, inspired from [145] but with the main difference from the probabilistic bound: in this work the probability associated to the bound $\alpha_i(\Delta, \cdot) - x$ is the probability that $\alpha_i(\Delta, \cdot) - x$ upper bounds R_i. A probability equal to 1 would correspond to the deterministic case where $\alpha_i(\Delta, \cdot) - x$ upper bounds 100% of the cases and no R_i happens to be above the bound.

The request curve $\alpha_i^u(\Delta, x)$ of a request R_i is a non-decreasing non-negative function which satisfies

$$\mathbb{P}\left\{\alpha_i^u(t-s, \cdot) - [R_i(t) - R_i(s)] \geq x\right\} \leq F_i(x), \tag{3.40}$$

for all $0 \leq s \leq t$, $x \geq 0$ and $\Delta = t - s$.
In Inequality (3.40), $F_i(x)$ is the probability that $\alpha_i^u(\Delta, x)$ upper bounds the request curve of τ_i in the interval $[s, t]$. The probabilistic request curve is $\langle \alpha^u(\Delta, x), F(x) \rangle$.

This definition introduces the notion of probabilistic curve $\langle \alpha^u(\Delta, x), F(x) \rangle$ as a set of bounding curves $\alpha^u(\Delta, x)$, one curve per value of x. Each of those curves parametrized by x has a probability associated $F(x)$ being the CDF of probability thresholds $\alpha^u(\Delta, x_j) \mid \gamma_i = F(x_j) = \mathbb{P}\{\mathcal{X}_i \leq x_j\}$. Hence, the set of curves can be described with a PDF continuous or discrete according to the variability of x. Figure 3.11(a) shows an example of probabilistic request curve.

In the same vein, it is possible to define probabilistic lower-bounding curve $\langle \alpha^l(\Delta, x), H(x) \rangle$ with bounding curves $\alpha^l(\Delta, \cdot)$ and probabilities $H(x)$.

By following the same reasoning as for tasks, it is possible to derive a probabilistic abstraction based on curves for the resource provisioning. Naming $S(t)$ the total amount of resource provided at time t, the resource provisioning is characterized by considering upper and lower bounding service curve β in the interval domain. From now on it is referred to them as resource curves. The tuple $\beta(\Delta, \cdot) = [\beta^u \Delta, \cdot), \beta^l \Delta, \cdot)]$ of upper and lower resource curves abstract the resource provisioning, and the upper and lower bound on the available resources in any time interval of length Δ, are defined such that:

$$\beta^u(t-s, \cdot) \leq S(t) - S(s) \leq \beta^u(t-s, \cdot) \forall s < t, \tag{3.41}$$

$$\beta^l(t-s, \cdot) \leq S(t) - S(s) \geq \beta^l(t-s, \cdot) \forall s < t. \tag{3.42}$$

As for the requests and generalizing the notion of resource provisioning (service), the resource provided can be modeled according to some distribution $Y(t)$, and based on probabilities it can be derived probabilistic resource bounds as $S(t)^+ = \sup\{S(t) \mid \mathbb{P}\{Y(t) < S(t)\} \leq \phi\}$ and $S(t)^- = \inf\{S(t) \mid \mathbb{P}\{Y(t) \geq S(t)\} \leq \phi\}$.

Then parameterizing the resource provisioning function with a variable y, it follows that for all $0 \leq s \leq t$ and $\Delta = t - s$, the lower bounding service curve $\beta(t-s, y) \overset{\text{def}}{=} \beta(\Delta, \cdot) + y$ is such that $[S(t) - S(s)] - \beta(t-s, \cdot) \geq y$. Thus, it is possible defining the probabilistic service curve differentiating from [145] by the definition of bounding probability: in here it is the probability that $\beta(\Delta, \cdot)$ lower bounds any resource provisioning $S(t)$.

Definition 3.3.8 (Probabilistic resource curve) *The resource curve $\beta(\Delta, y)$ of a resource provisioning S is a non-decreasing non-negative function which satisfies*

$$\mathbb{P}\left\{[S(t) - S(s)] - \beta(t-s, \cdot) \geq y\right\} \leq G(y), \tag{3.43}$$

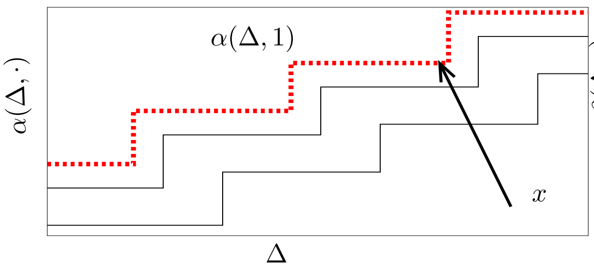
for all $0 \leq s \leq t$, $y \geq 0$ and $\Delta = t - s$.
In Inequality (3.43), $G(y)$ is the probability that $\beta(\Delta, y)$ lower bounds the resource provisioning $S(t)$. The probabilistic resource curve is $\langle \beta^l(\Delta, y), G(y) \rangle$.

With the notion of probabilistic curve for resource provisioning, $\langle \beta(\Delta, y), G(y)\rangle$ there is a set of bounding curves $\beta(\Delta, y)$, each with the changing y and a probability associated $G(y)$. $G(y)$ is the cumulative probability threshold $\beta(\Delta, y_j) \mid \phi = G(y_j) = \mathbb{P}\{\mathcal{Y} \leq y_j\}$; see Figure 3.11(b) as an example of probabilistic resource provisioning curve. In the same manner, it is possible defining the probabilistic upper bounding curve $\langle \beta^u(\Delta, y), I(y)\rangle$.

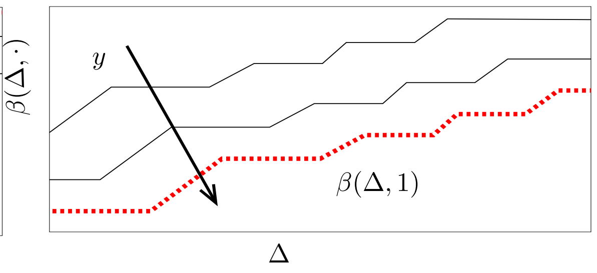
The resource is provided by system elements which execute, hence have dynamic behaviors. Although not as effective as for the tasks, the probabilistic resource curve allows to extend the probability model to the entire system having probabilistic elements.

The concept of curves simplifies and unifies many common timing models, see [108]. Besides, the probabilities allows to adjust the accuracy of bounds by thresholding the curves. Decreasing of probabilities of bounding is equivalent of having less accurate bounding curves. On the other hand, a curve able to bound 100% of the cases could be too pessimistic for the system analysis, see Figure 3.11 for a better insight. The curves are represented with increasing cumulative probability, till $p = 1$. Schedulability and composability conditions makes use only of the upper bound of the request curve and the lower-bound of the resource curve. Figures 3.12 and 3.13 gives another example of probabilistic curves and their probabilistic representations with PDF and CDF.

To develop the algebra for the real-time analysis of probabilistic curves it is needed the notion of total ordering among probabilistic curves. According to Diaz et al. [104] it is possible to define a stochastic order among random variables which is similar to the notion of first stochastic order defined in statistics.



(a) Probabilistic request curve: the probability of bounding requests decreases as the quality of the upper bound decreases, $\forall\ x_i \leq x_j,\ \alpha(\Delta, x_i) \leq \alpha(\Delta, x_j),\ F(x_i) \leq F(x_j)$.



(b) Probabilistic resource curve: the probability of bounding the resource decreases as the quality of the lower bound decreases, $\forall\ y_i \leq y_j,\ \beta(\Delta, y_i) \geq \beta(\Delta, y_j)$ and $G(y_i) \leq G(y_j)$.

Figure 3.11: Probabilistic request and resource curves.

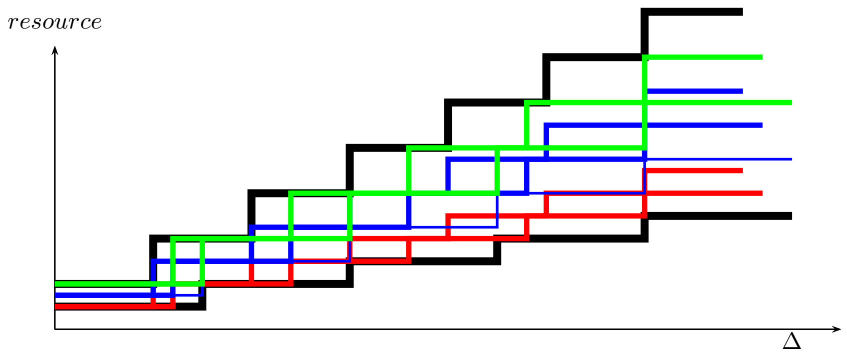


Figure 3.12: Probabilistic curve: ordered multiple curves $\alpha_i(\Delta, x)$ each with an associated probability.

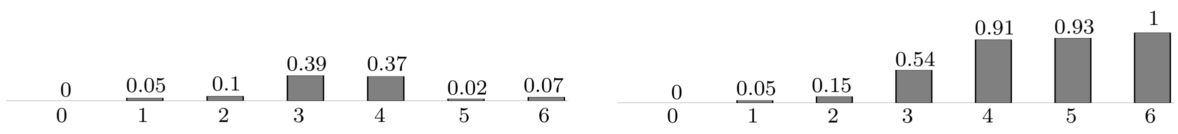


Figure 3.13: Probabilistic curve: distribution function $f_i(x)$ and cumulative distribution function $F_i(x)$ indexed by x.

As it will be shown in the next chapter, it is fairly easy to draw parallels between pCalculus and probabilistic version of demand bound function or probabilistic version of workload bound function. The probabilistic model of task execution results into probabilistic bounds which could come from the RTC basis (pCalculus) or from the probabilistic version of the dbf in case of EDF scheduling, or from the probabilistic version of the wbf_i in case of FP scheduling.

3.3.4 Probabilistic schedulability analysis

In the following are presented some works that apply probabilistic task models in order to define probabilistic schedulability conditions. There are illustrated works which exploit the flexibility of the pCalculus and of probabilistic bounds. With those tools, probabilistic schedulability conditions are defined.

A second work on probabilistic schedulability analysis applies formal methods to represent the execution of probabilistic tasks and to compute probabilistic response times of job and tasks. In there, the motivations for the use of formal methods are presented.

pCalculus and probabilistic bounds

The pCalculus is applied into probabilistic schedulability to define flexible schedulability conditions [189, 197].

A component-based view of real-time systems models each system element as a component [129]. Schedulers, resource reservation mechanisms, and everything composing the system can be modeled as a real-time component with well defined timing properties. Each component has an associated interface describing its functional and non-functional aspects, and in case of real-time components, the real-time interfaces have to include also the timing requirements of the components so that it is possible to carry out schedulability condition through interface composition.

Within the real-time calculus modeling framework, real-time interfaces involves:

- the resource provisioning β as the actual resource amount provided to the component,
- the input task flow α representing the resource demand for the component,
- the output task flow α' as the stream of executed tasks with the available resource and according the scheduler policies,
- and the residual resource β' which is the unused one and that can be passed to other components of the system.

In RTC there is the concept of output curves which are the results of the scheduling process in terms of task, then the executed task flow, and residual resource which is the unused resource to schedule the task. The probabilistic output curves can be inferred by applying the relationship among the arrival and service inputs as for the classical RTC [134]. In [83], the probabilistic output arrival curve has been defined as follows.

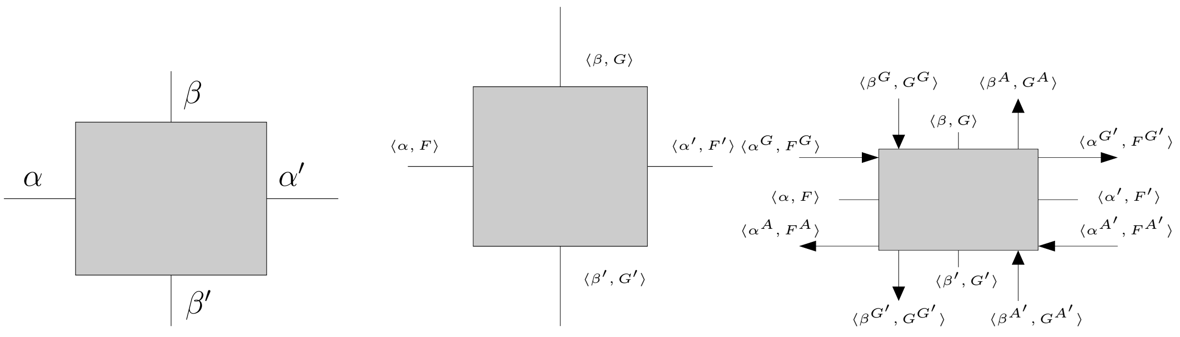
Definition 3.3.9 (Output arrival curve) *Consider a system component with a probabilistic service curve $\langle\beta(\Delta, x), g(x)\rangle$ and the probabilistic arrival curve $\langle\alpha(\Delta, x), f(x)\rangle$ as inputs. The task execution flow has a probabilistic output curve (poc) $\langle\alpha'(\Delta, x), f'(x)\rangle$ with $\alpha'(\cdot) = \alpha\overline{\oslash}\beta(t_2 - t_1)$ and the bounding function $f'(\cdot) = f * g(\cdot)$*[1]*, bounded as:*

$$P\{sup_{0\leq t_1\leq t_2}\{\alpha\overline{\oslash}\beta(t_2 - t_1) - [a'(t_2) - a'(t_1)]\} \geq x\} \leq f * g(x).$$

The probabilistic output arrival curve is represented as $\langle\alpha'(\Delta, x), f'(x)\rangle$.

The unused resource by processing real-time scheduling elements is passed to other parts of the system according to a specific strategy. It is defined a probabilistic version of the residual curve as follows.

[1] $*$ is the convolution between curves in the classical algebra; $\otimes$ is the min-plus convolution for RTC curves, and $\overline{\otimes}$ is the min-plus deconvolution.


β
α
α'
β'
$\langle\beta, G\rangle$
$\langle\alpha, F\rangle$
$\langle\alpha', F'\rangle$
$\langle\beta', G'\rangle$
$\langle\beta^G, G^G\rangle$
$\langle\beta^A, G^A\rangle$
$\langle\beta, G\rangle$
$\langle\alpha^G, F^G\rangle$
$\langle\alpha^{G'}, F^{G'}\rangle$
$\langle\alpha, F\rangle$
$\langle\alpha', F'\rangle$
$\langle\alpha^A, F^A\rangle$
$\langle\alpha^{A'}, F^{A'}\rangle$
$\langle\beta', G'\rangle$
$\langle\beta^{G'}, G^{G'}\rangle$
$\langle\beta^{A'}, G^{A'}\rangle$

(a) RTC Component

(b) RTC Interface: probabilistic bounding behaviors

(c) Assume-Guarantee real-time probabilistic component interface

Figure 3.14: A component and its probabilistic interface abstractions.

Lemma 3.3.10 (Residual service curve) *Consider a component with a probabilistic service curve $\langle\beta(\Delta, x), g(x)\rangle$ and an arrival which has a probabilistic arrival curve $\langle\alpha(\Delta, x), f(x)\rangle$. Let $b'(t_2) = b(t_2) - a(t_2)$ be the system residual resource process and $\beta'(\Delta)$ its bounded residual service curve, then $b'(t)$ is bounded by*

$$P\{[b'(t_2) - b'(t_1)] - \beta \oslash \alpha(0) > x\} \leq f * g(x). \tag{3.44}$$

$\langle\beta', g'\rangle$ is the probabilistic residual curve (prc) with $\beta' = \beta \oslash \alpha(0)^1$, and $g'(x) = f * g(x)$.

The proof of the Lemma come form [188] and is inspired by the work done in [83].

The probabilistic service curve is then $\langle\beta'(\Delta, x), g'(x)\rangle$, with the cumulative distribution function $g'(x)$ resulting from the distribution of the input curves $\alpha(\Delta, x)$ and $\beta(\Delta, x)$.

It is noticeable that probabilistic curves are set of curves, varying x, with probability thresholds associated. To each composing curve the probability associated expresses the accuracy of that curve as a bound: it is the probability that the curve upper/lower bounds arrivals/services. In particular, the curve upper bounding the arrivals with a probability of 0 is indeed, the lower bound for the arrivals, while the lower-bound of the service with 0 probability is the upper bound of the resource provisioning.

In this framework, it is considered probabilistic real-time interfaces to model real-time components with a probabilistic behavior. The probabilistic interface of a generic probabilistic real-time component is defined as the tuple $(\langle\alpha, f\rangle, \langle\beta, g\rangle, \langle\alpha', f'\rangle, \langle\beta', g'\rangle)$ where the inputs and the outputs are in terms of probabilistic curves. Figure 3.14 represents generic probabilistic interface of a probabilistic real-time component.

In a component-based real-time system the system schedulability necessarily translates into component composability so that the resulting composable system offers schedulability guarantees. This means that the composability is affected by the scheduling policy applies defining the resource distribution among the components. For example, in case of fixed priority scheduling the priority describes the composition order among the tasks. It is the resource to encode the scheduling policy as well as the composition architecture among components.

With a probabilistic component model it is possible to define a flexible composability condition which translates into schedulability of the composed components.

Theorem 3.3.11 (Probabilistic composability) *Given two components i and j and two probabilistic curves $\langle\beta_i(\Delta, x), g_i(x)\rangle$ and $\langle\beta_j(\Delta, x), g_j(x)\rangle$ being respectively the probabilistic lower bound to the resource provisioning of the i-th component and the probabilistic upper bound to the resource demand of the j-th component. Then i and j are composable with a probability p iff*

$$\exists x_1 \ : \ \forall\Delta \ \beta_i(\Delta, x_1) \geq \beta_j(\Delta, x_1) \wedge p \leq min\{1, g_i(x_1) \cdot g_j(x_1)\}. \tag{3.45}$$

Proof of the theorem can be found in [183, 189, 197].

A schedulability condition for a probabilistic real-time system with probability level equal to 1 can be derived as follows.

^{1}As defined in the classical RTC [134], and by the notion of min-plus deconvolution

Theorem 3.3.12 (1-FP schedulability) *Any probabilistic task set $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ is schedulable with a probability level* 1 *(*100%*) under FP and a probabilistic resource provisioning $\langle \beta, G\rangle$ if for all $i \in \{1, 2, \dots, n\}$,*

$$\exists\, \Delta_0 \in \mathrm{schedP}_i \mid \omega_i^u(\Delta_0, \cdot) \leq \beta^\ell(\Delta_0, \cdot). \tag{3.46}$$

Here, schedP$_i$ *represents the set of points where to verify the schedulability as defined in [30], ω_i^u is the upper bound of the schedulability probabilistic level-i workload $\langle \omega_i(\Delta, x), H_i(x)\rangle$ and β^ℓ the lower bound of $\langle \beta(\Delta, x), G(x)\rangle$.*

The theorem describes the classical real-time schedulability condition with the level-i workload. The demonstration is then straightforward from [30, 97].

The following theorem extends Theorem 3.3.12 to the case where the probability level equal to $p \in [0, 1[$, thus taking the probability function into account in the schedulability conditions.

Theorem 3.3.13 (p-FP schedulability) *Any probabilistic task set $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$ is schedulable with a probability level $p \in [0, 1[$ under FP and a probabilistic resource provisioning $\langle \beta, G\rangle$ if for all $i \in \{1, 2, \dots, n\}$*

$$\exists\, \Delta_0 \in \mathrm{schedP}_i \textit{ and} \, \exists\, x_1, x_2 \geq 0 \textit{ such that } \quad \omega_i(\Delta_0, x_1) \leq \beta(\Delta_0, x_2) \textit{ and} \, H_i(x_1) \cdot G(x_2) \geq p, \tag{3.47}$$

with $\langle \omega_i(\Delta, x), H_i(x)\rangle$ the probabilistic level-i workload.

Theorem proof can be found in [189].

The schedulability criteria in case of FP scheduling policies applied can be derived in a compositional manner. In [168] it has been derived the resource demand of a FP scheduling component assuming, without loss of generality, the task set ordered by decreasing priority. Therefore, to guarantee the satisfaction of timing constraint for task τ_n the service provided to task τ_n must be at least $\beta_n^A(\Delta, \cdot) = \alpha_n(\Delta - D_n, \cdot)$ which is the assumed resource amount the n-th task expect to guarantee its deadlines. In FP tasks are ordered according their priority, meaning that the resource for task τ_n is provided by task τ_{n-1}. This also implies that the remaining resource curve after have served $\tau_1, \tau_2, \dots, \tau_{n-1}$ must be at least $\beta_n^A(\Delta, \cdot)$. The resource requirement of the whole FP scheduling component is the resource bound of the most priority task composing the component $\beta_1^A(\Delta, \cdot)$. To derive that, it has to be sequentially computed the service bounds $\beta_n^A(\Delta, \cdot), \beta_{n-1}^A(\Delta, \cdot), \dots, \beta_2^A(\Delta, \cdot)$ that each element expects to work properly. The resource requirement of a generic task k is given as

$$\beta_{k-1}^A(\Delta, \cdot) = \max\{\beta_{k-1}^\sharp(\Delta, \cdot), \alpha_{k-1}^u(\Delta - D_{k-1}, \cdot)\}. \tag{3.48}$$

with $\beta_{k-1}^\sharp(t) = \beta_k^A(\Delta - \lambda, \cdot) + \alpha_{k-1}(\Delta - \lambda, \cdot)$ where $\lambda = sup\{\psi \ : \ \beta_k^A(\Delta - \psi, \cdot) = \beta_k\}$, that guarantees the remaining service to be passed to the k-th component to be no less than what the k-th component requires, $\beta_k^A(\Delta, \cdot)$. Equation (3.48) differs from the results of the Theorem 3.4 of [83] because of the real-time constraints that have to be guaranteed in the RTC.

In order to guarantee the timing constraint of τ_{k-1}, the service provided to the $k-1$-th element $\beta_{k-1}^A(t)$ must be no less than $\alpha_{k-1}(\Delta - D_{k-1}, \cdot)$. Applying Equation (3.48) for $k = n-1, n-2, \dots, 2$ we guarantee the tasks timing constraints by computing the resource required. $\beta_1^A(\Delta, \cdot)$ is the resource amount that has to be provided in order to satisfy the timing constraints of all the tasks scheduled: the assumed resource requirements.

Theorem 3.3.14 (Probabilistic FP composability) *A FP component is composable with a resource provisioning component that guarantees β amount of resource if*

$$\forall\, t \quad \beta_1^A(t) \leq \beta(t), \tag{3.49}$$

with β_1^A computed following Equation (3.48).

The 100% schedulability obtained with the former assumed bound $\beta_1^A(\Delta, \cdot)$ guarantee also the level 1 schedulability of the FP scheduling. The same reasoning can be applied with the level-p schedulability where the probability are computed using the resource chain.

Since the EDF scheduler refers to the schedulability condition of the whole task set instead of each task, the request curve of the task set Γ is given by $\langle \overline{\alpha}, \overline{F}\rangle$ where $\overline{\alpha}(\Delta, x) \overset{\text{def}}{=} \otimes_{i=1}^n \alpha_i(\Delta, x)$ and $\overline{F}(x) \overset{\text{def}}{=} \otimes_{i=1}^n F_i(x)$.

From these observations, the following result apply.

Theorem 3.3.15 (1-EDF Schedulability) *Any probabilistic task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ is EDF schedulable with a probability level* 1 *upon a resource provisioning curve $\langle \beta, G \rangle$ if*

$$\forall\, \Delta,\ \overline{\alpha}^u(\Delta, \cdot) \leq \beta^\ell(\Delta, \cdot). \tag{3.50}$$

Here, $\overline{\alpha}^u$ represents the request curve upper bounding $\langle \overline{\alpha}, \overline{F} \rangle$ and β^ℓ the resource provisioning curve lower bounding $\langle \beta, G \rangle$.

Theorem proof can be found in [189].

Theorem 3.3.16 (p-EDF Schedulability) *Any probabilistic task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ resulting in a probabilistic resource demand $\langle \overline{\alpha}, \overline{F} \rangle$ is schedulable under EDF with a probability level $p \in [0, 1[$ and upon a resource curve $\langle \beta, G \rangle$ if*

$$\forall\, \Delta,\ \exists\, x_1, x_2 \geq 0 \textit{ such that } \overline{\alpha}(\Delta, x_1) \leq \beta(\Delta, x_2) \textit{ and } F(x_1) \cdot G(x_2) \geq p. \tag{3.51}$$

Theorem proof can be found in [189].

By its compositional nature, EDF is more pessimistic, hence the accuracy from the probabilistic framework is more evident and needed.

3.3.5 Formal methods for probabilistic real-time

The probabilistic schedulability analysis is key topic of the thesis of Jasdeep Singh. In the thesis, it is chosen to approach such complex problems with formal methods. The models obtained with formal methods are for representing task executions under different conditions, and to develop schedulability analysis with probabilities.

With respect to pRTSs, formal methods have to be able to model (i) the non-determinism i.e., the existence of choice. This happens with preempted tasks or tasks which continue/resume execution at non-deterministic time instances; and (ii) the probability in the choice i.e., existence of weight to each choice because with pWCETs, the choices are probabilistic in nature. Among the formal methods that satisfy those requirements, the Continuous Time Markov Chain (CTMC) is the one chosen for this work. CTMC is able to model pRTSs with continuous probabilistic behaviors. This makes it possible to directly map continuous pWCETs onto a CTMC models and study the task behavior.

With the use of CTMC, it is developed an approach to model and perform schedulability analysis of pRTSs described with continuous pWCETs. For that, CTMC models are built to represent the task behaviors. The executions are mapped into CTMC states and state transitions accounting for the probabilistic nature of task executions and the probabilistic interference from concurrent tasks. The CTMC models are then applied into schedulability analysis to compute the probability of deadline miss and the probabilistic response time for each task. The correctness of the CTMC models and of the schedulability analysis is proved with model checking, where timing properties for states and state transitions are formally validated. A case study is applied to show how effective are CTMC models in representing and analyzing pRTSs.

To note that the works with CTMCs differ from [48] which also makes use of Markov Chains with pRTSs, in particular CTMCs. Markov Chain is used for modeling and analyzing jobs through a scheduling policy by accounting for all possible interactions between tasks. Instead, in [48] the MC is only to represent the computation utilization in the hyperperiod. Secondly, the two works differ in the assumption of the given input distribution: in this work are continuous distributions, while in [48] there are considered discrete distributions.

The first step of Jasdeep Singh thesis has been investigating the state of the art of formal methods applied to probabilistic schedulability analysis. Formal methods which could cope with the requirements of non-determinism and probability of choice are probabilistic timed automata, stochastic timed automata, stochastic Petri Net, stochastic model checking, and Markov chains (Mc).

Through PTA, many state transitions can emanate from a state; each transition has an associated probability [132]. With pWCETs there is no question of choice of what the task does, since it can only execute after it is released once the computational resource is made available. PTA cannot model the time spent in executing (time staying in a state), as pWCET would represent.

Another formal method which could be used with pRTSs is stochastic model checking [43]. With this, performing Monte Carlo simulation requires a source of randomness. Unfortunately, safety of the source of randomness being used in real-time system analysis is an open question.

Stochastic Petri Net can model probabilistic task executions through stochastic time triggers. Existing work on stochastic Petri Net [152] can analyze probabilistic task set at a high computational cost. On the other hand, it provides exact result of probability of deadline miss along with the trace representing the probability of task being executed at a certain time.

The Mc is a set of states and transitions in which each transition is labeled with the probability of being chosen [115]. Markov Chain possesses the memoryless property in which the determination of future state depends only on the present state. Some works apply Mc to model cache memories with random replacement policies [39]. They are SPTA approaches, and use Mc to compute discrete pWCETs. Another set of works has applied Mc with schedulability analysis [48]. In there, the Mc is only to represent the system utilization and study its stability in the hyperperiod; the Mc does not model task executions nor task interactions as it is planned to do with CTMC.

This brief state of the art on formal methods for real-time has been made by Jasdeep Singh during his stage at the ONERA. The CTMC is the formal methods applied in [201, 203].

Continuous time Markov chain. CTMC is a Mc which is able to characterize the task continuous execution behaviors [115]. With CTMC, it is possible to represent the task executions and to build schedulability analysis with probabilistic information; both task representations and analysis can be validated with model checking. The complexity of the approach with CTMC is proven to be far less than other existing methods, such as [152]. This is mainly due the highly reduced requirement of distribution convolutions and the sequential nature of the process implemented.
A CTMC has an exponentially distributed rate λ associated to state transition: (i) the time spent in a state is exponentially distributed, and (ii) there exist a probability of choosing an outgoing transition from that sate, [86]. The probability of choosing a state transition r with rate λ_r out of m outgoing state transitions is given by $P(\lambda_r) = \frac{\lambda_r}{\sum_{k=1}^{m} \lambda_i}, r \leq m$, and the rate of exponentially distributed time spent in each state P_k, denoted by Λ, is given as $\Lambda = \sum_{i=1}^{m} \lambda_i$.
The transitions for a CTMC 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.
The CTMC states represent the execution of each job after its arrival, after preemption and at the end of execution. The exponential rates for a state describe the job execution; they include the delays caused by any interfering job. The transitions between states are labeled with exponential rate and represent the probabilistic time of executing until the end of execution.

Properties. For a job J_{ij}, a property of its CTMC is a function:

$$\mathcal{P}_{ij}(\text{state}, \text{time}) = P(\text{state}, \text{time}). \tag{3.52}$$

It refers to the CTMC of the job J_{ij} and returns the probability that the CTMC is in the state *state* at the time *time*. The argument *time* could be a single instant or an interval. Examples are $\mathcal{P}_{ij}(\text{finished}, \text{deadline})$ checks the CTMC model of the job J_{ij} and returns the probability that it has finished execution at the deadline.
PRISM Model Checker [87] is used to verify these properties. In
urlhttps://forge.onera.fr/projects/probscheduling it has been developed an interface to relate CTMC models with PRISM model checker.

The notion of backlog is used to represent task interference and to build CTMC models. Hereby, the definition of backlog and the classification of various ways in which backlog is produced.

The backlog is the delay or the ‘push’ that a victim job experiences from a higher priority job. The backlog can come from:

- A job released earlier - a job is released earlier than the victim job, thus its execution delays the victim job;
- Synchronously released jobs - higher priority jobs and the victim job are released synchronously. The higher priority jobs execute first and thus delay the victim job;
- Jobs preempting - higher priority jobs preempt the already executing victim job. From the preemption instant the victim job is delayed and awaits execution.

The backlog changes the distribution of the victim job to incorporate the affect resulting in a new exponential distribution. More details in [201, 202, 203].

For the job J_{ij}, an initial set of states $X_{ij} = \{P_0, F\}$ is given. State P_0 represents the execution of job as soon as it arrives. State F represents end of execution. If there exists a preemption to the job i.e., if set $\hat{J}(J_{ij})$ is non-empty, a new state is added. For k^{th} preemption this state is P_k. Thus the complete representation of the job is a set of states $X_{ij} = \{P_0, P_1, ...P_K, F\}$. State F can be reached from any state and it is the final state of the CTMC as shown in Figure 3.15. The initial state of X_{ij} is P_0 (or P_k such that there is a non-zero labeled transition going out of it with minimum k). The states are linked with transitions given by a Q-matrix as follows with 0 representing nonexistence of transition.

$$\begin{bmatrix} & P_0 & P_1 & P_2 & \dots & P_K & F \\ P_0 & -(\lambda_{10} + \lambda_{f0}) & \lambda_{p_0} & 0 & \dots & 0 & \lambda_{f_0} \\ P_1 & 0 & -(\lambda_{p_1} + \lambda_{f_1}) & \lambda_{p_1} & \dots & 0 & \lambda_{f_1} \\ P_2 & 0 & 0 & -(\lambda_{p_K} + \lambda_{f_2}) & \dots & \lambda_{p_K} & \lambda_{f_2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P_K & 0 & 0 & 0 & \dots & -\lambda_{f_K} & \lambda_{f_K} \\ F & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

The subscript of λ for a transition denotes the final state, f if it goes to state F or the k if it goes to P_k state, and k^{th} state from which it goes out. For example λ_{f2} denotes rate of transition from state P_2 to state F.

Rates determination. The unknowns to be calculated are the rates in the Q-matrix. Here, block 'Backlog' is referred. If there is no preemption, i.e; $k = K$, $\lambda_{f_K} = \lambda^*_{f_0}$. For all positively increasing $k > 1$, the model is checked using the block 'Property' with the property $\mathcal{P}(P_k, t_p^k)$ and new state P_{k+1} is added. The addition of new state expands the Q-matrix by adding the rates λ_{p_K}, λ_{f_k} and $\lambda_{f_{k+1}}$ at the appropriate places in the matrix. Once all the job models are in hand, probability of deadline miss and response time curve can be extracted from

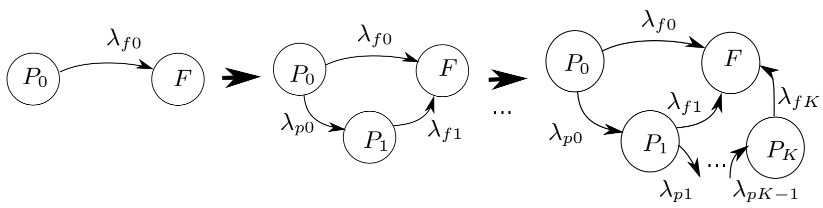


Figure 3.15: Iterative process to build a CTMC model; the preemption effects are added and validated one preemption by another.

them. This is what is called probabilistic schedulability analysis and is formalized as follows.

Deadline Miss. For each job J_{ij}, the probability of deadline miss $P(DM)_{ij}$ is given by one minus the probability that the job CTMC is in the state F at the deadline,

$$P(DM_{ij}) = \mathcal{P_{DM}}(X_{ij}, Q_{ij}) = 1 - P(state = F, time = Deadline) = \int_{d_{ij}}^{\infty} F_{\mathcal{RT}_{ij}} dx. \quad (3.53)$$

The probability of deadline miss of a task is considered to be the maximum of its jobs:

$$P(DM)_i = max_j(P(DM)_{ij}) = max_j\{P(DM)_{ij}\}. \quad (3.54)$$

Response Time. In a similar manner the response time CDF for a job can be obtained. As a reminder, in the probabilistic framework, the response time $\mathcal{RT}$ is a distribution, and can be represented with PDF, or with cumulative probabilities. It is done by checking the probability that the job is finished by some positively increasing time t.

$$\mathcal{RT}_{ij} = \mathcal{P}_{\mathcal{RT}}(X_{ij}, Q_{ij}) = P(\text{state} = F, \text{time} \leq t), \tag{3.55}$$

and that of a task is given as the worst of its jobs:

$$\mathcal{RT}_i = max_j(\mathcal{RT}_{ij}) = \max_j\{F'_{\mathcal{RT}_{ij}}\}. \tag{3.56}$$

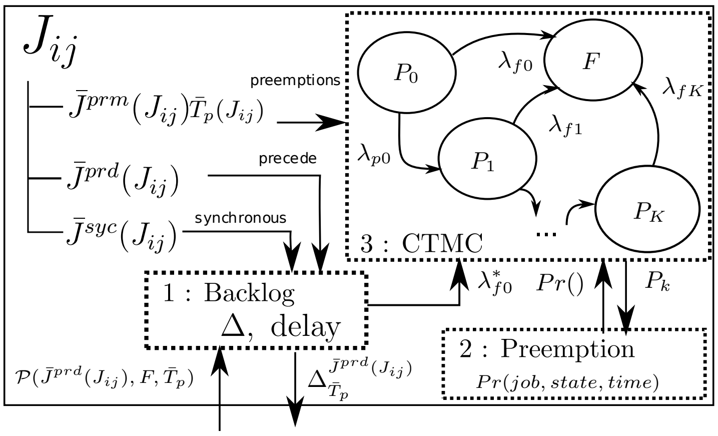


Figure 3.16: Job CMTC model formalization with constituents as blocks and input information.

Figure 3.16 illustrates the three main steps (backlog, CTMC, and preemption) to build CTMC models. There are illustrated the inputs and the interactions between those steps. The CTMC of a job is defined in incremental steps: first the modeling without preemption, then refinement adding preemptions and their effects one by one. Figure 3.15 present the CTMC model for a job which is build incrementally adding one preemption at the time.

3.3.6 Some conclusions on deterministic and probabilistic schedulability analysis

Abstract models for sensitivity analysis. The abstract representations as the C-space, the pC-space, the (α, Δ)-space, and probabilistic (α, Δ)-space, are alternative representations for developing schedulability analysis. They reduce the complexity of analytical scheduling conditions with feasibility regions in which comparison between the resource availability and resource demand can be made graphically with feasibility regions.
The sensitivity analysis, or parametric analysis, applies to the abstract representations in order to explore schedulability conditions and trade-offs between schedulability and resource. The sensitivity analysis is helpful in modeling scheduling conditions and quantifying/qualifying the impact that certain parameters have on task schedulability. The full potential of sensitivity analysis has not yet fully explored, especially with probabilistic models; future work will investigate that by developing strategies for efficient resource changes or parameters changes.
The sensitivity analysis will also prove to simplify complex scheduling problems such as the probabilistic scheduling or the MC scheduling problem. In Chapter 4 there are presented some initial thoughts about probabilistic sensitivity analysis which makes use of the pC-space and the probabilistic (α, Δ)-space. The sensitivity analysis there is used for the MC problem for defining and exploring computational resource-schedulability-criticality levels trade-offs.

Dynamic real-time systems. Adaptivity is a way of guaranteeing timing constraints during mode changes that dynamic real-time embedded systems have. Examples of adaptive resource reservation mechanisms and sensitivity analysis applied to evaluate possible mode changes can be found in [186].
The adaptivity for real-time systems is a topic investigated in collaboration with PhD students Francesco Prosperi, and with Tomasz Kloda, within the RETIS Lab at Scuola Superiore Sant'Anna, and the TIK group at the ETH Zurich. It has been part of projects like R2D2, and CORTEVA.
The research activity with Francesco Prosperi, the RETIS Lab, and the TIK focuses on both real-time systems and networked real-time systems in which changes can happen and for which timing guarantees have to be

offered [155, 167, 168, 169, 171, 185, 186, 187, 196, 199, 205, 206]. In particular, the research activity with Tomasz Kloda [174, 175, 176] proposes important results for time-triggered architectures and the timing definition language. In these works, schedulability tests are proposed under EDF on a single processor for applications that may change their operational modes.
To mention, the work [151] in collaboration with Prof. Giorgio Buttazzo which has applied probabilistic models to adaptive paradigms for soft real-time systems and space applications.
Dynamic real-time systems and adaptivity for real-time systems make the background of techniques and algorithms that will be applied to the MC problem. In particular, with those techniques there will be studied mode changes with mixed critical tasks. Moreover, the sensitivity analysis in [186] will be applied to decide affordable mode changes with the resource available, or what is the required resource to guarantee the worst possible mode changing conditions.

Real-time and probabilistic calculus. With pWCET, discrete or continuous distributions, there is the need to develop probabilistic schedulability analysis. the pCalculus is proposed for defining a version of probabilistic schedulability. It is inspired by the RTC and deterministic bounding functions, and by previous works on probabilistic timing models.
The pCalculus considers multiple bounding functions, each with an associated probability of being a bound to the task or resource behavior. The functions composes into a probabilistic bounding curve which is a distribution of functions; there can be defined probabilistic curves for the resource request and for the resource provisioning. The pCalculus formalizes the composition of such curves as well as schedulability criteria by confronting probabilistic bounds on resource provisioning (service) and resource demand (arrival).
In [188, 189, 197], the pCalculus is formalized; in [172, 173] it been applied to networked embedded systems. The pCalculus has been applied in projects like IRHEDO2 and WIRELESS.

Probabilistic schedulability analysis. The probabilistic schedulability analysis is developed from the pCalculus or probabilistic version of the resource provisioning, resource demand, and workload bounds. Works on probabilistic schedulability analysis have been object of publications on both task scheduling and networking performance guarantees [151, 152, 172, 173, 189, 197, 201, 203]. The project activities related are CORTEVA, CAPHCA, R2D2, and WIRELESS.
A key part of the activity on probabilistic schedulability analysis has been developed at the INRIA Nancy while collaborating with PhD student Dorin Maxim and PhD Liliana Cucu-Grosjean. The works on probabilistic re-sampling of pWCET discrete distributions in order to reduce complexity of probabilistic schedulability analysis [178], or priority assignments with probabilistic tasks [177] are not detailed here.
Probabilistic schedulability analyses are more flexible and less pessimistic than classical deterministic ones, since they can account for the multiple conditions that can happen, and that are parametrized with probabilities. The probabilistic schedulability conditions can cope with both hard and soft real-time.
Unfortunately, the flexibility of probabilities is payed off with an increased complexity. Few approaches have been developed in order to decrease complexity [178, 179, 189]; they all attempt to re-sample pWCET distributions reducing the number of values in those, and in turn reducing the number of functions. The safety of those is guaranted with an increased pessimism. Future work will be developed in order to reduce the complexity, and make the pCaclulus effective for MC.
In [151], another example of probabilistic schedulability analysis applied to soft real-time applications such as the space ones. In there, the composablity of system elements is validated with probabilistic bounds and probabilistic notions of interference. Composability and schedulability are verified with probabilistic utilization analysis.
The complexity also increase due to to the multiple possibilities that probabilities brings. Future works is under discussion in order to reduce the complexity with more efficient representations and computations for discrete distributions. Also formal methods are under investigation with that purpose.

Formal methods for probabilistic real-time. In [152, 201, 203], and with Jasdeep Singh thesis, we are investigating the application of formal methods for characterizing task executions and develop schedulability analysis for probabilistic real-time systems; both continuous and discrete distributions can be applied with formal methods and the system models derived. In those works, the mathematical foundation of formal methods is applied to verify the correctness of both system representations and schedulability analysis.
In `https://forge.onera.fr/projects/probscheduling` there is a tool developed by Jasdeep Singh for schedulability analysis with continuous pWCET distributions with continuous time Markov chains.
In [152, 201, 203], and in the tool developed at the ONERA, the effort has been applying CTMC and stochastic Petri nets to schedulability analysis of real-time systems. Although the use of formal methods makes execution models and scheduler more formal and easier to validate, there still exist problems to be solved. In particular:

- Probabilistic scheduling problems are complex because of the need to take into account for all the cases that probabilities represent. With continuous distributions, the CTMC complexity is mitigated with the use known distribution functions and the mathematics to represent and combine them; with stochastic Petri nets expressiveness improves but the complexity increases because of the need for solving differential equations. With discrete distributions the use of formal methods consist of moving the complexity to the building of task execution models. Hence, at this stage, using formal methods does not significantly reduce overall complexity of probabilistic schedulability analysis. Future contributions are needed in order to significantly reduce complexity and make probabilistic scheduling approach more usable.

- Task execution models from formal methods are still pessimistic. The pessimism comes from the need of bounding distributions with known distribution laws, e.g. the use of exponential distribution at each step of the modeling. Future work has to be devoted to reduce such pessimism: more precise upper bounds will be developed, and constraining hypotheses will be released.

Future work with formal methods and probabilistic scheduling developed with formal methods will show the strong points of formal methods applied to complex scheduling problems. It will also bring formality to the MC problem where the probabilities will allow defining efficient scheduling decisions.

Chapter 4

Current state of mixed criticality research

This chapter details my recent works, since 2015, on MC for real-time embedded systems, i.e., the present research. The papers and projects listed in the previous chapter are the background for these achievements: the tools developed are applied with mixed critical requirements. The current state of mixed criticality research tackles with both timing modeling and schedulability analysis as both contribute to MC. it is also motivate from both professional and academic experience, and from practical and theoretical needs of real-time embedded systems.

There are works for deriving *probabilistic MC task models – MC timing analysis approaches* [159, 192, 193]. Those contributions are on providing more information than the classical ones i.e., Vestal's model in Equation (2.1). They are motivated from the need for having more flexible representations and reduce the pessimism that MC timing models have. Probabilistic approaches to timing analysis are explored because they parameterize by the criticality levels: HI-criticality levels would require high guarantees, possibly deterministic ones; for lower-criticality, probabilistic guarantees would suffice for understanding task executions and system dynamics. Deterministic and probabilistic models can be combined together to represent the multiple tasks executions.

The probabilistic MC models proposed are flexible because they embed multiple WCET thresholds each with the probability associated. They are investigated in order to have new models able to reduce the pessimism of classical deterministic MC models. The pessimism reduces since more precise WCET threshold can be associated to each criticality level demanded.

There are also works on *MC schedulability analysis* which apply deterministic models as well as probabilistic models [165, 166, 184, 190, 191, 194, 198, 201, 202, 203].

The sensitivity analysis for MC scheduling, deterministic or probabilistic, is defined on abstract representations and is motivated by the need for simplifying the MC scheduling problems: the analytical MC scheduling conditions convert into graphical conditions with a schedulability region per criticality modes. The sensitivity analysis is developed also for having an efficient way of exploring trade-offs between schedulability, resource usage, and criticality levels. The trade-offs allow designing the MC system according to the criteria adopted; they allow also developing the best changing strategies possible for the specific execution conditions considered. Some works explore the qualities of sensitivity analysis and define task parameters to be investigated [184, 190, 191, 198]

Deterministic MC schedulability approaches without sensitivity analysis are proposed with the objective of reducing the pessimism of classical MC scheduling hypotheses. A newly developed algorithm come from the demand of better coping with realistic MC hypotheses in which not necessarily all the HI-criticality tasks are in HI-criticality mode at the same time. The scheduling algorithm allows for more efficient resource usage [166].

Probabilistic MC schedulability approaches without sensitivity analysis are proposed in order to apply probabilistic MC models. With those algorithms, it is possible to define scheduling decisions which makes use of probabilities. Accurate scheduling decisions can be are taken from the probabilistic models: they are accurate because they better cope with runtime conditions. With those, it is possible to achieve improved resource usage.

Besides, it is presented a work on probabilistic MC schedulability analysis which develops a MC scheduling algorithms inspired by classical deterministic MC schedulers. In it, probabilities are effectively applied into scheduling decisions [165]. Such work is motivated by the need for using probabilistic models, and by the need for leveraging probabilities into scheduling decisions. In [165], the contribution allows to see and implement what probabilities can do for MC scheduling: they allow to estimate probable changes, and take decisions that better cope with those.

Another set of works on probabilistic MC schedulability analysis [200, 201, 202, 203, 204] is presented; they all make use of formal methods for the MC scheduling problem. Formal methods are studied in order to reduce the complexity of MC schedulability problems. With formal methods, it is possible defining formal representations of task scheduling which are validate during construction. With those, probabilistic task models are applied, probability of mode changes are quantified, and "smart" scheduling strategies are developed. The strategies are for reducing probability of changes to HI-criticality mode, and for reducing the number of LO-criticality jobs to be dropped.

Fault happens within real-time embedded systems; they are tightly couples with execution conditions/criticality levels. For these, faults have to be included into task models as well as into schedulability analysis for MC. It is then proposed a work which embeds fault effects into task models, and that develop MC scheduling analysis with fault models [194]. In it, probabilistic timing models are developed depending on the fault conditions applied, and probabilistic sensitivity analysis is developed in order to evaluate impact of faults on schedulability and criticality levels that can be guaranteed.

The chapter is structured to show first works on deterministic mixed criticality. Among those, are listed the contributions which tackles with the schedulability analysis with and without sensitivity analysis. In particular:

- There is [198] that tackles with deterministic sensitivity analysis and formalize it with classical schedulers such as EDF, for defining resource-criticality trade-offs. Trade-offs are defined to explore MC schedulability conditions, resource usage, and criticality levels;
- There is [166] that releases the pessimistic hypothesis that once a HI-criticality task/job goes in HI-criticality mode, every HI-criticality task/job goes in HI-criticality mode. More realistic hypotheses allows reducing the pessimism.

Then, the probabilistic works on MC, both on timing models and on schedulability analyses are presented:

- [193] which formalizes probabilistic models that account for all the possible execution conditions with probability associated to each WCET threshold. A flexible and fine grained representation reduces pessimism and enhance representation of system execution conditions;
- [184, 191] for the probabilistic sensitivity analysis and how it is formalized for the MC problem. These two works are for investigate the potential of probabilistic abstract representations in reducing complexity of MC scheduling problems;
- [194] which embeds fault effects into both probabilistic timing models (pWCETs) and probabilistic schedulability analysis made with probabilistic sensitivity analysis. The work is for studying fault effects and the robustness of MC scheduling decisions with respect to fault conditions; sensitivity analysis defines fault models, schedulability, and criticality trade-offs
- [165] is a work on probabilistic MC schedulability analysis without the use of sensitivity analysis. It is the first work in which probabilities define scheduling decisions;
- [204] which introduces formal methods to model probabilistic MC scheduling decisions and to develop LO-criticality task/jobs dropping to maximize the resource usage. Formal methods prove to be effective in representing complex scheduling problems like the MC one, while verified scheduling decisions are developed from such representations.

Present contributions are concluded with remarks and possible future developments. In Figure 4.1 there is represented how the tools developed in past research apply into the MC problem; the works listed in the following details such contributions, while the tools applied are those presented in Chapter 3.

The results here presented are from ongoing collaborations with Prof. Zhishan Guo of University of Central Florida, Prof. Laurent George of University of Paris Est, PhD Konstantinos Bletsas of CISTER Porto, PhD Liliana Cucu-Grosjean of INRIA Paris, Prof. Iain Bate of University of York, and Prof. Kecheng Yang of Texas State University.

4.1 Deterministic approaches to mixed criticality

Hereby two works on deterministic MC analysis [166, 198]: [198] that applies sensitivity analysis and tackles with the industry perspective to MC; [166] that apply more classical MC scheduling with less constrained hypotheses.

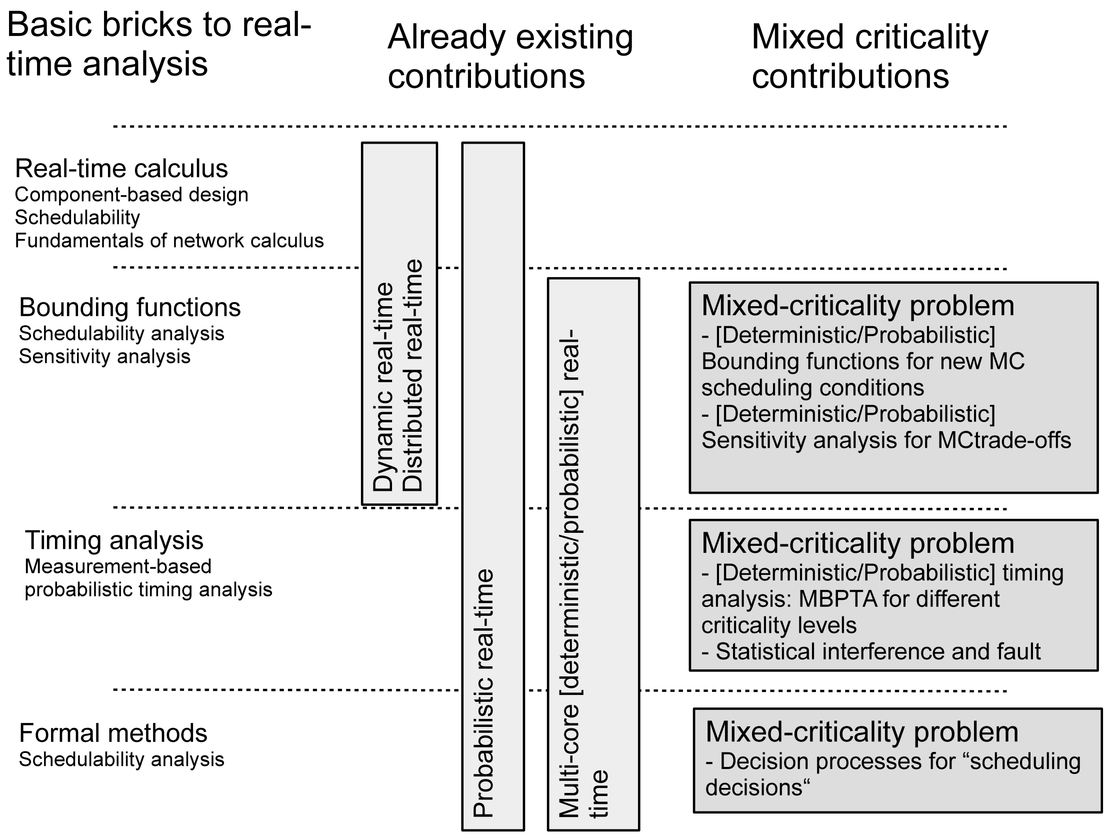

Basic bricks to real-time analysis
Already existing contributions
Mixed criticality contributions
Real-time calculus
Component-based design
Schedulability
Fundamentals of network calculus
Bounding functions
Schedulability analysis
Sensitivity analysis
Timing analysis
Measurement-based probabilistic timing analysis
Formal methods
Schedulability analysis
Dynamic real-time
Distributed real-time
Probabilistic real-time
Multi-core [deterministic/probabilistic] real-time
Mixed-criticality problem
- [Deterministic/Probabilistic] Bounding functions for new MC scheduling conditions
- [Deterministic/Probabilistic] Sensitivity analysis for MCtrade-offs
Mixed-criticality problem
- [Deterministic/Probabilistic] timing analysis: MBPTA for different criticality levels
- Statistical interference and fault
Mixed-criticality problem
- Decision processes for "scheduling decisions"

Figure 4.1: Current state of contribution to MC problem. Listed tools applied, already existing contributions and MC extension of those.

4.1.1 Sensitivity analysis for mixed criticality

In [198], in collaboration with Prof. Zhishan Guo, the deterministic sensitivity analysis is applied to task models and schedulability analysis with MC. This work makes use of deterministic approaches to MC real-time; the MC perspective explored here is more the industrial one, in which partitions and partitions effects are evaluated with the help of sensitivity analysis. The sensitivity analysis applied is for defining and evaluating possible trade-offs.

The MC task modeling is laid out with multiple bounding functions such as resource demand and workload bound functions. Those functions are defined in order to bound task behaviors under the possible execution modes that can happen at runtime. Each bound represents a criticality level as well as an execution mode for the task set. The set of bounding functions is applied to develop the schedulability conditions with MC. Different levels of schedulability are defined from the possible criticality levels for fixed priority and EDF scheduling. Finally, the sensitivity analysis applies to evaluate the impact that MC task behaviors have on schedulability. Trade-offs between schedulability, criticality levels and resource availability are explored. The work here listed is a continuation of [190] which applies more specifically to cyber-physical systems. In both [190, 198] the MC perspective approached is more the industrial one than the academic one.

Bounding with mixed criticality

In MC real-time systems, task parameters such as WCETs depend on criticality levels. Safety critical applications have to be assured against any possible execution condition, faults included. A way to do that is to consider WCETs large enough to account for such conditions. Instead, if the task is mission critical or non-critical, its WCET requirement would be smaller as the task demands less in terms of resource and assurance [54, 140], [165]. The modeling and analysis are restricted to two criticality levels: the high criticality HI and the low criticality LO. Nonetheless, the reasoning can generalize to any criticality level.

HI-criticality tasks i.e., safety critical, can have two execution modes, HI-criticality mode represented with $C_i(\text{HI})$, and LO-criticality mode represented with $C_i(\text{LO})$. $C_i(\text{HI})$ models the HI-criticality behavior of the task τ_i

(most critical), thus the worst possible conditions it can suffer [165]. $C_i(\text{LO})$ models the LO-criticality conditions for τ_i. It does not assure against faults, at least it does not against all of them [165]. It has to be $C_i(\text{HI}) \geq C_i(\text{LO})$ [140].

The model of a HI-criticality task is:

$$\tau_i = ([C_i(\text{LO}), C_i(\text{HI})], T_i, D_i, \chi_i). \tag{4.1}$$

For it, two are the resource requests rbf possible, depending on the criticality mode active: $\mathsf{rbf}^{\text{HI}}_{\text{HI},i} = \max\left\{0, \left(\left\lceil \frac{t}{T_i} \right\rceil \cdot C_i(\text{HI})\right)\right\}$ which models the resource request in HI-criticality mode, and $\mathsf{rbf}^{\text{LO}}_{\text{HI},i} = \max\left\{0, \left(\left\lceil \frac{t}{T_i} \right\rceil \cdot C_i(\text{LO})\right)\right\}$ which models the resource request in LO-criticality mode. χ_i is the task criticality level indicating the task actual criticality mode, $\chi_i \in \{\text{HI}, \text{LO}\}$.

LO-criticality tasks are tasks that can only execute under LO-criticality mode. $C_i(\text{LO})$ is sufficient to model the task behavior. The LO-criticality task model is:

$$\tau_i = (C_i(\text{LO}), T_i, D_i), \tag{4.2}$$

with only the LO-criticality mode possible, and $\mathsf{rbf}_{\text{LO},i} = \max\left\{0, \left(\left\lceil \frac{t}{T_i} \right\rceil \cdot C_i(\text{LO})\right)\right\}$ models the resource request of the LO-criticality task τ_i.

The MC real-time application Γ composes of a HI-criticality part Γ_{HI} which includes all and only the HI-criticality tasks, and a LO-criticality part Γ_{LO} which includes all and only the LO-criticality tasks; $\Gamma = \Gamma_{\text{HI}} \cup \Gamma_{\text{LO}}$.

At runtime there are different possible combinations of tasks executing in their criticality modes. For example, it could exist combinations of only HI-criticality tasks executing in HI-criticality mode. There could also exist combinations where some HI-criticality tasks execute in HI-criticality mode, others executes in LO-criticality mode, and LO-criticality tasks executes as well. It is also possible to have all the HI-criticality task executing in LO-criticality mode together with some or all LO-criticality tasks. Each combination k is a scheduling that can happen at runtime, and has a criticality level associated χ^k resulting from the combination of the criticality mode applied in the scheduling/combination.

The system criticality level χ describes the combination of tasks and task modes at runtime, $\chi \in \{\chi^1, \chi^2, \dots\}$. The purpose of this work is to model all the possible combinations, and apply them into schedulability analysis.

From the MC task modeling, Equation (4.1) and Equation (4.2), multiple bounds can be defined to the task set resource request. They represent execution conditions as combination of tasks and task modes that can happen at runtime.

- $\mathsf{rbf}^{\text{HI}}_{\text{HI}}$ - is the resource request from **all and only** the HI-criticality tasks being in HI-criticality mode: $\mathsf{rbf}^{\text{HI}}_{\text{HI}} = \sum_{\forall \tau_i \in \Gamma_{\text{HI}}} \mathsf{rbf}^{\text{HI}}_{\text{HI},i}$;
- $\mathsf{rbf}^{\text{HI,LO},j}_{\text{HI}}$ - is the resource request from **all** the HI-criticality tasks in HI-criticality mode which are **combined** with LO-criticality tasks: $\mathsf{rbf}^{\text{HI,LO},j}_{\text{HI}} = \sum_{\forall \tau_i \in \Gamma_{\text{HI}}} \mathsf{rbf}^{\text{HI}}_{\text{HI},i} + \sum_{\text{some } \tau_k \in \Gamma_{\text{LO}}} \mathsf{rbf}_{\text{LO},k}$. $\overline{\mathsf{rbf}}^{\text{HI,LO}}_{\text{HI}} = \{\mathsf{rbf}^{\text{HI,LO},j}_{\text{HI}}\}$ is the set of those rbfs with index j specifying which LO-criticality tasks are added to the HI-criticality ones;
- $\mathsf{rbf}^{\text{HI}-\text{LO},r}_{\text{HI}}$ - is the resource request where **some** HI-criticality tasks in HI-criticality mode, the rest in LO-criticality mode, in **combination** with LO-criticality tasks: $\mathsf{rbf}^{\text{HI,LO},r}_{\text{HI}} = \sum_{\text{some } \tau_i \in \Gamma_{\text{HI}}} \mathsf{rbf}^{\text{HI}}_{\text{HI},i} + \sum_{\text{rest } \tau_j \in \Gamma_{\text{HI}}} \mathsf{rbf}^{\text{LO}}_{\text{HI},j} + \sum_{\text{some } \tau_k \in \Gamma_{\text{LO}}} \mathsf{rbf}_{\text{LO},k}$ The whole set of combinations is $\overline{\mathsf{rbf}}^{\text{HI}-\text{LO}}_{\text{HI}} = \{\mathsf{rbf}^{\text{HI}-\text{LO},r}_{\text{HI}}\}$ with r the index to the combinations specifying which LO-criticality tasks are applied and which HI-criticality tasks are in HI-criticality mode;
- $\mathsf{rbf}^{\text{LO},k}_{\text{HI}}$ - is the resource request where **all** the HI-criticality tasks are in LO-criticality mode and the **combination** with LO-criticality tasks: $\mathsf{rbf}^{\text{LO},k}_{\text{HI}} = \sum_{\forall \tau_i \in \Gamma_{\text{HI}}} \mathsf{rbf}^{\text{HI}}_{\text{HI},i} + \sum_{\text{some } \tau_k \in \Gamma_{\text{LO}}} \mathsf{rbf}_{\text{LO},k}$. $\overline{\mathsf{rbf}}^{\text{LO}}_{\text{HI}} = \{\mathsf{rbf}^{\text{LO},k}_{\text{HI}}\}$ is the set of such combinations, with k the index to represent which LO-criticality tasks are added;
- rbf_{LO} - is the resource request from **only** LO-criticality tasks: $\mathsf{rbf}_{\text{LO}} = \sum_{\forall \tau_i \in \Gamma_{\text{LO}}} \mathsf{rbf}_{\text{LO},i}$.

The resource requests of all the combinations between tasks and task modes can be grouped as:

$$\overline{\mathsf{rbf}} \overset{def}{=} \{\mathsf{rbf}^{\text{HI}}_{\text{HI}}, \overline{\mathsf{rbf}}^{\text{HI,LO}}_{\text{HI}}, \overline{\mathsf{rbf}}^{\text{HI}-\text{LO}}_{\text{HI}}, \overline{\mathsf{rbf}}^{\text{LO}}_{\text{HI}}, \mathsf{rbf}_{\text{LO}}\}. \tag{4.3}$$

To each resource request there is a system criticality level χ^k associated, $\chi^k \in \overline{\chi}$.

With the MC model there exist a set of level-i workload bounds, each obtained with the combination of HI- and LO-criticality tasks in their respective modes. Only the tasks higher priority than τ_i are combined for the level-i workload, and χ^k_i is the criticality level associated of the level-i workloads combination. There exist:

- $\mathsf{wbf}_{\text{HI},i}^{\text{HI}}$ - **only** the HI-criticality tasks all in HI-criticality mode; $\chi_{\text{HI},i}^{\text{HI}} = \text{HI}$ represents its criticality level;
- $\overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI,LO}}$ - the HI-criticality tasks **all** in HI-criticality mode **combined** with LO-criticality tasks. To each $\mathsf{wbf}_{\text{HI},i}^{\text{HI,LO},k} \in \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI,LO}}$, $\chi_{\text{HI},i}^{\text{HI,LO},k}$ represents its criticality level;
- $\overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI}-\text{LO}}$ - **some** of the HI-criticality tasks in HI-criticality mode (the rest is in LO-criticality mode) **combined** with LO-criticality tasks. To each $\mathsf{wbf}_{\text{HI},i}^{\text{HI}-\text{LO},j} \in \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI}-\text{LO}}$, $\chi_{\text{HI},i}^{\text{HI}-\text{LO},j}$ represents its criticality level;
- $\overline{\mathsf{wbf}}_{\text{HI},i}^{\text{LO}}$ - **all** the HI-criticality tasks in LO-criticality mode **combined** with LO-criticality modes. To each $\mathsf{wbf}_{\text{HI},i}^{\text{LO},r} \in \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{LO}}$, $\chi_{\text{HI},i}^{\text{LO},r}$ represents its criticality level;
- $\mathsf{wbf}_{\text{LO},i}$ - **only** LO-criticality tasks; $\chi_{\text{LO},i} = \text{LO}$ represents its criticality level.

All the combination of the level-i workload are grouped as:

$$\overline{\mathsf{wbf}}_{\text{HI},i} \overset{def}{=} \{\mathsf{wbf}_{\text{HI},i}^{\text{HI}}, \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI,LO}}, \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI}-\text{LO}}, \overline{\mathsf{wbf}}_{\text{HI},i}^{\text{LO}}, \mathsf{wbf}_{\text{LO},i}\}, \tag{4.4}$$

with the criticality levels for the level-i workload as:

$$\overline{\chi}_i \overset{def}{=} \{\chi_{\text{HI},i}^{\text{HI}}, \overline{\chi}_{\text{HI},i}^{\text{HI,LO}}, \overline{\chi}_{\text{HI},i}^{\text{HI}-\text{LO}}, \overline{\chi}_{\text{HI},i}^{\text{LO}}, \chi_{\text{LO},i}\}. \tag{4.5}$$

The bounds in Equation (4.4) can be ordered in increasing order, $\mathsf{wbf}^j \leq \mathsf{wbf}^{j+1}$; the set $\overline{\chi}_i$ from Equation (4.5) is ordered accordingly and such that χ_i^j is for wbf_i^j and χ_i^{j+1} is for wbf_i^{j+1}.

In case of $\text{HI} - \text{LO}$, instead of enlisting all the combinations it is possible to define 'envelope' bounds depending on the number of HI-criticality tasks that are in HI-criticality mode at the same time: $\mathsf{wbf}_{\text{HI},i}^{*\text{HI}-\text{LO},k} \overset{def}{=} \max_j\{\mathsf{wbf}_{\text{HI},i}^{\text{HI}-\text{LO},j}\}$. k is the number of HI-criticality tasks in HI-criticality mode considered for the combination. $\overline{\mathsf{wbf}}_{\text{HI},i}^{*\text{HI}-\text{LO}}$ collects all those envelopes, for all k, and can be applied into $\overline{\mathsf{wbf}}$ instead of $\overline{\mathsf{wbf}}_{\text{HI},i}^{\text{HI}-\text{LO}}$. This would reduce the number of possible combinations and criticality levels, in turn reducing the complexity of the modeling.

Under EDF, the different combinations are represented with dbfs. There exist: i) $\mathsf{dbf}_{\text{HI}}^{\text{HI}}$ - **only** the HI-criticality tasks all in HI-criticality mode; $\chi_{\text{HI},i}^{\text{HI}} = \text{HI}$ represents its criticality level; ii) $\overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI,LO}}$ - the HI-criticality tasks **all** in HI-criticality mode **combined** with LO-criticality tasks. To each $\mathsf{dbf}_{\text{HI}}^{\text{HI,LO},k} \in \overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI,LO}}$, $\chi_{\text{HI}}^{\text{HI,LO},k}$ associated to, is its criticality level; iii) $\overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI}-\text{LO}}$ - **some** of the HI-criticality tasks in HI-criticality mode (the rest is in LO-criticality mode) **combined** with LO-criticality tasks. To each $\mathsf{dbf}_{\text{HI}}^{\text{HI}-\text{LO},j} \in \overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI}-\text{LO}}$, $\chi_{\text{HI}}^{\text{HI}-\text{LO},j}$ is its criticality level; iv) $\overline{\mathsf{dbf}}_{\text{HI}}^{\text{LO}}$ - **all** the HI-criticality tasks in LO-criticality mode **combined** with LO-criticality modes. To each $\mathsf{dbf}_{\text{HI}}^{\text{LO},r} \in \overline{\mathsf{dbf}}_{\text{HI}}^{\text{LO}}$, $\chi_{\text{HI}}^{\text{LO},r}$ is its criticality level; v) dbf_{LO} - **only** LO-criticality tasks; $\chi_{\text{LO},i} = \text{LO}$ representing its criticality level. The set of all those dbfs is:

$$\overline{\mathsf{dbf}}_{\text{HI}} \overset{def}{=} \{\mathsf{dbf}_{\text{HI}}^{\text{HI}}, \overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI}}, \overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI}-\text{LO}}, \overline{\mathsf{dbf}}_{\text{HI}}^{\text{LO}}, \mathsf{dbf}_{\text{LO}}\}, \tag{4.6}$$

and the set of criticality levels is:

$$\overline{\chi} \overset{def}{=} \{\chi_{\text{HI}}^{\text{HI}}, \overline{\chi}_{\text{HI}}^{\text{HI,LO}}, \overline{\chi}_{\text{HI}}^{\text{HI}-\text{LO}}, \overline{\chi}_{\text{HI}}^{\text{LO}}, \chi_{\text{LO}}\}. \tag{4.7}$$

The bounds in Equation (4.6) can be ordered in increasing order, $\mathsf{dbf}^i \leq \mathsf{dbf}^{i+1}$; the set $\overline{\chi}$ from Equation (4.7) is ordered accordingly and such that χ^j is for dbf^j and χ^{j+1} is for dbf^{j+1}.

In the $\text{HI} - \text{LO}$ case, bounds can be defined bounds such that: $\mathsf{dbf}^{*\text{HI}-\text{LO},k} \overset{def}{=} \max_j\{\mathsf{dbf}^{\text{HI}-\text{LO},j}\}$. k the number of HI-criticality tasks in HI-criticality mode; $\overline{\mathsf{dbf}}_{\text{HI}}^{*\text{HI}-\text{LO}}$ collects them all and can be applied into $\overline{\mathsf{dbf}}$ instead of $\overline{\mathsf{dbf}}_{\text{HI}}^{\text{HI}-\text{LO}}$. This allows reducing the number of possible combinations and criticality levels, in turn reducing the complexity of the modeling.

Figure 4.2 illustrates an example of some level-i bounds $\mathsf{wbf}_i \in \overline{\mathsf{wbf}}$, while Figure 4.3 is an example of some demand bounds $\mathsf{dbf} \in \overline{\mathsf{dbf}}$ from different task mode combinations. For each bound there is associated a criticality level. In the figures there are represented few bounds (3 LO and a HI) which can be compared among them and with the available resource sbf. It is: $\mathsf{wbf}_{\text{LO},i}^{\text{HI},r} \leq \mathsf{wbf}_{\text{LO},i}^{\text{HI},t} \leq \mathsf{wbf}_{\text{HI},i}^{\text{HI},k} \leq \mathsf{wbf}_{\text{HI},i}^{\text{LO},s}$ and $\mathsf{dbf}_{\text{LO}}^{\text{HI},m} \leq \mathsf{dbf}_{\text{LO}}^{\text{HI},n} \leq \mathsf{dbf}_{\text{HI}}^{\text{HI},j} \leq \mathsf{dbf}_{\text{HI}}^{\text{LO},v}$.

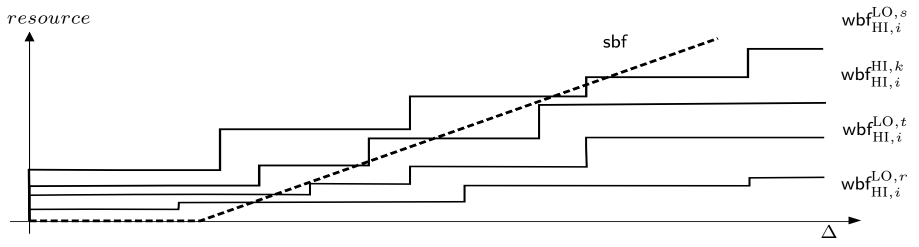


Figure 4.2: level-i wbfs from MC executions under FP. Some wbf_is compared with the resource provisioning sbf.

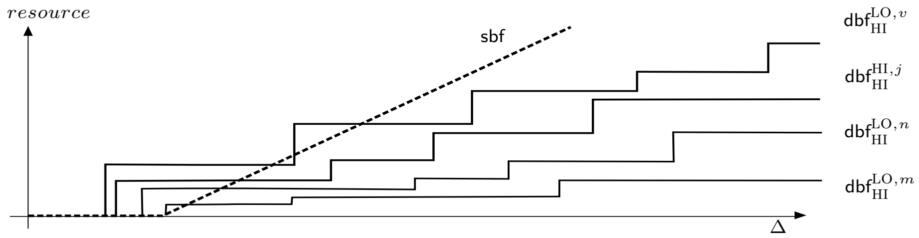


Figure 4.3: dbfs from MC executions under EDF. Some dbfs compared with the resource provisioning sbf.

Scheduling with mixed criticality

There are proposed two schedulability analyses based on FP and EDF that apply MC tasks Equation (4.1) and Equation (4.2). They are off-line analysis which account for all the criticality mode combinations that can happen at runtime. They embeds criticality levels into schedulability conditions.

These analyses focus on finding which are the criticality levels (criticality mode combinations) that can be assured schedulable. They also allow for evaluating the resource applied to execute Γ_{HI}, and thus the remaining resource is left to execute Γ_{LO} without harming HI-criticality tasks' executions.

To guarantee schedulability, the resource provisioning sbf is compared with resource requests (workloads) in case of FP, or with the resource demand in case for EDF. Figure 4.2 and Figure 4.3 illustrate the comparison between bounds and some available sbf. The way to compare depends on the scheduling policy.

Theorem 4.1.1 (FP schedulability with MC) *Considering a mixed criticality task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ *of* n *tasks ordered with decreasing priority i.e.,* τ_1 *is assigned the highest priority whereas* τ_n *is assigned the lowest priority.* Γ *composes of* HI*-criticality tasks* Γ_{HI} *defined as in Equation (4.1), and* LO*-criticality tasks* Γ_{LO} *defined as in Equation (4.2).* $\forall \tau_i \in \Gamma$, $hp(i) = \{\tau_j, \tau_k, \ldots, \tau_i\}$ *is the set of tasks with priority higher or equal to* τ_i*; tasks in* $hp(i)$ *belongs to* Γ_{HI} *and* Γ_{LO}*. The level-i workloads are:* $\overline{wbf_i} = \{wbf_{\mathrm{HI},i}^{\mathrm{HI}}, \overline{wbf_{\mathrm{HI},i}}^{\mathrm{HI,LO}}, \overline{wbf_{\mathrm{HI},i}}^{\mathrm{HI-LO}}, \overline{wbf_{\mathrm{HI},i}}^{\mathrm{LO}}, wbf_{\mathrm{LO},i}\}$*, according to Equation (4.4), and* $\overline{\chi}_i$ *defines the set of criticality levels for the level-i workloads Equation (4.5).* Γ *is FP schedulable under resource provisioning sbf with system criticality level* $\chi = min_i\{\chi_i\}$*,* χ_i *being the level-i criticality level in* $\overline{\chi}_i$*, if for all* $i \in \{1, 2, \ldots, n\}$ $\exists t_0 \in \mathrm{schedP}_i$ *such that:*

$$wbf_i^{\chi_i}(t_0) \leq sbf(t_0); \tag{4.8}$$

schedP_i *is the set of deadlines of all* $\tau_j \in hp(i)$*.*

Equation (4.8) in Theorem 4.1.1 defines the FP schedulability conditions which apply MC models. It proposes different degree of schedulability for MC tasks.

Theorem 4.1.2 (EDF schedulability with MC) *Considering a mixed criticality task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ *with* HI*-criticality tasks* Γ_{HI} *defined as in Equation (4.1) and* LO*-criticality tasks* Γ_{LO} *defined as in Equation (4.2),* $\Gamma = \Gamma_{\mathrm{HI}} \cup \Gamma_{\mathrm{LO}}$*. For* Γ*, the ordered demand bound functions are* $\overline{dbf} = \{dbf_{\mathrm{HI}}^{\mathrm{HI}}, \overline{dbf_{\mathrm{HI}}}^{\mathrm{HI,LO}}, \overline{dbf_{\mathrm{HI}}}^{\mathrm{HI-LO}}, \overline{dbf_{\mathrm{HI}}}^{\mathrm{LO}}, dbf_{\mathrm{LO}})$*, Equation (4.6), with* $\overline{\chi}$ *defining the ordered set of levels of criticality Equation (4.7).*
Γ *is EDF schedulable under resource provisioning sbf with system criticality level* χ *if* $\forall t_0 \in D$*:*

$$dbf^{\chi}(t_0) \leq sbf(t_0); \tag{4.9}$$

$\chi \in \overline{\chi}$ *and* $dbf^{\chi} \in \overline{dbf}_{\mathrm{HI}}$*.*

Equation (4.9) in Theorem 4.1.2 defines EDF schedulability conditions which apply the MC models. It proposes different degree of schedulability for MC tasks.

The proof of previous theorems can be found in [198].

The FP scheduling condition parametrized with χ_i, Equation (4.8), translates into comparing feasibility regions and points within the (α, Δ)-space. A feasibility region Φ^{χ_i} is defined such that: $\Delta \leq \min_i \max_{t \in SchedP_i} \left\{ t - \frac{\mathsf{wbf}_i^{\chi_j}(t)}{\alpha} \right\}$. It has associated the criticality level χ_i such that all the wbf_i, for all i applied, are from the same criticality level χ_i, Theorem 4.1.1.

For EDF it is the same with the scheduling condition in Equation (4.9). A feasibility region Φ^{χ} is defined such that: $\Delta \leq \min_{t \in \mathrm{D}} \left\{ t - \frac{\mathsf{dbf}^{\chi}(t)}{\alpha} \right\}$, and is parametrized with χ.

To both FP and EDF, the set $\overline{\Phi}$ of feasibility regions can be computed for the possible criticality levels resulting from the mode combinations – one set per scheduling policy. It is:

$$\overline{\Phi} \overset{def}{=} \{\Phi_{\mathrm{HI}}^{\mathrm{HI}}, \overline{\Phi}_{\mathrm{HI}}^{\mathrm{HI,LO}}, \overline{\Phi}_{\mathrm{HI}}^{\mathrm{HI-LO}}, \overline{\Phi}_{\mathrm{HI}}^{\mathrm{LO}}, \Phi_{\mathrm{LO}}\}, \tag{4.10}$$

with the criticality level associated:

$$\overline{\chi} \overset{def}{=} \{\chi_{\mathrm{HI}}^{\mathrm{HI}}, \overline{\chi}_{\mathrm{HI}}^{\mathrm{HI,LO}}, \overline{\chi}_{\mathrm{HI}}^{\mathrm{HI-LO}}, \overline{\chi}_{\mathrm{HI}}^{\mathrm{LO}}, \chi_{\mathrm{LO}}\}. \tag{4.11}$$

$\overline{\Phi}$ can be made of envelope bounds, with $\overline{\Phi}_{\mathrm{HI}}^{*\mathrm{HI-LO}}$ instead of $\overline{\Phi}_{\mathrm{HI}}^{\mathrm{HI-LO}}$. The feasibility regions in Equation (4.10) can be ordered in increasing order (from the small region to the larger), with the consequent ordering of the criticality levels in Equation (4.11).

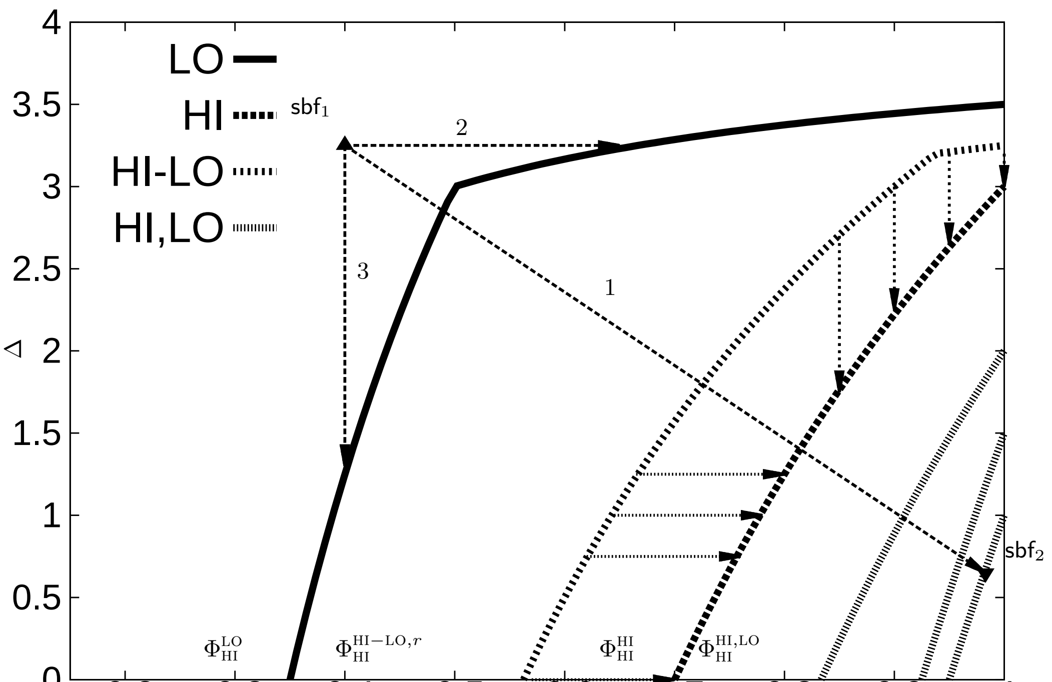


Figure 4.4: FP feasibility regions for an example application Γ.

Figure 4.4 details some feasibility regions for FP scheduling with mixed criticality. Here it is possible comparing regions between them (*ordering between regions* $\Phi^k \leq \Phi^j$), and compare each region with the available resource sbf (*sbf is inside* Φ^j *thus* χ^j *is guaranteed*). LO-criticality conditions (LO) are more prone to schedulability since they require less computational resource – larger feasibility region. The more HI-criticality tasks are scheduled in HI-criticality mode or the more LO-criticality tasks are scheduled together with HI-criticality tasks, the larger is the resource required to schedule – smaller feasibility regions.

It is intended to use sensitivity analysis to investigate multiple elements which can impact the design of MC real-time systems. In particular, sensitivity analysis is applied with schedulability conditions parametrized with criticality levels, Theorem 4.1.1 and Theorem 4.1.2. The proposal is illustrated with three questions.

Q1) *Which is the criticality level that can be assured with the available resource provisioning?* This is a critical question for MC scheduling as it focus on how enhancing computational resource usage by scheduling both HI-criticality and LO-criticality tasks together. With the (α, Δ)-space representation, the sensitivity analysis answers Q1 finding the combinations that can be scheduled for a given resource. Considering the (k, m) formalization, k LO-criticality tasks out of m total task executing, Q1 becomes seeking how many LO-criticality tasks can be executed together with $m-k$ HI-criticality task in HI-criticality modes. k is the parameter to be studied in order to find the largest value that can be guaranteed with the available resource. With the MC modeling proposed in combination with the (α, Δ) representation, this can be solved seeking for the largest feasibility region that include the sbf given. It is exploring an index in $\overline{\Phi}$ seeking for regions.
Q2) *What is the cost to guarantee schedulable a certain criticality level?* The cost being in terms of computational resource. The sensitivity analysis can be used to define what is the resource change necessary to guarantee the schedulability up to a specific criticality level. This is very helpful in defining and evaluating trade-offs between resource and criticality/schedulability. The Euclidean distance $dist(\mathsf{sbf}_2, \mathsf{sbf}_1)$ between two points in the (α, Δ)-space, defined as:

$$dist(\mathsf{sbf}_2, \mathsf{sbf}_1) \overset{def}{=} (\delta\alpha = \alpha_2 - \alpha_1, \delta\Delta = \Delta_2 - \Delta_1), \tag{4.12}$$

quantifies the distance between two resource provisioning $\mathsf{sbf}_2 - \mathsf{sbf}_1 = (\alpha_2 - \alpha_1, \Delta_2 - \Delta_1)$. The cost here is the resource provisioning change necessary to move from sbf_1 to sbf_2. To note that in order to increase the resource provisioning, α has to increase and Δ has to decrease. There exist also the distance between a point and a feasibility region, $dist(\mathsf{sbf}_1, \Phi^j)$. It is defined as:

$$dist(\Phi^j, \mathsf{sbf}_k) \overset{def}{=} (\pm_{\geq 0/<0} min|\delta\alpha|, \pm_{\geq 0/<0} min|\delta\Delta|). \tag{4.13}$$

Metric (4.13) quantifies the resource change to guarantee schedulable the configuration represented by Φ^j. With all positive δs, the sign of the minimum between the absolute values $|\cdot|$ is $+$; with all negative δs, the sign is $-$. αs and Δs for Φ^j are taken from the region border, and the min is for the δs between sbf^k and all those points.
Q3) *What is the cost to change a system criticality mode?* With this, it is intended the possibility in the (α, Δ)-space to quantify the computational resource difference between two criticality levels. The sensitivity analysis quantifies that difference as distance between the two regions which is defined as:

$$dist(\Phi^k, \Phi^j) \overset{def}{=} (\pm_{\geq 0/<0} min|\delta\alpha|, \pm_{\geq 0/<0} min|\delta\Delta|). \tag{4.14}$$

Metric (4.14) is applied at iso-parameter, which means computing the $\delta\Delta$ with the same α, and $\delta\alpha$ with the same Δ. With all positive δs, the sign of the minimum between the absolute values $|\cdot|$ is $+$; with all negative δs, the sign is $-$; with both negative and positive δs, the min is 0 as the intersection between the regions. The αs and the Δs are taken from the regions border. Metric (4.13) and Metric (4.14) are computed differently to signal the resource difference that exist between the two cases.

Figure 4.4 is an example of sensitivity analysis for evaluating the resource necessary to guarantee schedulability – Q1 and Q2 with Metric (4.12) and Metric (4.13). There are 6 regions grouped in 4 different classes: LO for only LO-criticality modes combined, HI for only HI-criticality modes combined, $\textsc{hi} - \textsc{lo}$ for some HI-criticality tasks in HI-criticality modes combined with some LO-criticality tasks, $\textsc{hi}, \textsc{lo}$ for all HI-criticality tasks in HI-criticality mode combined with LO-criticality tasks. There are three cases which define three resource provisioning changes from an initial resource sbf_1. With sbf_1 available is not possible guarantee any of the schedulability level represented, since sbf_1 in not included in any of those feasibility regions.
Change 1: change of sbf_1 to sbf_2 to guarantee schedulability of $\textsc{hi}, \textsc{lo}$, the most demanding case among the represented ones. $dist(\mathsf{sbf}_2, \mathsf{sbf}_1) = (\delta\alpha = 0.985 - 0.4 = 0.585, \delta\Delta = 0.65 - 3.25 = -2.6)$, Metric (4.12). Change 2 iso-Δ: change of sbf_1 to LO configuration schedulability by modifying only α Metric (4.13), $dist(\Phi_{\textsc{lo}}, \mathsf{sbf}_1) = (\delta\alpha = 0.65 - 0.4 = 0.25, \delta\Delta = 0)$. Change 3 iso-$\alpha$: change of sbf_1 to LO configuration schedulability by modifying only Δ Metric (4.13), $dist(\Phi_{\textsc{lo}}, \mathsf{sbf}_1) = (\delta\alpha = 0, \delta\Delta = 1.3 - 3.25 = -1.95)$. The difference between change 2 and change 3 is in terms of changing either α's or Δ's. Distances where one dimension is 0 are advantaged, since the resource change necessary is easier to apply – less constraints.

Figure 4.4 presents also an example of cost evaluation with sensitivity analysis, Q3 and Metric (4.14). From HI to one $\textsc{hi} - \textsc{lo}$ configuration k, it is: $dist(\Phi^{\textsc{hi}}, \Phi^{\textsc{hi}-\textsc{lo},k}) = (0.17, -0.2)$. It quantifies the resource difference between $\Phi^{\textsc{hi}}$ and $\Phi^{\textsc{hi}-\textsc{lo},k}$, equivalently the resource increase necessary to schedule $\Phi^{\textsc{hi}-\textsc{lo},k}$ from a schedulable $\Phi^{\textsc{hi}}$. This translates also into the resource necessary to add LO-criticality tasks into the scheduling.

4.1.2 Mixed criticality scheduling with limited HI-criticality behaviors

In the following, some details and results from [166]. In this work, in collaboration with Prof. Zhishan Guo and Prof. Kecheng Yang, it is proposed a new MC system model able to cope with more realistic assumptions for real-time embedded systems. This work is about applying deterministic approaches to MC.

The proposed model is more general than the existing well-studied Vestal's model in the sense that it allows a system designer to specify the number of HI-criticality tasks that can exceed their LO-WCET simultaneously. It is analyzed how this additional specification could impact the schedulability and develop an MC scheduler for this new model. Schedulability experiments are conducted to compare the results from the scheduler proposed and from a classical MC scheduler, namely EDF-VD. The advantages from having only subsets of HI-criticality tasks exceeding their LO-WCET thresholds simultaneously are validated by these experimental results.

Limited HI-Criticality Behaviors. In some systems, it could be reasonable to assume that only a limited number N of HI-criticality tasks that may exceed their LO-WCET and reach their HI-WCET simultaneously, where $N \leq n_{\text{HI}}$. In contrast, existing MC analysis usually makes the most pessimistic assumption that all of the n_{HI} HI-criticality tasks may execute beyond their LO-WCET and reach its HI-WCET simultaneously. Even if this could actually happen, it can also be viewed as a special case ($N = n_{\text{HI}}$) under the new MC model proposed in this paper. By saying simultaneously (or "at the same time"), it is intended within any time window of length $\overline{T} = \max_i\{T_i\}$ [1]. That is, at most N HI-criticality tasks can require an execution time larger than their $c_i(\text{LO})$ within any time window of length $\overline{T}$. Again, please note that the Vestal's model is a special case of the model, by assigning $N = n_{\text{HI}}$.

Determine N. In [166], it is generally assumed that the parameter N is a parameter given offline, instead of to be determined online by the scheduler. That is, how to determine N is not the focus of this paper, and the focus is on the problem of how to schedule the tasks with a valid schedulability test when N is given as an input parameter. Nonetheless, for the sake of inspiring future work, it is also briefly discuss a couple of potential sources for where the N parameter could come from.

First, it could come from physical constraints in the systems. Different set of HI-criticality tasks may be triggered to perform their HI-criticality behaviors by different physical measurements. Such difference may be significant enough so that they cannot have simultaneous impacts on the system.

Second, it could come from contradicting logic control flows in the code. When the code of tasks has branches, which branch is chosen to execute may depend on some global variables. Different task might have the same global variables in their code, and the same global variables control the branch choices in multiple tasks. As a result, it could be logically impossible for some HI-criticality tasks to take their worst branch choices simultaneously. That is, they cannot have their HI-criticality behaviors to have simultaneous impacts on the system.

Third, it could also come from probabilistic analysis if the WCETs of HI-criticality tasks are independent [42]. In this approach, the probability of multiple HI-criticality tasks performing HI-criticality behaviors could be calculated as a product of multiple (hopefully small) probabilities for each individual task to perform its HI-criticality behavior. When this product is sufficiently small, the simultaneous HI-criticality behaviors of these tasks could be probabilistically deemed impossible.[2] This setting was also considered in [165], which more focuses on the various detailed combinations of tasks that may not perform their HI-criticality behaviors. Therefore, a somewhat complicated scheduling approach was studied there. In this paper, the focus is on the maximum number of such tasks only, and therefore enable the applicability of the relatively simple scheduler, EDF-VD.

It is revised a commonly used and adapted MC scheduler, namely EDF-VD [22], which was proposed for the classical Vestal's model. There are are defined the original analysis of EDF-VD to cope with less pessimistic assumptions, and derive a new schedulability test for EDF-VD under the new model proposed in this paper.

EDF-VD. Similar to the classical EDF scheduler, EDF-VD is a deadline-based, dynamic-priority scheduler. In contrast to EDF, EDF-VD assigns virtual deadlines, which are earlier than the actual deadlines, to HI-criticality jobs. In the runtime, their priorities are determined by their virtual deadlines in the LO-criticality mode; upon a mode switch, their priorities are changed back to their actual deadlines in the HI-criticality mode. Intuitively, the virtual deadlines in the LO-criticality mode provide the room for the HI-criticality tasks to still meet their actual deadlines in the HI-criticality mode, when they occasionally overrun their LO-WCETs.

Let τ denote the MC implicit-deadline sporadic task system that is to be scheduled on a preemptive single-processor. Prior to runtime, EDF-VD performs a schedulability test to determine whether τ can be correctly

[1] When considering a certain time window of length $\overline{T}$, any task τ_i with a partially overlapping scheduling window that experience HI-criticality behavior counts (although it may be already finished by the beginning of the period of interest, or it did not start executing by the end of the period of interest).

[2] Or equivalently, even if it does happen, it is viewed as erroneous, and the system design does not take care of it.

scheduled by it or not. If τ is deemed schedulable, then an additional parameter x is computed for setting virtual deadlines to HI-criticality tasks. Each virtual relative deadline T'_i can be calculated by "shrinking" the actual relative deadline T_i by the scaling factor x.

Next, it is described a schedulability test for EDF-VD under the proposed new model and prove its correctness. Note that, when $N = n_{\text{HI}}$, this schedulability test reduces to the one for the classical Vestal's model in [22].

Schedulability test. First, given an MC task instance, the parameter x is calculated as follows:

$$x \leftarrow \frac{U_{\text{HI}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}}. \tag{4.15}$$

By Theorem 4.1.3 (to be presented later), this assignment of x will be able to guarantee the schedulability under LO-criticality mode.

Then, the schedulability under HI-criticality mode can also be guaranteed if the following inequality holds:

$$xU_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i \leq 1. \tag{4.16}$$

That is, given an MC task instance, the schedulability test needs to check whether Inequality (4.16) is satisfied.

The schedulability test returns success if Inequality (4.16) is satisfied, and failure otherwise.

Upon success, EDF-VD assigns virtual deadline parameters for all HI-criticality tasks as follows:

$$T'_i \leftarrow x \cdot T_i. \tag{4.17}$$

Correctness proof. The correctness proof of the above schedulability test contains two parts: (i) all deadlines being met under LO-mode (Theorem 4.1.3) and (ii) HI-criticality deadlines under HI-mode (Theorem 4.1.5).

Theorem 4.1.3 *Under EDF-VD, all tasks meet their deadlines in* LO*-mode (where all jobs complete upon receiving execution time up to their* LO*-WCETs) if*

$$x \geq \frac{U_{\text{HI}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}}. \tag{4.18}$$

Lemma 4.1.4 *For any period of length t, total demand by* HI*-criticality tasks can not exceed* $(U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i)t$.

Theorem 4.1.5 *Under EDF-VD, all* HI*-criticality tasks meet their deadlines in* HI*-mode if Inequality* (4.16) *holds. In the* HI*-mode, some but no more than N* HI*-criticality job(s) have not completed upon receiving execution time up to their* LO*-WCETs but will complete upon receiving execution time up to their* HI*-WCETs.*

Theorems and lemma proofs are available in [166].

Runtime behavior. During runtime, if a LO-criticality job of task τ_i arrives at time-instant t_a, then the priority of this job is determined by its deadline $t_a + T_i$, whereas its priority will be determined by its virtual deadline $t_a + T'_i$ if it is a HI-criticality job. If any HI-criticality job executes for a duration exceeding its LO-WCET without signaling completion, the scheduler immediately discards all LO-criticality jobs[1] and executes HI-criticality HI-criticality tasks according to EDF order with their actual (instead of virtual) deadlines. Moreover, idleness always serves as the trigger to LO-criticality mode of the system.

Additional discussions. Under the MC scheduling approach, LO-criticality jobs will be dropped in the HI-criticality mode, and any HI-criticality job overrunning its LO-WCET will trigger the mode switch. With the proposed model, this dropping may not be necessary. The following inequality should be examined before the system starts any execution:

$$U_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i \leq 1. \tag{4.19}$$

If Inequality (4.19) is true, then actually no mode switch nor virtual deadline is needed. The system can be scheduled by ordinary preemptive EDF scheduler and all deadlines will be met. This result directly follows from Lemma 4.1.4. If Inequality (4.19) is false, it is then applied the MC scheduling techniques described earlier in this section, and examine Inequality (4.16) to verify the schedulability.

Case studies and experimental results are available in [166].

[1]An efficient implementation of such a runtime dispatcher may be obtained using the technique described in [22, Sec. V-A], to have runtime that is logarithmic in the number of tasks.

4.1.3 Some conclusions on deterministic approaches to mixed criticality

Sensitivity analysis for mixed criticality. In [190, 198] is formalized the deterministic sensitivity analysis through which explore trade-off between computational resource, criticality levels, and schedulability. The effect of changes on resource availability or on different modes are quantified with the abstract representation of the deterministic (α, Δ)-space.
In those papers there are presented multiple case studies from ONERA projects, e.g. ROSACE application `http://sites.onera.fr/schedmcore/ROSACE` and
`https://svn.onera.fr/schedmcore/branches/schedmcore-RTAS2014/Case_Study_RTAS/`, which is a multi-periodic controller executing on a multi/many-core target. In those, the deterministic sensitivity analysis applies at design and development time. The MC perspective adopted is more the industrial one, with no mode changes but just possible modes and mode combinations. Task sets are partitioned and for each partition, classical EDF or FP schedulers are applied.
Future works will develop strategies for optimal or sub-optimal changes to resource or system parameter. Also, future contributions will extend sensitivity analysis to probabilistic MC task models.

Mixed criticality scheduling with limited HI-criticality behaviors. Future development on deterministic MC is necessary to overcome the pessimism of the classical Vestal's model which states that if one HI-criticality task goes in HI-criticality mode, all the HI-criticality tasks go in HI-criticality mode. This assumption is pessimistic and not necessarily representing real MC application; relaxing the assumption will continue to be investigated in order to achieve more efficient resource utilization under mixed criticality. As possible approaches, there are those that explore (k, n) definition of system HI-criticality mode, or those that apply the notion of graceful degradation.
The collaboration with Prof. Zhishan Guo and Prof. Kecheng Yang will continue developing more efficient deterministic MC scheduling algorithms, but also to reduce the complexity of MC scheduling with more realistic hypotheses.

4.2 Probabilistic approaches to mixed criticality

In this section there are presented the probabilistic approaches for the MC problems. First the timing models, then the schedulability analyses are detailed.

4.2.1 On the criticality of pWCET models

The work in [193] proposes a probabilistic representation framework for real-time tasks which composes of multiple probabilistic worst-case models, each estimating the worst-case of a specific possible execution condition that both tasks and the system can encounter.

The probabilistic task model developed can be applied to the MC problem, since probabilities and multiple pWCETs can characterize the criticality modes as different task conditions as well as different confidences. Besides, the probabilistic representation can be used for characterize the effects that faults have on the tasks executions, proving to be flexible and safe.

Worst-case bounding

The *absolute* pWCET $\mathcal{C}_i$ is the worst-case distribution that upper bounds every task execution time obtained under any possible execution scenario $s^j \in S$. The absolute pWCET $\mathcal{C}_i$ is safe if it upper bounds every task execution time under any execution scenario.

Given $s^j \in S$, the pWCET $\mathcal{C}_i^{s^j}$ comes from the measurements taken under s^j and the EVT applied to them (Figure 3.4). $\mathcal{C}_i^{s^j}$ is the pWCET specific to s^j, the *relative* pWCET; the relative pWCET $\mathcal{C}_i^{s^j}$ is safe if it upper bounds any task execution time under s^j.

Measurement representativity is a fundamental requirement for guaranteeing both absolute and relative pWCETs. Hereby the focus is on representativity as the measurement capability of well characterizing multiple execution conditions (worst-cases included) like in [2, 113]. Differently than those works, it is not considered artificially randomized systems that aim at increasing the chances of measuring the worst-case. The input representativity can be built from an enhanced knowledge of the system and of its scenarios, thus *from a study of the system, its S and the coverage of the execution conditions.*

From the partial ordering between pWCETs [48], it is possible defining a notion of *dominance* for scenarios. With respect to task τ_i, given s^r and s^t from S, s^r dominates s^t if and only if $\mathcal{C}_i^{s^r}$ is greater than or equal to $\mathcal{C}_i^{s^t}$, $\mathcal{C}_i^{s^r} \succeq \mathcal{C}_i^{s^t}$, $\succeq$ being the "greater than or equal to" operator which defines the partial ordering between distributions [48].

It is also possible defining the notion of *equivalence* between scenarios. Given s^k and s^j from S, with respect to task τ_i s^k is equivalent to s^j if and only if there exist values in the support of $\mathcal{C}_i^{s^k}$ and $\mathcal{C}_i^{s^j}$ for which $\mathcal{C}_i^{s^k} \succeq \mathcal{C}_i^{s^j}$ and there exist other values in the support of $\mathcal{C}_i^{s^k}$ and $\mathcal{C}_i^{s'^j}$ for which $\mathcal{C}_i^{s^j} \succeq \mathcal{C}_i^{s^k}$.

For a set of equivalent scenarios $S^j = \{s^j, s^k, \dots, s^t\} \subseteq S$ ($s^k, \dots, s^t$ equivalent to s^j), it is possible defining the scenario s^{j*} that dominates all the scenarios in S^j. It would be such that $\mathcal{C}_i^{s^{j*}} \overset{def}{=} max_{s^j \in S^j}\{\mathcal{C}_i^{s^j}\}$, while with the ICDF representation, it would be $F'_{\mathcal{C}_i^{s^{j*}}}(C) \overset{def}{=} max_{s^j \in S^j}\{F'_{\mathcal{C}_i^{s^j}}(C)\}$. s^{j*} is not a real scenario, but it dominates all the $s^k \in S^j$.

Worst-case set. The *Worst-Case Set* task representation is the collection of all the pWCET from S; $\overline{\mathcal{C}}_i$ is the Worst-Case Set representation as a set of pWCET estimates such that:

$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^1}, \mathcal{C}_i^{s^2}, \dots, \mathcal{C}_i^{s^n}). \tag{4.20}$$

With partial ordering between relative pWCETs it is possible ordering scenarios and get $S = \{s^1, s^2, s^3, \dots, s^k\}$ such that $\mathcal{C}^{s^k} \succeq \mathcal{C}^{s^{k-1}} \succeq \dots \succeq \mathcal{C}^{s^1}$; the Worst-Case Set becomes:

$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^1}, \mathcal{C}_i^{s^2}, \dots, \mathcal{C}_i^{s^k}), \tag{4.21}$$

with s^k the worst-case scenario for τ_i, $s^{worst} \equiv s^k$.

Although, the focus on MBPTA, the Worst-Case Set representation applies to both MBPTA and SPTA with multiple execution conditions possible.

Worst-case set and dominance

Although with actual real-time systems it is reasonable to assume a finite number of measurement scenarios, enumerating them all remains a complex problem. With dominance between scenarios, it would be possible neglecting the dominated scenarios in order to ease the task representation from Equation (4.20) and Equation (4.21). Moreover, with equivalence between scenarios it would be possible to assume the correspondent dominating scenario $s^{*,j}$ to represent all the equivalent scenarios S^j.

From Equation (4.20) and Equation (4.21), fewer dominating scenarios S^* could be considered to represent the task execution behavior. $S^* = \{s^r, s^j, s^k\} \subseteq S$ is such that s^r dominates some scenarios in S, s^j dominates other scenarios as well as s^r and s^k dominates all the scenarios. The Worst-Case Set becomes:

$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^r}, \mathcal{C}_i^{s^j}, \mathcal{C}_i^{s^k}), \tag{4.22}$$

as a less complex probabilistic representation to the task executions; Equation (4.22) remains a safe representation for the task behavior since the worst-cases s^k and $\mathcal{C}_i^{s^k}$ are included.

Figure 4.8 depicts $S = \{s^1, s^2, \dots, s^k \equiv s^{worst}\}$, each scenario s^j with a trace of execution time measurements; the resulting pWCETs $\mathcal{C}_i^{s^k}$ are illustrated with the partial ordering guaranteed by $\succeq$.

At this stage, S is assumed to be known; future work will investigate how to obtain the different scenarios and how to guarantee the existence of worst-cases among them.

Combining the orthogonal information of WCET thresholds (with probability associated) and execution scenarios from the Worst-Case Set, Equation (4.20), Equation (4.21) or Equation (4.22), *inter-scenario* and *intra-scenario* representations exist.

Inter-scenario representation. The inter-scenario representation characterizes task behavior across scenarios. Given an exceeding probability p and the WCET threshold at that exceeding probability for each scenario $s^j \in S$ (equivalently $s^j \in S^*$), it is $\langle \overline{C}_i, p \rangle$ such that:

$$\overline{C}_i \overset{def}{=} (C_i^{s^1}, C_i^{s^2}, \dots, C_i^{s^k}) \tag{4.23}$$

is the set of WCET thresholds such that $\langle C_i^j, p \rangle$ $\forall s^j \in S$. As an example, it is possible picking $p = 10^{-9}$ with $\langle C_i^j, 10^{-9} \rangle$ $\forall s^j \in S$. Equation (4.23) is the inter-scenario representation for the task worst-case execution time.

Intra-scenario representation. The intra-scenario representation describes the task behavior focusing on a specific scenario. For a given scenario $s^j \in S$ (equivalently $s^j \in S^*$) and a set of exceeding thresholds probabilities $(p_1, p_2, \dots p_m)$ it is:

$$\hat{C}_i^{s^j} \overset{def}{=} (\langle C_{1,i}^{s^j}, p_1\rangle, \langle C_{2,i}^{s^j}, p_2\rangle, \dots, \langle C_{m,i}^{s^j}, p_m\rangle). \tag{4.24}$$

Equation (4.24) is the intra-scenario representation for the task worst-case execution time on a specific scenario with all the meaningful WCET thresholds and exceeding probabilities.

Inter- and intra-scenario representations will be proven to be handy for schedulability and sensitivity analysis with future work.

Each pWCET estimations composing the Worst-Case Set representation implicitly carries confidence (as safety) of being the absolute task pWCET; execution scenarios may be more or less safe in defining pWCET estimates and WCET thresholds. For example, s^1 from Equation (4.21) provides the least confident absolute pWCET as $\mathcal{C}_i^{s^1}$: $\mathcal{C}_i^{s^1}$ is the least safe absolute pWCET for τ_i; s^2 provides slightly more confidence that $\mathcal{C}_i^{s^2}$ is the absolute pWCET for τ_i: $\mathcal{C}_i^{s^2}$ is relatively safer than $\mathcal{C}_i^{s^1}$. Going on with the scenarios within S, the safety increases up until s^k which is the worst-case scenario and $\mathcal{C}_i^{s^k}$ is the only 100% safe absolute pWCET for τ_i; $\mathcal{C}_i^{s^k}$ is the safest among the pWCETs.

The MC task model makes use of multiple WCETs for characterizing the task behavior; such bounds results from different timing analysis tools as well as different criticality requirements that task can respect at runtime. For example, in the two-criticality-level case, each task is designated as being of either higher (HI) or lower (LO) criticality, and two WCETs are specified for each HI-criticality task: a LO-WCET determined by a less pessimistic tool or a less demanding safety requirements (e.g. mission-critical or non-critical) , and a larger HI-WCET determined by a more conservative tool or more safety-critical requirements.

For real-time systems, safety and criticality have a strong relationship so that they can be interchanged whenever applied for timing analysis and schedulability analysis: a safe pWCET is the worst-case models which can apply with high critical modes.

Least critical Scenario LO-critical. In case of s^1 from Equation (4.21), the task has the least execution time, thus the least dominating relative pWCET $\mathcal{C}_i^{s^1}$. $\mathcal{C}_i^{s^1}$ upper bounds any (and only) possible execution time resulting from s^1; it is the last safe absolute pWCET, equivalently the least critical LO-criticality. $\mathcal{C}_i^{s^1}$ is applied to characterize LO-criticality requirements of τ_1 and the LO-criticality functional mode.

Critical Scenarios MI-critical. From S ordered by dominance, Equation (4.21), s^2 dominates s^1 because under s^2 the task suffers execution times bigger than under s^1. Considering $\mathcal{C}_i^{s^2}$ as absolute pWCET, it would be slightly safer than $\mathcal{C}_i^{s^1}$, but it is not safe enough to upper bound the other $s^j \in S$. $\mathcal{C}_i^{s^2}$ is the middle criticality (MI-criticality) characterization for τ_i.
With s^3, it is $\mathcal{C}_i^{s^3}$ dominating $\mathcal{C}_i^{s^2}$ and $\mathcal{C}_i^{s^1}$ since s^3 produces larger execution times than s^1 and s^2. Thus, $\mathcal{C}_i^{s^3}$ would be safer than $\mathcal{C}_i^{s^2}$ as absolute pWCET. Also $\mathcal{C}_i^{s^3}$ is a MI-criticality characterization for τ_i but more critical than $\mathcal{C}_i^{s^2}$.
There are distinguished between MI-2-criticality and MI-3-criticality, respectively for s^2 and s^3, and $\mathcal{C}_i^{s^3} \succeq \mathcal{C}_i^{s^2}$. Other intermediate criticality levels can be defined from $s^j \in (S \setminus s^k)$.

Most Critical Scenario, HI-critical. The pWCET $\mathcal{C}_i^{s^k}$ from s^k is the safest absolute pWCET. $\mathcal{C}_i^{s^k}$ is also the HI-criticality bound to the task behavior. $\mathcal{C}_i^{s^k} \equiv \mathcal{C}_i^{s^{worst}}$ represents the worst conditions and is the most conservative upper bound for τ_i to be applied in the highest critical modes.
From $\mathcal{C}_i^{s^1}$ it would be possible to extract $\langle C_i(\text{LO}), 10^{-9}\rangle$. It is named $C_i(\text{LO})$ the LO-critical WCET threshold as it results from the least safe pWCET model $\mathcal{C}_i^{s^1} \equiv \mathcal{C}_i^{\text{LO}}$. $C_i(\text{LO})$ is the LO-criticality WCET threshold with a confidence of 10^{-9}.
s^j, with $1 < j < k$ form Equation (4.21), is a MI-j-criticality scenario that upper bounds all the scenarios s^r such that $r \leq j$; $\mathcal{C}_i^{s^j} \equiv \mathcal{C}_i^{\text{MI}-j}$ is the MI-j-criticality pWCET. $\mathcal{C}_i^{s^j} \succeq \mathcal{C}_i^{s^{j-1}}$ and $\langle C_i(\text{MI}-j), 10^{-9}\rangle$ is such that $C_i(\text{MI}-j) \geq C_i(\text{MI}-j-1)$; $C_i(\text{MI}-j)$ is the MI-j-criticality WCET threshold with a confidence of 10^{-9}.
s^k is the worst-case scenario and $\mathcal{C}_i^{s^k}$ is the absolute pWCET for τ_i. $\mathcal{C}_i^{s^k} \equiv \mathcal{C}_i^{\text{HI}}$ is the HI-criticality pWCET and s^k is the HI-criticality scenario for the worst conditions. From $\mathcal{C}_i^{\text{HI}}$, it is $\langle C_i(\text{HI}), 10^{-9}\rangle$ such that $C_i(\text{HI})$ is the HI-criticality WCET threshold. $\mathcal{C}_i^{\text{HI}} \succeq \mathcal{C}_i^{\text{MI}-j}$ and $C_i(\text{HI}) \geq C_i(\text{MI}-j)$.

From the difference in safety/criticality between s^{worst}, s^3, s^2 and s^1 execution conditions, it is $\mathcal{C}_i^{\text{HI}} \succeq \mathcal{C}_i^{\text{MI}} \succeq \mathcal{C}_i^{\text{LO}}$. Also, $C_i(\text{HI}) \geq C_i(\text{MI}) \geq C_i(\text{LO})$ for the same probability p from respectively $\mathcal{C}_i^{\text{HI}}$, $\mathcal{C}_i^{\text{MI}}$ and $\mathcal{C}_i^{\text{LO}}$. How much they differ depends on the relationship between the scenarios and the impact that the scenarios have on the execution times of tasks. $p = 10^{-9}$ is chosen arbitrarily, but the probabilistic modeling proposed can make use of any probability, depending on the confidence requirements.

The MC Worst-Case Set representation for τ_i is:

$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{\mathrm{LO}}, \dots, \mathcal{C}_i^{\mathrm{MI}-j}, \dots, \mathcal{C}_i^{s^k}). \tag{4.25}$$

For the intra- and inter-scenario perspective, adding criticality levels to Equation (4.23) and Equation (4.24) it is $\overline{C}_i \overset{def}{=} (C_i(\textsc{lo}), \dots, C_i(\textsc{hi}))$ for the inter-scenario MC representation $\langle \overline{C}_i, p\rangle$ at probability p and $\hat{C}(\updownarrow)_i \overset{def}{=} (\langle C(\chi_i)_{1,i}, p_1\rangle, \langle C(\chi_i)_{2,i}, p_2\rangle, \dots, \langle C(\chi_\rangle)_{m,i}, p_m\rangle)$ for the intra-scenario MC representation at the criticality level χ_i and probabilities $p_1, p_2, \dots p_m$.

MC probabilistic task model. With three criticality levels, the MC task model based on the Worst-Case Set is:

$$\tau_i = ([\overline{\mathcal{C}}_i, \langle \overline{C}_i, 10^{-9}\rangle, (\hat{C}_i^{\mathrm{LO}}, \hat{C}_i^{\mathrm{MI}}, \hat{C}_i^{\mathrm{HI}})], T_i, D_i), \tag{4.26}$$

where $\overline{\mathcal{C}}_i = (\mathcal{C}_i^{\mathrm{LO}}, \mathcal{C}_i^{\mathrm{MI}}, \mathcal{C}_i^{\mathrm{HI}})$ and $\langle \overline{C}_i, p\rangle = (C_i^{\mathrm{LO}}, C_i^{\mathrm{MI}}, C_i^{\mathrm{HI}})$. The intra-scenario representation is such that $\hat{C}_i^{\mathrm{LO}} = (\langle C_{1,i}^{\mathrm{LO}}, 10^{-3}\rangle, \langle C_{2,i}^{\mathrm{LO}}, 10^{-6}\rangle, \langle C_{3,i}^{\mathrm{LO}}, 10^{-9}\rangle)$ for the LO-safety,
$\hat{C}_i^{\mathrm{MI}} = (\langle C_{1,i}^{\mathrm{MI}}, 10^{-3}\rangle, \langle C_{2,i}^{\mathrm{MI}}, 10^{-6}\rangle, \langle C_{3,i}^{\mathrm{MI}}, 10^{-9}\rangle)$ for the MI-safety scenario and
$\hat{C}_i^{\mathrm{HI}} = (\langle C_{1,i}^{\mathrm{HI}}, 10^{-3}\rangle, \langle C_{2,i}^{\mathrm{HI}}, 10^{-6}\rangle, \langle C_{3,i}^{\mathrm{HI}}, 10^{-9}\rangle)$ for the HI-safety scenario.

The MC task model is essentially asserting that depending on the conditions for the timing analysis applied it is possible to have more or less guarantees on the pWCET and the WCET thresholds estimates. Only by considering most of the possibilities (necessarily the dominating ones) the MC worst-case models are safe. The MC task model can be generalized to multiple criticality levels and different probabilities in order to better cope with the requirements.

It is necessary to investigate the statistical independence between criticality levels and pWCETs for the MC Worst-Case Set representation. It has already been showed that there exist independence between absolute pWCETs, thus between HI-criticality representations $\mathcal{C}_i^{\mathrm{HI}}$ and $C_i(\textsc{hi})$ from s^k.

Supposing s^j represents a criticality level other than HI-criticality, what happens to the pWCET estimates of τ_i and τ_k? Under s^j, the conditional probability $\mathsf{pdf}_{\mathcal{C}_i^{s^j}|\mathcal{C}_k^{s^j}}$ equals $\mathsf{pdf}_{\mathcal{C}_i^{s^j}}$ (equivalently $\mathsf{pdf}_{\mathcal{C}_k^{s^j}|\mathcal{C}_i^{s^j}}$ equals $\mathsf{pdf}_{\mathcal{C}_k^{s^j}}$) because all the effects from s^j have been included into the relative pWCETs $\mathcal{C}_i^{s^j}$ and $\mathcal{C}_k^{s^j}$. This assures tasks independence with the same scenario.

With s^r dominating s^j for both τ_i and τ_k, what happens to $\mathcal{C}_i^{s^r}$ and $\mathcal{C}_k^{s^j}$? It is $\mathsf{pdf}_{\mathcal{C}_i^{s^r}|\mathcal{C}_k^{s^j}} = \mathsf{pdf}_{\mathcal{C}_i^{s^r}}$, since all the effects of s^j and τ_k on τ_i have been already taken into account by $\mathcal{C}_i^{s^r}$. This guarantees the independence between τ_i and τ_k under s^r and s^j, respectively for τ_i and τ_k.

Worst-Case Set representations guarantee tasks independence and independence between criticality levels which will ease task combination and schedulability analysis.

Faults (either transient or permanent) translate into penalties δ (latency) to the task execution time which depends on the time t the fault happens, $\delta(t)$. With $C(t)$ the expected task execution time at time t, in presence of fault it would be $C(t) + \delta(t)$ the task execution time accounting for the fault penalty on task computations. With a measurement-based approach, fault effects on task execution can be measured and directly embedded into traces of execution time measurements; then, with EVT it is possible infer pWCET estimates which upper bounds faulty execution conditions.

Different scenarios are possible with respect to faults. By considering non-faulty conditions (fault never happening), it is scenario s^{NF} that describes the task behavior. Here, the execution times observed exploit only the task functional behavior due to the absence of faults. For s^{NF} it exists $\mathcal{C}_i^{s^{NF}}$; s^{NF} is the LO-criticality scenario with $\mathcal{C}_i^{\mathrm{LO}} \equiv \mathcal{C}_i^{NF}$, $C(\textsc{lo})$ and $\hat{C}_i(\textsc{lo})$ representing it.

It could also exist s^{FW} which assumes that the worst fault condition manifests at runtime; $\mathcal{C}_i^{FW}$ is the pWCET estimate for s^{FW}. s^{FW} is the wort-case scenario where the task executes always under the most critical conditions; $\mathcal{C}_i^{\mathrm{HI}} \equiv \mathcal{C}_i^{FW}$, $C_i(\textsc{hi})$ and $\hat{C}_i(\textsc{hi})$ represents it.

In between these two extreme scenarios, it exist a set of possible faulty scenarios where faults are not as extreme as s^{FW} and s^{NF}, nonetheless they happen and affect the normal task behavior. For example, it could exist s^{F1} which is the MI-1-criticality scenario with $\mathcal{C}_i^{\mathrm{MI}-1} \equiv \mathcal{C}_i^{s^{F1}}$, $C_i(\textsc{mi}-1)$ and $\hat{C}_i(\textsc{mi}-1)$; $C_i(\textsc{mi}-1) \geq C_i(\textsc{lo})$ and $\mathcal{C}_i^{\mathrm{MI}-1} \succeq \mathcal{C}_i^{\mathrm{LO}}$. It could also exist s^{F2} as the MI-2-criticality scenario with $\mathcal{C}_i^{\mathrm{MI}-2} \equiv \mathcal{C}_i^{s^{F2}}$, $C(\textsc{mi}-2)$ and $\hat{C}_i(\textsc{mi}-2)$; $C_i(\textsc{mi}-2) \geq C_i(\textsc{mi}-1)$ and $\mathcal{C}_i^{\mathrm{MI}-2} \succeq \mathcal{C}_i^{\mathrm{MI}-1}$.

Specific to faults and faulty scenarios, it is $S = \{s^{NF} \equiv s^{\mathrm{LO}}, s^{F1} \equiv s^{\mathrm{MI}-1}, s^{F2} \equiv s^{\mathrm{MI}-2}, \dots, s^{FW} \equiv s^{\mathrm{MI}}\}$ with the task MC Worst-Case Set given by $\overline{\mathcal{C}}_i = (\mathcal{C}_i^{s^{\mathrm{LO}}}, \mathcal{C}_i^{s^{\mathrm{MI}-1}}, \mathcal{C}_i^{s^{\mathrm{MI}-2}}, \dots, \mathcal{C}_i^{s^{\mathrm{HI}}})$.

What is about to be proposed is a representation framework that applies to faults effects. It could abstract different faults and fault tolerant mechanisms implemented as recovery functions or task extra-executions resulting into larger task execution times and worst-case execution times.

The complexity from such MC representations is not considered here, preferring at this stage to investigate the benefits of a detailed probabilistic model to task MC behaviors.

4.2.2 Probabilistic sensitivity analysis for mixed criticality

[191] is the first work on defining probabilistic sensitivity analysis for probabilistic MC problems. This work is the result of the collaboration between Prof. Laurent George and Prof. Zhishan Guo on probabilistic sensitivity analysis for MC problems. The attempt is to leverage probability information into scheduling and trade-off decisions aiming at resource usage parametrized with probability and criticality level. In here, it is defined the notion of parameter β, and how it can be applied for probabilistic sensitivity analysis.

Probabilistic bounds. Real-time systems with probabilistic models require schedulability conditions which involve probabilities. Given a random process $\mathcal{C}_i$ describing the evolution of τ_i worst-case execution time, it is possible to state the notion of probabilistic demand bound function, [189, 197].

The $\mathsf{dbf}_{i,j}(t)$ is the demand bound function in the interval $[0, t]$: $\mathsf{dbf}_{i,j}(t) \overset{def}{=} \lfloor \frac{t-D_i}{T_i} + 1 \rfloor \times c_{i,j}$. The bound is the result of a specific WCET threshold $c_{i,j}$, and by definition, it represents an upper-bound to any $\mathsf{dbf}_{i,k}$ obtained from $c_{i,k} \leq c_{i,j}$. The confidence of the bound associated is the confidence $p_{i,j}$ of the WCET $c_{i,j}$ applied. $c_{i,j}$ defines a probabilistic bound $\langle \mathsf{dbf}_{i,j}(t), p_{i,j} \rangle$ to the task resource demand; $\langle \mathsf{dbf}_{i,j}(t), p_{i,j} \rangle$ is such that $p_{i,j}$ is the probability that $\mathsf{dbf}_{i,j}(t)$ is an upper-bound to the τ_i resource demand in $[0, t]$. Equivalently to $\mathsf{dbf}_{i,j}(t)$ it can be written $\mathsf{dbf}_i(t, c_{i,j})$.

As $\langle \mathsf{dbf}_i(t, c_{i,j}), p_{i,j} \rangle$ represents a single demand bound function with its associated confidence, it exists a $\mathsf{dbf}_i(t, c_{i,j})$ for each $c_{i,j} \in \mathcal{C}_i$. All together the $\langle \mathsf{dbf}_i(t, c_{i,j}), p_{i,j} \rangle$ form a distribution of demand bound functions, $\mathcal{DBF}_i(t) = (\mathsf{dbf}_i(t, \cdot), p_i(\cdot))$ which is the probabilistic demand bound function (probabilistic demand curve) of τ_i in $[0, t]$. $\mathcal{DBF}_i(t)$ collects the set of all demand bound functions $\mathsf{dbf}_i(t)$ and the set of all confidences p_i. In particular, the p_is forms the CDF of $\mathcal{DBF}_i(t)$, $\mathsf{cdf}_{\mathcal{DBF}_i(t)}$, as cumulative probabilities. To note that the set of probabilities p_i does not change with the interval $[0, t]$, therefore form one interval to another is only the bounds $\mathsf{dbf}_i(t)$ to change, but not their confidence.

The application Γ probabilistic demand curve $\mathcal{DBF} = (\mathsf{dbf}(t, \cdot), p(\cdot))$ results from the combination of tasks demand bound functions $\mathcal{DBF}_i$:

$$\mathcal{DBF}(t) = \otimes_i \mathcal{DBF}_i(t), \tag{4.27}$$

with $\otimes$ the convolution of the distributions. $\mathsf{dbf}(t, \cdot)$ is the set of all the demand bound functions:

$$\mathsf{dbf}(t, \cdot) \overset{def}{=} +_i \mathsf{dbf}_i(t, \cdot), \tag{4.28}$$

with $+$ the combination (sum) of all the demand bound function. $p(\cdot)$ is the set of all the confidence probabilities:

$$p(\cdot) \overset{def}{=} \times_i p_i(\cdot), \tag{4.29}$$

with $\times$ the combination (product) of all the demand bound function probabilities.

The demand bound function $\mathsf{dbf}(t, \bar{c})$ is the application demand with $\bar{c} = (c_{1,j}, c_{2,k}, \ldots)$ the array of WCET thresholds used for achieving $\mathsf{dbf}(t, \bar{c})$; $p(\bar{c})$ is the confidence of $\mathsf{dbf}(t, \bar{c})$ such that:

$$p(\bar{c}) = p_1(c_{1,j}) \times p_2(c_{2,k}) \times \ldots. \tag{4.30}$$

The probability multiplication for the joint probability $p(\bar{c})$ is possible due to the worst-case distribution assumption. As $\mathcal{C}_i$ are pWCETs they are independent, the distributions $\mathcal{DBF}_i$ are independent among each other; consequently the joint probability $p(\cdot)$ could result from the probability multiplication, Equation (4.30).

The schedulability under EDF states that

$$\forall t \in D, \quad \mathsf{dbf}(t) \leq t, \tag{4.31}$$

with D the set of Γ deadlines within the hyperperiod, according to [29, 31].

With a probabilistic framework each condition $\mathsf{dbf}(t, \cdot) \leq t$ has a probability $p(t)$ associated, which is the confidence on the demand bound function $\mathsf{dbf}(t, \cdot)$. Being $p = p(\bar{c})$ the probability of not passing $\mathsf{dbf}(t, \bar{c})$, with the condition $\mathsf{dbf}(t) \leq t$ the probability p could be also interpreted as the probability of verifying the condition.

For all $t \in D$, it exist $\overline{c}^*$ such that $\mathsf{dbf}(t, \overline{c}^*) = max\{\mathsf{dbf}(t, \overline{c}) \mid \mathsf{dbf}(t, \overline{c}) \leq t\}$. $P(t) = p(\overline{c}^*)$ from Equation (4.30) is the probability for which $\mathsf{dbf}(t) \leq t$ is true. The overall schedulability probability P is given such that all the conditions are satisfied:

$$P = P(t_1) \times P(t_2) \times \ldots, \tag{4.32}$$

with $P(t_k)$ the schedulability probability of the k-th condition $\mathsf{dbf}(t_k) \leq t_k$, $t_k \in D$. The independence between conditions and the probability product as the joint probability, are guaranteed by the use of pWCET distributions. $1 - P$ is the probability that at least one condition is not respected, thus the probability of deadline miss.

Probabilistic C-space. From Condition (4.31) and Equation (4.32) it is possible to build the probabilistic version of the C-space (pC-space), [65]. The pC-Space is the abstraction that applies the schedulability condition, Condition (4.31), to a vector of execution times $\overline{c} = \{c_1, c_2, \ldots\}$. Each point $\overline{c} = \{c_1, c_2, \ldots\}$ in the pC-Space is a combination of task WCET thresholds. Within the pC-Space, given the scheduling policy, it is possible to define the schedulability region where every point $\overline{c}$ within the region is a schedulable WCET thresholds configuration, and the points outside the region do not represent schedulable WCET thresholds configurations. [65] for the details on the definition of the deterministic C-space under EDF.

The pC-Space maps also probabilities onto points. Each $\overline{c}$ within the space has a probability associated which is the probability of being the application set of worst-case execution times, Equation (4.30). Then, depending on where the point is with respect to the schedulability region, the probability could translate into schedulability probability. For the points at the feasibility region border, their ps, Equation (4.30), are exactly the schedulability probability, Equation (4.32). With the different probabilities P within the region and at the border it would be possible to classify portions of the regions with respect to the schedulability probability P.

The probabilities within the pC-Space can be interpreted in various ways:

- As the confidence of not passing the WCET thresholds of $\overline{c}$. With the criticality levels there is also the probability of remaining at a certain criticality level $p(crit) = p_1(crit) \times p_2(crit) \times \ldots$. Consequently it is quantifiable the possibility of changing that level as $1 - p(crit)$.

- As the confidence on the system schedulability P, or the confidence per schedulability condition, P_k. The feasibility region is characterized by P and all the points inside the region are schedulable but with a confidence of at least P, Equation (4.32). It translates into a per-condition schedulability probability of P_k.

- As the confidence β on the worst-case behavior of the tasks. β is the probability of passing the $\overline{c}(\beta)$; per-task it would be $c_j(\beta)$.

With different probability interpretations the pC-Space can be used for different purposes. At one end there is the modeling of the probabilistic applications; on the other end, it is possible to develop analysis on top of the pC-Space with probabilities.

pC-Space for MC. The pC-Space is the extension of C-space developed for pWCETs applying the new task model of Equation (4.26) for different execution scenarios and different safety levels. The pC-Space is used for formalizing the sensitivity sensitivity and the schedulability analysis with probabilities and safety levels. The pC-Space accounts for the fault models with the discrete and finite possible execution conditions per task, Equation (4.26). pC-Space is a space built on tasks WCET thresholds of Equation (4.23), where it is possible to represent schedulable task set, safety levels and confidence probabilities. A point $\overline{c}$ outside the feasibility region is a task set not schedulable.

The task model $\tau_i = ((C_i(\text{LO}), C_i(\text{HI})), T_i, D_i)$ results into the set of dbfs:

$$\mathsf{dbf}_i(t, C_i(\text{HI})) \overset{def}{=} \lfloor \frac{t - D_i}{T_i} + 1 \rfloor \times C_i(\text{HI}) \text{ and } \mathsf{dbf}_i(t, C_i(\text{LO})) \overset{def}{=} \lfloor \frac{t - D_i}{T_i} + 1 \rfloor \times C_i(\text{LO}),$$

with $\mathsf{dbf}_i(t, C_i(\text{HI})) \geq \mathsf{dbf}_i(t, C_i(\text{LO}))$. Some tasks combinations accounting for the safety levels are:

$$\begin{aligned}
\overline{c}^{\text{LO}} &= (C_1(\text{LO}), C_2(\text{LO}), \ldots, C_n(\text{LO})),\\
\overline{c}^{1} &= (C_1(\text{LO}), C_2(\text{LO}), \ldots C_n(\text{HI})),\\
\overline{c}^{2} &= (C_1(\text{LO}), C_2(\text{LO}), \ldots, C_{n-1}(\text{HI}), C_n(\text{LO})),\\
&\ldots ,\\
\overline{c}^{\text{HI}} &= (C_1(\text{HI}), C_2(\text{HI}), \ldots C_n(\text{HI})).
\end{aligned}$$

As a result, there exist multiple possible demand bound functions for the set of tasks depending on the WCET thresholds chosen:

$$\begin{aligned}
\mathsf{dbf}(t, \overline{c}^{\mathrm{LO}}) &= \sum_{i}^{n} \mathsf{dbf}_i(t, C_i(\textsc{lo})),\\
\mathsf{dbf}(t, \overline{c}^{1}) &= \sum_{i=1}^{n-1} \mathsf{dbf}_i(t, C_i(\textsc{lo})) + \mathsf{dbf}_n(t, C_n(\textsc{hi})),\\
\cdots &,\\
\mathsf{dbf}(t, \overline{c}^{\mathrm{HI}}) &= \sum_{i=1}^{n} \mathsf{dbf}_i(t, C_i(\textsc{hi})).
\end{aligned}$$

As a reminder, the combinations and dbfs here before are not all the possible combinations. They are a subset where each element dominates other configurations i.e., upper-bounds other configurations.

The different WCET thresholds per task allow enriching the pC-Space representation with more informations about the safety or not of the points $\overline{c}$. Moreover, by considering the probability associated to the C_is, it is possible to add probabilities to the pC-Space. Indeed, with $\langle C_{i,j}, p_{i,j}\rangle$, the resulting demand bound function $\mathsf{dbf}_i(t, C_{i,j}) = \lfloor \frac{t-D_i}{T_i} + 1 \rfloor \times C_{i,j}$ has $p_{i,j}$ associated which is the confidence on the demand bound function upper-bounding task behaviors. With probability $p_{i,j}$, τ_i demands more computation time than $\mathsf{dbf}_i(t, C_{i,j})$. For a point $\overline{c}$, the probability is given by the product of the WCET thresholds composing $\overline{c}$ due to the pWCET independence. The HI and LO safety levels and the probabilities remain separated.

The probabilistic version of the sensitivity analysis [65] intends to combine the information from the probabilistic models (the pWCETs, β, and the confidences β and ps) and the pC-Space representation.

The $\mathcal{C}_i$s discretize the pC-Space as they maps the points to only the possible WCET thresholds of the tasks. Out of that, the probabilistic sensitivity analysis can be applied to quantify the effects of changes in terms of schedulability, probabilities/confidences, and criticality.

- What are the resource demand that can be accommodated? Hence, which task combinations can be accounted for a schedulable systems, the criticality levels that can be considered in order to make the system schedulable, etc..
- What can be done with β? By acting on β (limiting task WCETs to $c(\beta)$) it is possible to evaluate the effect on the execution of tasks. What are the effect of β on the tasks criticality levels? With the relationship $\beta \to crit$ is is possible to infer the criticality levels which subdue to the $\overline{c}(\beta)$ bounding.

Furthermore, with the probabilistic sensitivity analysis it is possible to evaluate the effect of changes on Γ. For example a change on β, from β to β' would result into a WCET threshold change $\overline{c}$ to $\overline{c}'$, such that $\overline{c} = (c_{1,j}, c_{2,k}, \ldots)$ and $\overline{c}' = (c_{1,r}, c_{2,s}, \ldots)$. The change of probabilities, from $p(\overline{c})$ to $p(\overline{c}')$, is an immediate consequence of the change of β. It would also be evident the effects of changes on the allowed criticality levels, from $\beta \to crit$.

While P does not change by moving the points toward the feasibility region (by limiting task execution behavior with β), it is possible to increase the confidence that the feasibility condition is respected.

4.2.3 Open problems with probabilistic sensitivity analysis

With the pWCETs it is possible to construct the probabilistic version of the C-space (pC-space) and the probabilistic version of the (α, Δ)-space. This work intends to provide an initial evaluation to the flexibility brought by the probabilistic models and the probabilistic scheduling to the mixed criticality problem. The sensitivity analysis is enhanced with probabilities, and the paper illustrates some initial thoughts and possible strategies for resource allocation with mixed critical tasks are provided. It has been proposed to the open problems workshop because how to efficiently use the sensitivity analysis remains an open problem.

This abstract presented at RTSOPS is for illustrating possible ways to apply sensitivity analysis with probabilistic models and ease the complexity of actual probabilistic schedulability analysis. The sensitivity analysis would allow exploring the flexibility of the probabilistic models into real-time systems design and development. Moreover, the sensitivity analysis would directly link probabilities to scheduling conditions and effectively apply them into scheduling decisions. This is an open problem since so far there are not proposed effective works on that. The abstract illustrates ideas around sensitivity analysis and probabilities which could lead to unexplored contributions for reducing the pessimism and implementing better probabilistic schedulers.

(α, Δ)-space

With probabilities there exist multiple probabilistic feasibility regions depending on the WCET thresholds $\overline{C}$ applied. Each $\langle \Phi_\Gamma(\overline{C}, P\rangle$ is a feasibility region $\Phi_\Gamma(\overline{C})$ from the WCET thresholds selected $\overline{C}$ and the probability P associated to it. P is the same as the one of $\langle \mathsf{dbf}(t, \overline{C}), P\rangle$ and is the probability of exceeding $\Phi_\Gamma(\overline{C})$; it can be also interpreted as the probability of verifying the condition $\mathsf{dbf}(\cdot,\cdot) \leq \mathsf{sbf}$, thus the 'schedulability probability'.

In (α, Δ)-space, the Euclidean distance $(\delta\alpha = \alpha_2 - \alpha_1, \delta\Delta = \Delta_2 - \Delta_1)$ defines the distance between two resource supply lsbf; it can be extended with probabilities and used for sensitivity analysis.

With the probabilistic (α, Δ)-space, the sensitivity analysis would:
1) Evaluate the resource demand for probabilistic schedulability answering to: For a specific probability, what are the resource provisioning necessary to guarantee the probabilistic real-time application? Resource evaluation can be applied to develop and guarantee strategies that change resource provisioning and obtain specific probabilistic schedulability levels.
2) Define and explore the resource provisioning-probabilistic schedulability trade-offs. The trade-offs are to explore the effects that the WCET thresholds have on the schedulability and on the resource necessary to achieve certain probabilistic schedulability levels.

The proposal is to do sensitivity analysis with the use of a parameter β applied to pWCETs. β would define the cumulative probability at which extract pWCET thresholds from a pWCET. It is possible to define a unique β for Γ or a β_i per task τ_i, $\overline{\beta}_i = \{\beta_1, \beta_2, \ldots\}$. β is represented in Figure 4.5 with the multiple possible WCET threshold extracted from a continuous distribution. The more β reduces, the more the WCET threshold increases and becomes more "confident" to be the worst-case single-value model. β applies the same way to discrete pWCET distributions.

In Figure 4.6 there is an example of that where the 5 feasibility regions derived from 5 different $\overline{\beta}$. The arrows propose possible changes to resource reservations in order to achieve specific properties. The sensitivity analysis with strategies for changes and the use of β is under development. Under definition there are the metrics for quantifying the advantages from probabilistic models (pessimism reduced for different degrees of schedulability). The complexity of the probabilistic analysis and for developing strategies appears reduced with the (α, Δ).

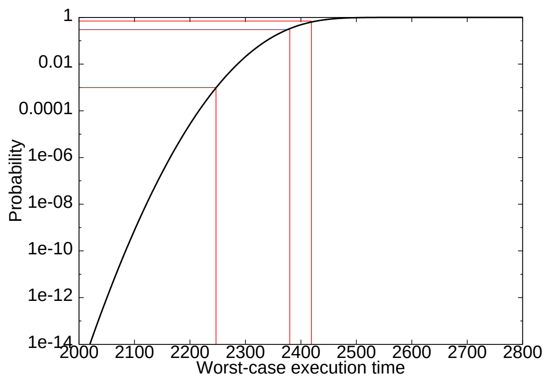


Figure 4.5: β thresholds on a continuous distribution in its CDF representation. From β, WCET thresholds can be extracted to impose certain behavior to tasks.

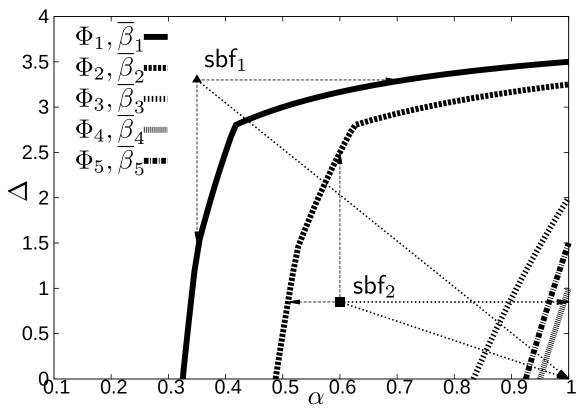


Figure 4.6: (α, Δ)-space and the sensitivity analysis that applies to it.

β is the key parameter for sensitivity analysis to be investigated in the future and to build strategies around. In [184, 191] the properties of β parameter begin to be defined. Future contributions will be proposed to develop β into a scheduling and trade-off worthy parameter.

C-space

From probabilistic models (pWCET and probabilistic dbf), it is possible to build the probabilistic version of the C-space, such that each point $\bar{c}$ has a P probability associated. $\bar{c} = \{c_1, c_2, \ldots\}$ is a combination of task WCET thresholds (possible WCET thresholds, one per task, like $\overline{C}$), while the probability P is the probability of exceeding such thresholds $P = P_1(c_{1,j}) \times P_2(c_{2,k}) \times \ldots$. In the probabilistic the C-space, P cannot represent the schedulability probability.

Given the scheduling policy, in the probabilistic C-space there is the feasibility region where every point $\bar{c}$ within the region is a schedulable WCET thresholds configuration. The points outside the region do not represent schedulable WCET thresholds configurations. Figure 4.7 shows an example probabilistic real-time application with three tasks, and its pC-Space in 2D form; to each point, there is a probability associated.

With the probabilistic C-space it is possible to apply Euclidean distance ($\delta c_1 = c_{1k} - c_{1j}, \delta c_2 = c_{2r} - c_{is}, \ldots$) combined with probabilities, and evaluate the impact that WCET choices have on the schedulability. The sensitivity analysis for the probabilistic C-space can also work with parameter β. It is under discussion how to develop effective sensitivity analysis for the probabilistic C-space and with β. The goal is also to show a reduced complexity of the probabilistic schedulability analysis from the sensitivity analysis with the C-space.

It is possible to define a design parameter β as the *probability threshold* for the pWCET defining the level of confidence for a WCET limit imposed to a task. β comes from the quantile $q(p)$ as the probability threshold p, and $q(\mathcal{C}_i, \beta)$ is the WCET threshold such that $\beta \times 100\%$ of the worst-case execution time experienced by τ_i are below that threshold.

β offers another perspective to the task execution model. By fixing β it is possible to specify which is the limit WCET reachable, $c_i(\beta)$; β imposes a bound to the task WCET such that $c_i(\beta) = q(\mathcal{C}_i, \beta)$.

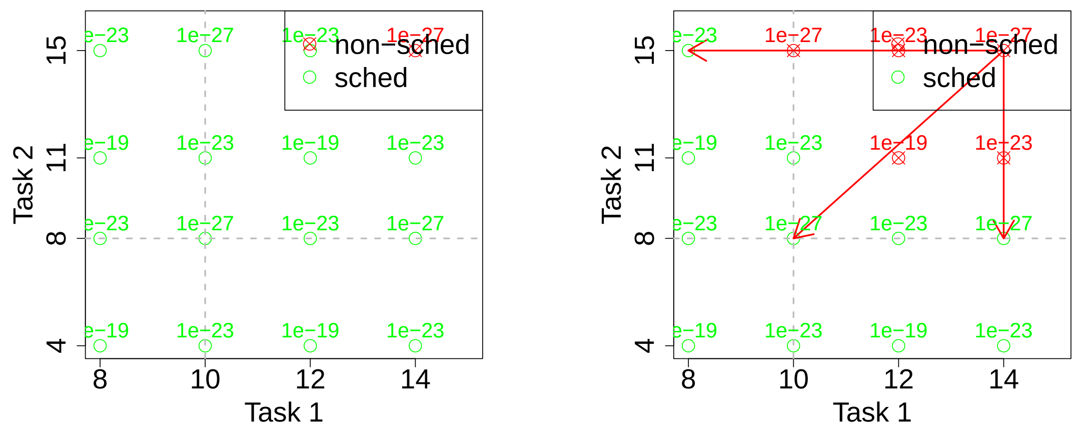
A trace $\mathcal{T}_{\mathcal{C}_i}$ reports the sequence of WCET values that τ_i has assumed from one execution instance to another. From $\mathcal{T}_{\mathcal{C}_i}$ it is then possible to infer the timing behavior of the task WCETs as well as identify $c_i(\beta) = q(\mathcal{T}_{\mathcal{C}_i}, \beta)$. Therefore, β can model the task (or the whole application) timing behavior and it could be applied as design parameter: by imposing $c_i(\beta)$ as the task WCET value the behavior of τ_i is limited to $c_i(\beta)$. With respect of the actual task behavior $\mathcal{T}_{\mathcal{C}_i}$ (which follow $\mathcal{C}_i$, β is the confidence that τ_i respects it WCET limit $c_i(\beta)$.

From β it is also possible to infer the criticality level *crit* that would allow respecting $c_i(\beta)$:

$$max\{crit\} \text{ such that } c_i(crit) \leq c_i(\beta). \tag{4.33}$$

It is $\beta \neq p(crit)$, as $c(\beta) \neq c(crit)$, but there is a close relationship between the two thresholds $c(\beta)$ and $c(crit)$ which come from the probabilistic modeling of the task (the pWCET).

In Figure 4.7 there are represented possible choices to change task WCET and reach schedulability condition in 2D projections of multiple dimensions pC-Spaces; such changes have impact also on the probabilistic task models. The example here is a 2D representation extracted from a 3D space of a real-time application of 3 tasks $\{\tau_1, \tau_2, \tau_3\}$.



(a) 2D plane (τ_1, τ_2), $C_3 = 15$ (b) 2D plane (τ_1, τ_2), $C_2 = 20$

Figure 4.7: 2D representations with safety levels HI and LO to distinguish effects on schedulability. Some possible transitions from non-schedulability to schedulability are represented.

In [191] the focus is on the pC-Space in defining β parameter interests to probabilistic sensitivity analysis and MC. Future contributions will be proposed to develop the pC-Space (at the actual stage, the (α, Δ)-space and sensitivity analysis in it is more developed than in the pC-Space) and how to cope β properties with pC-Space formalism.

4.2.4 Fault-aware sensitivity analysis for probabilistic real-time systems

In [194], in collaboration with Prof. Laurent George and Prof. Zhishan Guo, there are presented some development in modeling tasks in presence of faults. Probabilistic models comes from different faulty conditions and various effects that fault have on task behaviors. The impact of faults onto MC schedulability is also investigated.

Fault models are applied for abstracting the effects that faults have on tasks executions; the faults are modeled according to probability laws which characterize fault manifestation and fault impacts in realistic systems. The task execution behavior is represented in both non-faulty conditions and multiple faulty conditions; pWCET models are proposed for the different system execution conditions. The pWCETs are obtained with measurement-based probabilistic timing analysis from measurements of the actual task behavior, in presence or absence of faults. Furthermore, it is proposed the formalization of the scheduling problem for probabilistic task models i.e., pWCETs, and sensitivity analysis applied to task models and schedulability analysis to quantify fault effects on system schedulability. Performance-predictability trade-offs are exploited through the fault modeling and the analysis. A case study is applied to validate the framework and illustrate the benefits of probabilistic modeling and schedulability analysis to design and validate real-time systems.

This is a generic work which apply to any possible fault that can happen to task executions. Since fault effects are plugged into measurements, it would be possible to include multiple fault model representations from different distributions laws.

Probabilistic worst-case execution time modeling

A real-time task consists of a sequence of recurring jobs; each job must complete execution before a given deadline.

In a periodic task system, a task is described with the tuple $(O_i, T_i, D_i, \mathcal{C}_i)$. O_i is the offset that specifies the time instant at which the first job of τ_i is released; T_i is the period representing the temporal separation between two successive jobs; D_i is the deadline that defines the time interval in which task execution has to terminate; $\mathcal{C}_i$ is the worst-case execution time defining the execution processing requirements for each job. In a probabilistic framework, $\mathcal{C}_i$ is the worst-case distribution pWCET that upper bound any task execution time.

The scheduling policy decides the task execution ordering, possibly with task preemption or migration between cores. Schedulability analysis of task models guarantees the respect of the timing constraints (deadline); in particular, it checks if there are enough resources for the tasks to finish executions by their deadlines.

The MBPTA applies the Extreme Value Theory (EVT) for computing pWCET estimates out of measured behaviors, [156, 162], [101, 142]. The EVT guarantees that if certain hypotheses are verified, from the actual measured behavior it is possible to infer rare events, where the worst-case execution time lie; whenever correctly applied, the EVT produces a continuous distribution which is a safe pWCET estimate $\mathcal{C}_i$.

The guarantees from EVT to the worst-case are still questionable, depending on what is measured, e.g. the execution conditions for the measurements, the confidence or representativity of the measurements. A trace of execution time measurements accounts for some of the interfering conditions and the inputs which happens at runtime; the pWCET estimate $\mathcal{C}_i$ from the EVT embeds those system conditions and others that have not been measured, since the EVT is able to infer some of the unknown from the known measurements.

An execution scenario s^j abstract the execution conditions as an instance of the inputs I, of the environment Env, of the task mapping Map, etc., $s^j(I, Env, Map, \ldots)$. Given s^j, the worst-case profile for s^j is $\mathcal{C}_i^{s^j}$ and it comes from the measurements taken under s^j. For a real-time system, there exist a finite set S of all the possible execution scenarios, $S = \{s^j\}$, and the only way for guaranteeing safe pWCET estimates $\mathcal{C}_i$ is to account for all the scenarios within S.

Worst-case scenario: The scenarios in S can be ordered based on the pWCETs associated to them. There exist scenarios with worst and less worst execution time measurements and consequently worst and less pWCETs. The worst scenario s^{worst} in S would be the scenario that produces the worst interference on the task and the worst pWCET. Analyzing S with respect to the *worst-case scenario* consists of seeking for s^{worst} and then estimating $\mathcal{C}^{s^{worst}}$ from execution time measurements under it; the worst pWCET $\mathcal{C}_i$ is $\mathcal{C}_i \stackrel{def}{=} \mathcal{C}^{s^{worst}}$.

Envelope: With S the finite set of possible measurement scenarios for the system, the worst worst-case estimate $\mathcal{C}_i$ could be defined as an envelope of all the possible probabilistic profiles: $\mathcal{C}_i \stackrel{def}{=} max_{s^j \in S}\{\mathcal{C}_i^{s^j}\}$. With the ICDF it is:

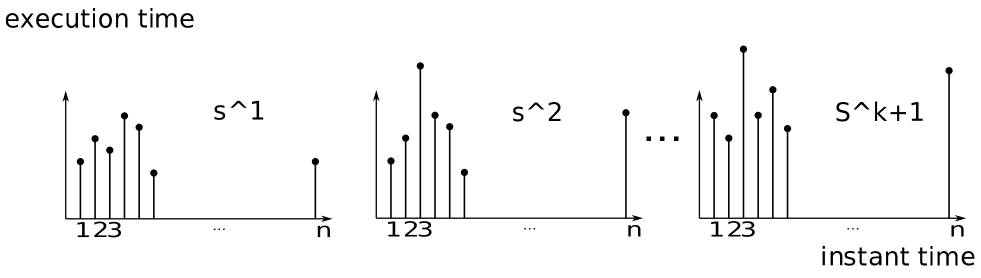
$$F'_{\mathcal{C}_i}(C) \stackrel{def}{=} max_{s^j \in S}\{F'_{\mathcal{C}_i^{s^j}}(C)\}. \tag{4.34}$$

This approach to worst-case profiling is named the envelope [192]. The worst pWCET could results not necessarily from an actual scenario, but from multiple scenarios contributing to the worst-case.
Worst-case set: It is possible to define a task execution model that keeps all the scenarios estimates $\mathcal{C}_i^{s^j}$; the *Worst-case set* representation collects all the pWCET from S as a set of pWCET estimates $\overline{\mathcal{C}}_i$:

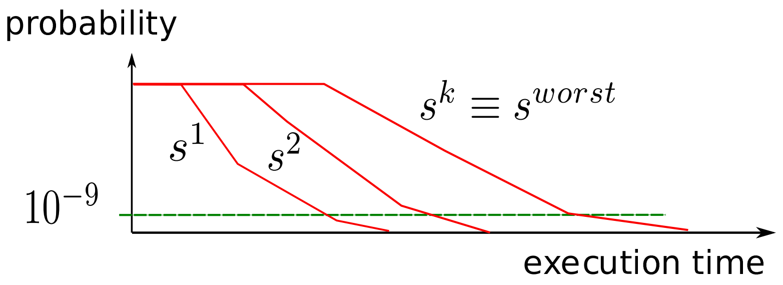
$$\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^1}, \mathcal{C}_i^{s^2}, \ldots, \mathcal{C}_i^{s^{|S|}}). \tag{4.35}$$

In this work it is used the *Worst-Case Set* representation, Equation (4.35), because it is more accurate and complete in describing the multiple possible task behaviors than the Worst-case scenario or the Envelope. Figure 4.8(a) depicts an example of traces of execution time measurements, one per scenario s^j. Figure 5.8(d) illustrates the three enumeration approaches (*worst-case scenario*, *envelope* and *worst-case set*) to task probabilistic worst-case modeling applied to $k+1$ different scenarios; in there, there exist a worst-case scenario s^{worst} which is also the envelope to all the pWCETs.

Scenario dominance. Although with today's real-time systems it is reasonable to assume a finite number of measurement scenarios, enumerating all the scenarios remains a complex problem. What would effectively allow applying the *orst-Case Set* representation, the *envelope* or the *worst-case scenario* is the existence of scenarios which *dominate* other scenarios.
Scenario dominance is in the sense that a dominant scenario s^k has larger pWCET than the pWCETs from dominated scenarios s^j. The partial ordering between pWCET can be defined according to [48]. With dominance between scenarios it would be possible neglecting the dominated scenarios and ease the task representation. For example, in case of Equation (4.35) fewer dominating scenarios could be considered to represent the whole S, $\overline{\mathcal{C}}_i \overset{def}{=} (\mathcal{C}_i^{s^k}, \mathcal{C}_i^{s^j}, \mathcal{C}_i^{s^r})$; in here, s^k dominates some scenarios in S, s^j dominates other scenarios as well as s^k and its dominated scenarios, s^r dominating all the scenarios. Figure 5.8(d) shows and example of scenario dominance where s^k dominates all the scenarios except s^{worst} and s^2.



(a) Different execution time scenarios and the resulting traces



(b) Different execution time scenarios and the resulting worst-case estimates with the ICDF representation

Figure 4.8: Representations of traces of execution times and pWCETs for different execution scenarios.

Probabilistic representations. With probabilistic models such the Worst-Case Sets of Equation (4.35), multiple are the possible representations for the task behavior. Two of them are hereby listed, named respectively *inter-scenario* and *intra-scenario* representations, which accounts for probability and scenario information.

Inter-scenario representation: Given a probability p and the WCET threshold $\langle C_i^{s^j}, p\rangle$ at p for the scenario s^j, for each scenario $s^j \in S$ the inter-scenario task set WCET threshold $\langle \overline{C}_i, p\rangle$ is such that:

$$\overline{C}_i \overset{def}{=} (C_i^{s^1}, C_i^{s^2}, \ldots, C_i^{s^{|S|}}). \tag{4.36}$$

In particular, it is possible to pick $p = 10^{-9}$, $\langle C_i^{s^j}, 10^{-9}\rangle$ $\forall s^j \in S$, and have the task set WCET threshold $\langle \overline{C}_i, 10^{-9}\rangle$. The inter-scenario representation $\langle \overline{C}_i, p\rangle$ of Equation (4.36) makes use of all the scenarios (at least the dominating ones) for characterizing the task execution behavior.

Intra-scenario representation: For a given scenario $s^j \in S$ and a set of exceeding thresholds probabilities $(p_1, p_2, \dots p_m)$, the set of WCET thresholds $\hat{C}_i^{s^j}$ for s^j is such that:

$$\hat{C}_i^{s^j} \overset{def}{=} (\langle C_{1,i}^{s^j}, p_1\rangle, \langle C_{2,i}^{s^j}, p_2\rangle, \dots, \langle C_{m,i}^{s^j}, p_m\rangle). \tag{4.37}$$

The intra-scenario representation focuses on a specific scenario with all the meaningful WCET thresholds and probabilities for that scenario. For example, with $(10^{-6}, 10^{-9}, 10^{-12})$ it would be
$\hat{C}_i^{s^j} = (\langle C_{1,i}^{s^j}, 10^{-6}\rangle, \langle C_{2,i}^{s^j}, 10^{-9}\rangle, \langle C_{3,i}^{s^j}, 10^{-12}\rangle)$ representing the task execution behavior with s^j.

Scenarios and faults

By considering non-faulty conditions i.e., fault never happening, there exist the so called *no-fault* scenario. In case of *no-fault* the task execute in its normal way (thus no faults happening) with $c_i(t)$ the execution time of the task at time t. The execution time variability comes from functional and systemic effects from the normal task behavior. From *no-fault*, it exist the pWCET estimate $\mathcal{C}_i^{no-fault}$ as the worst-case for *no-fault* only.

It could also exist the *all-fault* scenario which assumes that faults manifest at each task instance. In case of *all-fault* the task executes always under the most critical faulty condition. Traces of measurements under the *all-fault* scenario would have execution times, each impacted by the presence of faults. $\mathcal{C}_i^{all-fault}$ is the pWCET estimate for the scenario. The execution times under *all-fault* are $c(t) + \delta(t)$, with $\delta(t)$ the fault penalty model applied for *all-fault*. *all-fault* is an extreme scenario, virtually impossible to happen, and extremely pessimistic; it has been applied in [194], but it is here improved with more realistic, still pessimistic, scenarios.

In between these two extreme scenarios, there exists a set of possible faulty conditions where faults are characterized according to precise fault models (δ and fault frequency). Consequently, there exist different possible fault scenarios, each abstracted with $\mathcal{C}_i^{fault-k}$ and related to the fault condition modeled.

In this work, the focus is on two faulty scenarios *fault-1* and *fault-2*, the non-faulty scenario and the *all-fault* scenario:

$$S \overset{def}{=} \{no-fault, fault-1, fault-2, all-fault\}. \tag{4.38}$$

no-fault is the baseline scenario where no faults manifests; *fault-1* includes some faults and some fault effects, while *fault-2* models some other faults. It is assumed that *fault-2* dominates *fault-1* since the faults modeled by it have larger effects or more frequent manifestation that those in *fault-1*. *all-fault* is the scenario where faults manifests at each task execution; it is the worst scenario among the four and dominates all the others.

Safety levels and task modeling

The different execution scenarios affect the safety of pWCET estimations. To recall, a pWCET is a safe worst-case estimate if it upper bounds any task execution time with every possible scenarios. With pWCETs from multiple scenarios and dominance between scenarios, it is possible defining levels of safety in the worst-case probabilistic representations which resembles to the confidence of WCET thresholds.

***no-fault* scenario:** The *no-fault* scenario with $\mathcal{C}_i^{no-fault}$ as the pWCET; $\mathcal{C}_i^{no-fault}$ is a worst-case only in case of absence of faults; it cannot be considered a safe worst-case estimation for the task because the fault effects are not included. $\mathcal{C}_i^{no-fault}$ is not safe enough for characterizing the task worst-case behavior for S. $\mathcal{C}_i^{no-fault}$ is the low safe representation, equivalently LO-safe worst-case representation, $\mathcal{C}_i^{no-fault} \equiv \mathcal{C}_i^{\mathrm{LO}}$. From $\mathcal{C}_i^{no-fault}$ it would be possible to extract WCET thresholds $\langle C_i(\textsc{lo}), p\rangle$ that is named LO-safe thresholds; $C(\textsc{lo})$ is a LO-safe upper bound to the task behavior with confidence probability p.

***fault-1* scenario:** The *fault-1* scenario embeds the fault effects form the faults models in *fault-1*. *fault-1* dominates *no-fault* conditions, hence the pWCET $\mathcal{C}_i^{fault-1}$ could be considered a relatively safe estimation of the worst-case task behavior since it upper bounds *no-fault* and of course *fault-1* conditions. $\mathcal{C}_i^{fault-1}$ is a middle safe representation (MI-safe worst-case representation) for S, $\mathcal{C}_i^{fault-1} \equiv \mathcal{C}_i^{\mathrm{MI}}$. From $\mathcal{C}_i^{fault-1}$ it would be possible to extract $\langle C_i(\textsc{mi}), p\rangle$, each named MI-safe WCET threshold; $C(\textsc{mi})$ is the MI-safe upper bound to the task behavior at probability p.

***all-fault* scenario:** The *all-fault* scenario embeds all the fault effects at each task execution. This is the scenario which dominates all the others since it considers faults always manifesting, at each job. $\mathcal{C}_i^{all-fault}$

would be the most pessimistic pWCET and the most safe. *all-fault* is unrealistic because it consider impossible to happen conditions. It can be neglected and replaced with more realistic, still safe, scenarios.

***fault-2* scenario:** The *fault-2* scenario embeds the faults present in *fault-2*; it is a scenario which dominates non-faulty conditions and *fault-1* conditions. Hence, its pWCET $\mathcal{C}_i^{fault-2}$ could be considered a safe estimation of the worst-case task behavior since it upper bounds every realistic execution conditions. $\mathcal{C}_i^{fault-2}$ is a high safe representation, HI-safe worst-case representation, with S, unrealistic/too pessimistic cases excluded, $\mathcal{C}_i^{fault-2} \equiv \mathcal{C}_i^{\text{HI}}$. From $\mathcal{C}_i^{fault-2}$ it would be possible to extract $\langle C_i(\text{HI}), p\rangle$, each named HI-safe WCET threshold with associated the probability/confidence p.

From the difference between *fault-1*, *fault-2* and *no-fault* execution conditions, it is $\mathcal{C}_i^{\text{HI}}$ greater than or equal to $\mathcal{C}_i^{\text{MI}}$ which is greater than or equal to $\mathcal{C}_i^{\text{LO}}$, with the partial ordering between distributions defined as in [48]. Also $C_i(\text{HI}) \geq C_i(\text{MI}) \geq C_i(\text{LO})$ as they are both taken at the same probability p from respectively $\mathcal{C}_i^{\text{HI}}$ and $\mathcal{C}_i^{\text{LO}}$. In order to quantify the differences, the relationship between between faulty and non-faulty conditions have to be known as well as the fault impact models for the task execution time.

The modeling can apply to multiple faults; with specific fault and fault models, the safe levels will be instantiated case by case.

Figure 4.9 proposes three pWCETs, each resulting from a different scenarios $\{\text{LO}, \text{MI}, \text{HI}\}$. Comparison between the three distributions outlines the safety levels of pWCETs together with their flexibility in characterizing the worst-case behavior of the tasks under different execution conditions.

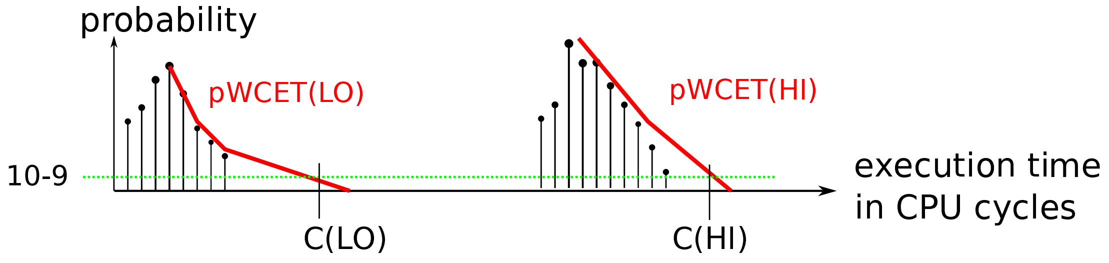


Figure 4.9: Different execution scenarios with different worst-case estimates; the pWCETs are represented as pdf.

To note that the probabilistic models here applied to faults can be extended to more abstract representations, including scenarios and task with different execution modes which are not strictly related to fault effects.

With three safety levels $\{\text{LO}, \text{MI}, \text{HI}\}$ it is:

$$\tau_i \overset{def}{=} ([\overline{\mathcal{C}}_i, \langle \overline{C}_i, 10^{-9}\rangle, (\hat{C}_i^{\text{LO}}, \hat{C}_i^{\text{MI}}, \hat{C}_i^{\text{HI}})], T_i, D_i), \tag{4.39}$$

where $\overline{\mathcal{C}}_i = (\mathcal{C}_i^{\text{LO}}, \mathcal{C}_i^{\text{MI}}, \mathcal{C}_i^{\text{HI}})$ is the inter-scenario representation at $p = 10^{-9}$; $p = 10^{-9}$ is taken because in certification documents is considered a low-enough probability for guaranteeing safety-critical systems.

With the intra-scenario representation is is $\hat{C}_i^{\text{LO}} = (\langle C_{1,i}(\text{LO}), 10^{-3}\rangle, \langle C_{2,i}(\text{LO}), 10^{-6}\rangle, \langle C_{3,i}(\text{LO}), 10^{-9}\rangle)$ for the LO-safe, $\hat{C}_i^{\text{MI}} = (\langle C_{1,i}(\text{MI}), 10^{-3}\rangle, \langle C_{2,i}(\text{MI}), 10^{-6}\rangle, \langle C_{3,i}(\text{MI}), 10^{-9}\rangle)$ for the MI-safe scenario and $\hat{C}_i^{\text{HI}} = (\langle C_{1,i}(\text{HI}), 10^{-3}\rangle, \langle C_{2,i}(\text{HI}), 10^{-6}\rangle, \langle C_{3,i}(\text{HI}), 10^{-9}\rangle)$ for the HI-safe scenario. The partial ordering between the WCET thresholds such that at the same probability $C_{k,i}(\text{LO}) \leq C_{k,i}(\text{MI}) \leq C_{k,i}(\text{HI})$ is guaranteed with the dominance within S.

Generalizing, there could exist larger sets of pWCETs $(\mathcal{C}_i^{\text{LO}}, \mathcal{C}_i^{1}, \ldots, \mathcal{C}_i^{\text{HI}})$ and larger sets of WCET thresholds $(C_i(\text{LO}), C_i(1), \ldots, C_i(\text{HI}))$ for a task. The intermediate safety levels would model execution conditions in between the worst-case and the *no-fault* case with different faults and fault impacts modeled.

The task model from Equation (4.39) is asserting that depending on the conditions for the timing analysis applied it is possible to have more or less guarantees on the pWCET and the WCET thresholds estimates. By considering all the possibilities, safe worst-case models are also flexible in representing the variability of the task/system behavior due to the changing conditions, including faults.

With the safety level notation $\{\text{LO}, \text{MI}, \text{HI}\}$, the set of scenarios S in Equation (4.38) becomes:

$$S = \{\text{LO}, \text{MI}, \text{HI}\}.$$

Probabilistic sensitivity analysis with faults

The sensitivity analysis studies how uncertainty in the output of a system can be appointed to the system inputs. When it comes to real-time systems, sensitivity analysis is coupled with schedulability analysis. It quantifies how much a task wort-case execution time must be reduced to achieve feasibility or must be increased in case of a performance improvements.

It is applied the sensitivity analysis with probabilities and safety levels on the pC-Space (C-Space+ in the paper referenced). With respect to faults, the sensitivity analysis allows defining metrics for i) quantifying their effects on schedulability and ii) quantifying the maximum safety level it is possible guaranteeing for a specific system configuration/scenario. With the probabilities, the sensitivity analysis quantifies the confidence cost of WCET thresholds changes.

- Through the sensitivity analysis there are quantified the cost of changes of task configurations, e.g. from non-schedulability to schedulability. There are defined two metrics Δ_{sl} and Δ_p, where Δ_{sl} quantifies the variation of safety level due to the task set configuration change and Δ_p quantifies the probability variation due to the task configuration change.

$$sl(\cdot)\ \{\text{LO}, \text{MI}, \text{HI}\} \rightarrow \mathcal{N}\ : sl(\text{LO}) \overset{def}{=} 1, sl(\text{MI}) \overset{def}{=} 2, sl(\text{HI}) \overset{def}{=} 3 \tag{4.40}$$

  For a task τ_i, the safety level change from a beginning WCET threshold C_i^{begin} to a final WCET threshold C_i^{final} it is $\Delta_{sl}(C_i) \overset{def}{=} sl(C_i^{final}) - sl(C_i^{begin})$. $\Delta_{sl} < 0$ would mean safety level reduction for τ_i, $\Delta_{sl} > 0$ increase of safety level, $\Delta_{sl} = 0$ the safety level does not change.
  For a point $\overline{c}$, it is $sl(\overline{c}) \overset{def}{=} (sl(C_1), sl(C_2), \ldots, sl(C_n))$, and Δ_{sl} from $\overline{c}^{begin}$ to $\overline{c}^{final}$ is $\Delta_{sl}(\overline{c}^{final}, \overline{c}^{begin}) \overset{def}{=} (sl(C_1^{final}) - sl(C_1^{begin}), \ldots, sl(C_n^{final}) - sl(C_n^{begin}))$.
  $p(C_i)$ is the probability associated to the WCET threshold C_i; $p(\overline{c})$ can be defined as in [194] where the safety level has to remain the same in order to combine the WCET thresholds. Within the same safety level, the probability change from $\overline{c}^{begin}$ to $\overline{c}^{final}$ is $\Delta_p(\overline{c}^{final}, \overline{c}^{begin}) \overset{def}{=} (p(C_1^{final}) - p(C_1^{final}), \ldots, p(C_n^{final}) - p(C_n^{final}))$.

- Through the sensitivity analysis it is possible defining policies to change system configurations, e.g. from non-schedulability to schedulability. For example it is possible to define:
  policy1: policy1 minimizes $\Delta_{sl}(\overline{c}^{final}, \overline{c}^{begin})$. policy1 plays with safety levels to reduce the safety level modifications. As an example, it is possible to change the system configuration seeking to make a non-schedulable configuration schedulable; policy1 can be applied in order to find $\overline{c}^{sched}$ such that $\Delta_{sl}(\overline{c}^{sched}, \overline{c}^{non-sched})$ is minimized. $\overline{c}^{sched}$ is the final schedulable point inside the feasibility region and $\overline{c}^{non-sched}$ is the starting configuration which is non-schedulable.
  policy1: policy2 minimizes $\Delta_p(\overline{c}^{final}, \overline{c}^{begin})$. policy2 makes the configuration change at fixed safety level. As an example, looking for making schedulable a non-schedulable point $\overline{c}^{non-sched}$ without changing the safety level, policy2 would minimize the probability increase (confidence reduction) due to the change, $\Delta_p(\overline{c}^{sched}, \overline{c}^{non-sched})$.

With metrics and change policies, the sensitivity analysis can enforce safety-schedulability (performance-predictability) trade-offs. For example, to guarantee all the timing behavior of the systems (including the worst faulty conditions), tasks would require large amount of computational resource which could jeopardize system schedulability. In order to make the system schedulable, the timing analysis has to make some concessions. For example, reducing the safety level of the WCET thresholds or increasing their probability (reducing their confidence).

The sensitivity analysis for studying real-time system schedulability provides useful feedback to system design, and will be further developed in future contributions.

4.2.5 Probabilistic scheduling for mixed criticality

In [165], with Prof. Zhishan Guo and Prof. Kecheng Yang, a first attempt in applying probabilistic models into mixed critical schedulers. In this work the probability is leveraged into scheduling decisions. In addition to the existing mixed criticality task model, this work introduces a new parameter to each task that represents the distribution information about its WCET. The aim is to provide schedulability analysis to instances with this additional probability information, with respect to the given safety certification requirement of the whole system, which is the permitted system failure probability per hour.

Considering the scheduling of dual-criticality task systems upon preemptive single-processor platforms. As stated above, dual-criticality tasks are traditionally characterized with two WCET estimations – a LO-WCET and a larger HI-WCET. The contributions are as follows:

- It is proposed a supplement to current MC task models: an additional parameter for each HI-criticality task, denoting the probability of no job of this task exceeding its LO-WCET within an hour of execution.
- It is further generalized the notion of system behavior by allowing for the specification of a permitted system failure probability per hour, denoting an upper bound on the probability that the system may fail to meet its timing constraints during any hour of running.
- It is derived a novel scheduling algorithm (and an associated sufficient schedulability test) for a given MC task set and an allowed system failure probability. Seeking to schedule the system such that the probability of failing to meet timing constraints during runtime is guaranteed to be no larger than the specified allowed system failure probability.

The algorithm, in the two criticality level case, requires just one probabilistic parameter per task – the probability that the actual execution requirement will exceed the specified LO-WCET in an hour. The scheduling algorithm is novel in that it is the first MC scheduling algorithm that makes scheduling decisions (e.g. when to trigger a mode switch) based not only on release dates, deadlines, and WCETs, but also on the probabilities drawn from probabilistic timing analysis tools (see e.g. [153, 156], [77]).

In the model, an allowed system failure probability F_S is specified. It describes the permitted probability of the system failing to meet timing constraints during one hour of execution[1]. F_S may be very close to zero, e.g. 10^{-12} for some safety critical avionics functionalities.

A failure probability parameter f_i can be added to the HI-criticality tasks. f_i denotes the probability that the actual execution requirement of any job of a HI-criticality task τ_i exceeds $c_i(\text{LO})$ (but still below $c_i(\text{HI})$) in one hour (i.e., the adequate long time interval assumed). f_i depends on a failure distribution $F_i(t)$ that describes the task τ_i probability of failure (at least) up to and including time t. Since $F_i(t)$ would refer to time (interval) and to task execution, it is going to be the one computed for one hour interval or any another interval. Thus, f_i is directly derived from $F_{\mathcal{C}_i}$.

Thus a HI-criticality task is represented in the model with four parameters: $\tau_i = ([c_i(\text{LO}), c_i(\text{HI})], f_i, p_i, \chi_i)$; LO-criticality tasks continue to be represented with three parameters as before. This enhanced model is essentially asserting, for each HI-criticality task τ_i, within a time interval of one hour, no job of τ_i has an execution greater than $c_i(\text{HI})$ and the probability of any job of τ_i has an execution greater than $c_i(\text{LO})$ is $f_i > 0$ — expect f_i to be a very small value. In this work it is assumed $c_i(\text{HI})$ the deterministic WCET, $\langle c_i(\text{HI}), 0\rangle$, while $\langle c_i(\text{LO}), f_i > 0\rangle$ the probabilistic WCET with $c_i(\text{LO}) \leq c_i(\text{HI})$. Normally it is not guaranteed higher assurance for LO-criticality tasks (than HI-criticality ones), and thus only $c_i(\text{LO})$ are adopted for them.

Definition 4.2.1 (MC task instance) *A MC task instance I is composed of a MC task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ and a system failure requirement $F_S \in (0, 1)$. (Although F_S may be arbitrarily close to 0, $F_S = 0$ is not an acceptable value — "nothing is impossible.")*

Let $n_{\text{HI}} \leq n$ denote the number of HI-criticality tasks in τ. It is assumed that the tasks are indexed such that the HI-criticality ones have lower indices; i.e., the HI-criticality tasks are indexed $1, 2, \ldots, n_{\text{HI}}$.

Seeking to determine the *probabilistic schedulability* of any given MC task instance:

Definition 4.2.2 (Probabilistic schedulability) *A MC task set is strongly probabilistic schedulable by a scheduling strategy if it possesses the property that upon execution, the probability of missing any deadline is less than F_S. It is weakly probabilistic schedulable if the probability of missing any HI-criticality deadline is less than F_S. (In either case, all deadlines are met during system runs where no job exceeds its LO-WCET.)*

That is, if a schedulability test returns strongly schedulable then all jobs meet their deadlines with a probability of no less than $1 - F_S$, while weakly schedulable only guarantees (with probability no less than $1 - F_S$ that) HI-criticality jobs meet their deadlines. Moreover, similar to all MC works, for either strongly or weakly probabilistic schedulable, all deadlines are met when all jobs finish upon executing their LO-WCETs. Again, F_S comes from the natural need of some system certifications, while f_i is the additional information for each task that has to be derived from WCET estimations to achieve such probabilistic certification levels.

In the model, the failure probability per hour of each task f_i represents the probability of any job of the task τ_i exceeding its LO-WCET. Thus dependences between tasks and task executions could have a strong impact on f_i.

In [42] it has been shown that neither probabilistic dependence among random variables nor statistical dependence of data implies the loss of independence between tasks' pWCETs or WCET estimates. The WCET is

[1]Failure probability are easily referable to failure rate, being careful at considering the failure rate as a probability.

an upper-bound to any execution time, and it embeds all the dependence effects. This makes the important consequence on the independence between WCETs: jobs and tasks modeled with WCETs are independent because WCETs already embed dependence effects. In the MC study, the LO-WCET may come from consideration of the execution time rather than of the WCET. Although both execution bounds (LO-WCET, HI-WCET) are so far called worst-case execution time estimations, the LO-WCET may also serve as an execution time upper-bound, where dependence between tasks and within tasks needs to be more carefully accounted for.

Each MC task may generate an unbounded number of jobs. Since jobs generated from the same task set typically represent execution of the same piece of code, the failure probability f_i of a task τ_i represents the likelihood that the required execution time of any job generated within an hour by τ_i will exceed $c_i(\text{LO})$. In [180, 182] it has been showed that real safety-critical embedded systems have natural variability on the task execution time, thus it is reasonable assume independence or extreme independence between jobs.

Concerning task dependencies, it is possible to cope with the dependence by specifying the task pairwise dependence model. Assuming it is given a list of pairs (τ_i, τ_j) indicating that (WC)ETs of these two tasks may be dependent on each other. It means that the probability of them both exceeding their LO-WCET is no longer the product of their individual probabilities. By knowing $P(\mathcal{C}_i > c_i(\text{LO}), \mathcal{C}_j > c_j(\text{LO}))$ it is possible to model (τ_i, τ_j) dependence including execution time task dependencies in the framework. It is however reasonable to assume that many (or most) task pairs do not have such dependencies to each other (although at the execution time level), since the limited impact of one task to another in a mixed critical partitioned system. Furthermore, it is worthy to note that execution times are observed with other tasks executing in parallel, thus the execution time measuring embeds already dependence effects from other tasks. In future work better task dependence modeling at runtime will be explained.

To resume, dependence between jobs of the same task and between tasks are covered by the model.

The notion of additional utilization cost, defined below, helps quantify the capacity that must be provisioned under HI-criticality mode.

Definition 4.2.3 (Additional utilization cost) *The additional utilization cost of* HI-*criticality task* τ_i *is given by*

$$\delta_i = (c_i(\text{HI}) - c_i(\text{LO}))/p_i. \tag{4.41}$$

Since it is considered EDF schedulability instead of fixed priority, whether and how likely system utilization may exceed 1 are under investigation: (i) if it is extremely unlikely that the total HI-criticality utilization exceeds 1 (weakly probabilistic schedulable), it is possible to assert a system that is infeasible in traditional MC model to be probabilistic feasible; (ii) if it is extremely unlikely that total system utilization exceeds 1 (strongly probabilistic schedulable), it could be possible to decide not to drop any LO-criticality task even if some HI-criticality tasks accidentally suffer from failures (that they require more execution time than expected).

In [165], the example has shown an infeasible task set (under traditional MC scheduling) being weakly probabilistic schedulable under the model. As seen from the definitions, existing mixed criticality systems are often analyzed under two modes – the HI mode and the LO mode, and mode switch is triggered when any HI-criticality job exceeds its LO-WCET without signaling finishing. Upon such a mode switch, deadlines of all LO-criticality jobs will no longer be guaranteed. A natural questions arises – is such sacrifice (dropping all LO-criticality jobs) necessary whenever a HI-criticality job requires execution for more than its LO-WCET?

The LFF-clustering algorithm

In this subsection, it is presented the strategy proposed for scheduling independent preemptive MC task instances, by combining HI-criticality tasks into clusters intelligently, and provide a sufficient schedulability test for it. Consider what done in [165], we: (i) conceptually combined the HI-criticality tasks τ_1 and τ_2 into a single cluster, provisioning an additional server into the system to accommodate their possible occasional HI-mode behaviors (execution beyond their LO-WCETs); and (ii) performed two EDF schedulability tests: one considering only HI-criticality tasks (with LO-WCETs) and the server, and the other also considering the LO-criticality task (τ_3). Since both tests succeed, it is declared strongly probabilistic schedulable for the given instance; it could be declared weakly probabilistic schedulable if the second schedulability test had failed while the first one succeeded.

The technique that was illustrated in the example [165] forms the basis of the scheduling strategy that is derived in in this section. To obtain a good upper bound to HI-criticality utilization of the system, tasks are combine into clusters – suppose that the n_{HI} HI-criticality tasks have been partitioned into M clusters $G_1, G_2, ..., G_M$, and let $y_i \in \{1, 2, ..., M\}$ denote to which cluster (number) task τ_i is assigned.

Definition 4.2.4 (Failure probability of a cluster) *Failure of a cluster G_m is defined as job generated by more than one tasks in a single cluster exceeding their* LO-*WCETs within an hour. The probability of a failure occurring in cluster m is denoted as g_m and is given by*

$$g_m \overset{def}{=} 1 - \prod_{i|y_i=m} (1-f_i) - \sum_{j|y_j=m} f_j \frac{\prod_{i|y_i=m}(1-f_i)}{1-f_j}, \tag{4.42}$$

where the second term of right hand side is the probability of no task (in the cluster) exceeding its LO-*WCET, and the last term represents the probability of exact one of the tasks exceeding its* LO-*WCET in an hour.*

Lemma 4.2.5 *If $g_m < F_S/M$ holds for any cluster G_m, then the probability of having no failure in any cluster is greater than $(1 - F_S)$.*

Lemma 4.2.5 provides a safe failure threshold F_S/M for each cluster; i.e., the rule for forming clusters is $g_m < F_S/M$, where M is the current number of clusters.

The additional utilization cost of a cluster G_m is defined to be equal to the additional utilization cost (δ_i) of the task within the cluster with the largest δ_i value; i.e.,

$$\Delta_m \overset{\text{def}}{=} \max_{i|\tau_i \in G_m} \delta_i. \tag{4.43}$$

The total system additional utilization cost is given by the sum of additional utilization cost of all M clusters;

$$\Delta \overset{\text{def}}{=} \sum_{m=1}^{M} \Delta_m. \tag{4.44}$$

A critical observation is that, if a task τ_i with additional utilization cost δ_i has been assigned to a cluster, assigning any other task τ_j with $\delta_j \leq \delta_i$ to the cluster will not increase the additional utilization cost. To minimize the total additional utilization cost of the entire task set, it is therefore greedily expanded existing clusters with tasks of larger additional utilization cost while ensuring that the relationship $g_m < F_S/M$ continues to hold, leading to the Largest Fit First (LFF)-Clustering algorithm.

LLF Algorithm can be found in [165]. This algorithm greedily expands each existing cluster with unassigned tasks while the condition $g_m < F_S/M$ holds; while a new cluster is created only if it is not possible to assign a task to any current cluster without violating the condition ($g_m < F_S/M$).

- **Remark 1.** Similar to what has been done in [48] and [109], it would be possible to achieve a precise distribution to the total utilization of all tasks by applying the convolution operation '$\otimes$', which results in an exponential ($O(2^{n_{\text{HI}}})$, to be precise) running time. The sufficient schedulability test based on the LFF-Clustering algorithm runs in $O(n_{\text{HI}}^2)$ time, where n_{HI} is the number of HI-criticality tasks.
- **Remark 2.** In the case that all tasks share the same f_i value, the schedulability test based on LFF-Clustering becomes necessary and sufficient.

runtime strategy. During execution, a HI-criticality server τ_s with utilization Δ and a period of 1 tick is added to the task system. It is needed the server period as 1 tick because the mechanism and the analysis will not work if there is release or deadline within a server period. At any time instant that the server is executing, the active[1] HI-criticality job, if any, with earliest deadline is executed; if there is no such job, the current job of the server is dropped[2]. All jobs including the server are scheduled and executed in EDF order, and a job is dropped at its deadline if it is not completed by then.

Note that although it is introduced a server task with period of 1, preemption does not necessarily happen that often. The goal of the sever task with utilization Δ is to preserve a "bandwidth" of at least Δ for HI-criticality jobs if the HI-criticality ready queue is not empty. There are three situations to be considered:

Situation 1: The job with the earliest deadline is a HI-criticality job. In this situation, it is executed the HI-criticality job with 100% processor share, and no more preemption is incurred by the server.

Situation 2: The job with the earliest deadline is a LO-criticality job and the HI-criticality ready queue is empty. In this situation, it is executed execute the LO-criticality job with 100% processor share, and hence there is no additional preemption in this situation either.

[1]A job is active if it is released and incomplete at that time instant.

[2]Since an integer model of time is assumed (i.e., all task periods are integers and all job arrivals occur at integer instants in time), and the server has a period of 1, it is safe to drop the current job of the server if there is no active HI-criticality jobs since there can be no HI-criticality job releases in the current period of the server.

Situation 3: The job with the earliest deadline is a LO-criticality job and the HI-criticality ready queue is not empty. In this situation, the aim is to preserve a processor share of Δ for HI- criticality jobs and to execute the LO-criticality ones with the rest $1 - \Delta$ of the processor capacity. Therefore, the server creates preemptions every time unit.

That is, only in Situation 3, that the algorithm "introduces" extra preemptions due to the server scheme, and normal EDF scheduling is applied in other cases. One may claim that such server allocation scheme may results in more preemptions than the approaches where the server capacity is only used for overruns. Actually this is because that the goal here is trying not to drop LO-criticality tasks even when a few HI-criticality ones exceed their LO-WCETs. Thus, in order to guarantee HI-deadline being met always, certain use of the server even when no HI-criticality behavior is detected – simply taking "precautions". Alternative way such as assigning HI-criticality jobs virtual deadlines may lead to fewer preemptions, at a cost of losing the performance of schedulability ratio (see experimental comparisons).

There are used servers to implement the algorithms and prove the possibility of proficiently apply failure probability to both MC modeling and MC scheduling. In future work, the server period assumption of 1 unit of time will be released by applying adaptivity to resource reservation [185, 206]. With the analysis of the deadline and task periods it will be possible to implement realistic servers which adapt their period and budget to the MC-scheduler needs, while leaving the system predictable at any time interval. Such adaptive behavior will not introduce any overhead, and mostly will allow not to miss task deadline.

It is evident that for strongly probabilistic schedulable (i.e., to ensure that the probability of missing any deadline is no larger than the specified system failure probability F_S – see Definition 4.2.2), it is (necessary and) sufficient that $\left(\sum_{i=1}^{n} c_i(\text{LO})/p_i + \Delta\right)$ must be no larger than the capacity of the processor (which is 1).

For weakly probabilistic schedulable (i.e., to ensure that the probability of missing any HI-criticality deadline is no larger than F_S – again, see Definition 4.2.2), it is necessary that $\left(\sum_{i|\chi_i=\text{HI}} c_i(\text{LO})/p_i + \Delta\right)$ must be no larger than 1 as well. The following theorem helps establish a sufficient condition for ensuring weakly probabilistic schedulable:

Theorem 4.2.6 *If no job exceeds its LO-WCET, then no deadline is missed if*

$$\Delta \cdot (1 - \sum_{i|\chi_i=\text{HI}} \frac{c_i(\text{LO})}{p_i}) + \sum_{i=1}^{n} \frac{c_i(\text{LO})}{p_i} \leq 1. \tag{4.45}$$

The complete Theorem proof can be found in [165].

Theorem 4.2.6 yields the schedulability test pMC, while Theorem 4.2.7 below establishes its correctness.

Algorithm 1 Algorithm LFF-Clustering

```
Input: $F_S$, $\{f_i\}_{i=1}^{n_{\text{HI}}}$, $\{\delta_i\}_{i=1}^{n_{\text{HI}}}$
Output: maximum total additional utilization cost $\Delta$
Sort the tasks in non-increasing order of $\delta_i$;
$m \leftarrow 1$, $M \leftarrow n_{\text{HI}}$, $y_i \leftarrow 0$ for $i = 1, ..., n$;
while $\prod_{i=1}^{n_{\text{HI}}} y_i = 0$ (an unassigned task exists) do
    $\Delta_m \leftarrow 0$ (additional utilization of each cluster);
    for $i = 1$ to $n_{\text{HI}}$ do
        if $y_i > 0$: continue;
        $y_i \leftarrow m$, $M \leftarrow M - 1$;
        if $g_m \geq F_S/M$ :  $y_i \leftarrow 0$, $M \leftarrow M + 1$;
    $\Delta_m \leftarrow \max_{i|y_i=m} \delta_i$;
    $m \leftarrow m + 1$;
return $\sum_{m=1}^{M} \Delta_M$
```

Theorem 4.2.7 *The schedulability test pMC is sufficient in the following sense:*

- *if it returns* ***strongly probabilistic schedulable****, the probability of any task missing its deadline is no greater than F_S; and*
- *if it returns* ***weakly probabilistic schedulable****, the probability of any HI-criticality task missing its deadline is no greater than F_S, and no deadline is missed when all jobs finish upon execution of their LO-WCETs.*

Proof to the lemma and the theorem are available in [165].

The schedulability test pMC returns strongly probabilistic schedulable if it is possible to schedule the system such that the probability of missing any deadline is at most the specified threshold F_S, or weakly probabilistic schedulable if it is possible to schedule the system such that the probability of missing any HI-criticality deadline is at most F_S. It will then use EDF to schedule and execute the task set with LO-WCETs and the additional server task $\tau_s = \{\Delta, 1, \text{HI}\}$.

In the case that the schedulability test pMC returns unknown, it is not possible to schedule the system using the proposed probabilistic analysis technique. Normally it is either that there is set a too high safety requirement to the system; i.e., the threshold F_S is too small, or the WCET estimations are not precise enough for HI-criticality tasks; i.e., the f_i's are not small enough comparing to F_S (and n_{HI}), and/or the $c_i(\text{LO})$'s are still not differentiable enough with respect to $c_i(\text{HI})$'s.

Experimental results with randomly generated task are available in [165].

4.2.6 Formal methods and mixed criticality

The PhD thesis of Jasdeep Singh has the objective to investigate and apply formal methods to the schedulability of pRTSs. Its original effort has been on classical schedulability and probabilistic schedulability analysis [201, 203], `https://forge.onera.fr/projects/probscheduling`. More recent works are focusing on the MC problem for pRTSs and the use of formal methods for "smarter" scheduling decisions [202], `https://forge.onera.fr/projects/probscheduling`.

[202] is about describing and studying the execution of MC jobs in pRTSs with the use of formal methods. Two problems are distinguished, and approached. The first problem is: P1 – to quantify the probability for the system to execute in high criticality mode. It is proposed a formal model with associated model checking to extract the probability of the system tasking paths through high criticality modes. Formal methods are used to ensure that both the MC models and the MC analysis with probabilities are correct (validated).
The second problem is: P2 – to control the schedulability of HI-criticality jobs based on their deadline miss probability. Here, it is reasoned a strategy to avoid dropping all the LO-criticality jobs in the system, once the system is in high criticality mode. The analysis which that is developed selectively decides which LO-criticality job to drop based on its effects on HI-criticality jobs. With that, it is possible to maximize the number of LO-criticality jobs which can execute with HI-criticality jobs without jeopardizing their timing constraints.

A real-time application Γ has m tasks $\Gamma = \{\tau_1, \tau_2, \ldots \tau_m\}$, $m \in \mathbb{N}^+$ where $\mathbb{N}^+$ is the set of positive natural numbers. A task τ_i is tuple $\tau_i = (\mathcal{C}_i, T_i, D_i, \chi_i)$ where $\mathcal{C}_i$ is the pWCET PDF, T_i is period, D_i is the deadline of the task; $D_i \leq T_i$. Both T_i and D_i are deterministic parameters – single valued. $\mathcal{C}_i$ can be continuous or discrete worst-case distribution. In the task model, χ_i identifies the task criticality level.

Since the tasks are periodic, each j-th instance of a task τ_i is a job J_{ij}. The job model is such that: $J_{ij} = (\mathcal{C}_i, a_{ij}, d_{ij}, p_{ij}, \chi_{ij})$, $j = 1, 2, ...n$. J_{ij} arrives at time $a_{ij} = (j-1) \cdot T_i$; d_{ij} is the job absolute deadline, $d_{ij} = a_{ij} + D_i$; p_{ij} is the job priority, 0 being the highest priority i.e., if $p_{ij} \leq p_{kr}$ then J_{ij} has higher priority than J_{kr}. In the job model, χ_{ij} identifies the job criticality level; it is assumed that $\chi_{ij} = \chi_i$. The hyperperiod, defined as the least common factor of the periods of all the tasks periods $lcm(T_i)$, $i = 1, 2, ..., m$ is the scope of the analysis. In the hyperperiod there are n jobs, with n_i jobs for each task τ_i, $n = \sum_{k=0}^{m} n_k$. The real-time application can be represented with the n jobs, $\Gamma = \{J_{ij}\}$ $i = 1, 2, ..., m; j = 1, 2, ..., n$.

In pRTSs, the worst-case response time of a job is a random variable, represented as $\mathcal{RT}$.

Definition 4.2.8 (Job probabilistic worst-case response time) *A probabilistic Worst-Case Response Time (pWCRT) for a job J_{ij}, denoted as $\mathcal{RT}_{ij}$, is a probabilistic distribution which upper bounds every possible job response time due to concurrent job interference and preemptions during its execution. It is represented with PDF as $f_{\mathcal{RT}_{ij}}(x)$, CDF as $F_{\mathcal{RT}_{ij}}(x)$ and ICDF as $F'_{\mathcal{RT}_{ij}}(x)$.*

Definition 4.2.9 (Job deadline miss probability) *The probability that a job executes until a time greater than or equal to the respective deadline is called deadline miss probability of the job. For a job J_{ij}, it is denoted as P_{ij}^{dm}:*

$$P_{ij}^{dm} = Pr(J_{ij} \text{ finishes executing after } d_{ij}) \overset{def}{=} \int_{d_{ij}}^{\infty} f_{\mathcal{RT}_{ij}} dx. \tag{4.46}$$

Assumptions. In a probabilistic framework, the schedulability analysis refers to the process to obtain the pWCRTs and deadline miss probabilities of the constituent jobs: a task set is schedulable if the deadline miss probability of each of its tasks is smaller than a certain required probability. The pWCRT can be a discrete or a continuous probability distribution. The framework proposed utilizes the pWCRT of each job in the system. This framework is valid irrespective of whether the input (pWCET) or the output (pWCRT) distributions of the analysis are discrete or continuous. Thanks to the worst-case assumption, pWCET of tasks are independent [42]. All the execution intereference during the job executions are considered in the schedulability analysis and are represented by their pWCRT. It is the probabilistic counterpart of the same assumption in the deterministic context. In this work, the scheduling follows the EDF preemptive policy which defines the job ordering by static job-wise priorities p_{ij}. It applies to single-core platforms, and it is assumed the tasks' jobs are suspended at the deadline.

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.

A HI-criticality job J_{ij}^{HI} is the tuple: $J_{ij}^{\text{HI}} \overset{def}{=} (\mathcal{C}_i, a_{ij}, d_{ij}, p_{ij}, l_{ij}, \chi_{ij})$. $\mathcal{C}_i$, a_{ij},p_{ij} and d_{ij} are as defined earlier. χ_{ij} is the job criticality level defined for a job J_{ij} [22] which can take two values at runtime: HI and LO; $\chi_{ij} = \{\text{HI}, \text{LO}\}$. $l_{ij} \leq D_i$ describes the threshold with which the job criticality mode is defined.

A LO-criticality job J_{kr}^{LO} is the tuple: $J_{kr}^{\text{LO}} \overset{def}{=} (\mathcal{C}_k, a_{kr}, d_{kr}, p_{kr}, \chi_{kr})$. χ_{kr} for a LO-criticality job can take only one value, $\chi_{kr} = \{\text{LO}\}$.

For the jobs, the criticality level of its task is inherited, e.g. for J_{ij} and J_{ik} have the same criticality level as of the task τ_i level. However, the actual criticality mode of the jobs can change at runtime depending on their scheduling.

The real-time application is formed from these tasks which is partitioned between their HI-criticality jobs and LO-criticality jobs. $\Gamma^{\text{HI}} = \{J_{ij}^{\text{HI}}\}$ is the set of high criticality jobs with n^{HI} number of HI-criticality jobs; $\Gamma^{\text{LO}} = \{J_{kr}^{\text{LO}}\}$ 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; $m^{\text{HI}} + m^{\text{LO}} = m$.

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 [22]. The following more generic and flexible definition of system criticality is proposed.

Definition 4.2.10 ((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.*

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 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 it is needed to use reliable probabilistic analysis tools to analyze such a system.

Criticality threshold. The job criticality mode is defined using a threshold $l_{ij} \leq D_i$ which applies to the job pWCRT. The probability P_{ij}^{HI} that a job J_{ij} executes in the high criticality mode is:

$$P_{ij}^{\text{HI}} \overset{def}{=} \int_{l_{ij}}^{\infty} f_{\mathcal{RT}_{ij}}(x)dx. \tag{4.47}$$

If the pWCRT is discrete as shown in Figure 5.12(b) with WCET as the maximum possible execution time, $P_{ij}^{\text{HI}} \overset{def}{=} \sum_{x \geq l_{ij}}^{WCET} f_{\mathcal{RT}_{ij}}(x)$.
The MC definitions and modeling proposed, is slightly different than the classical ones with multiple WCET thresholds [140], [165]. With the MC modeling via pWCRT, it is possible to distinguish the job behavior 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 proposed 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.

Discrete time Markov chain. It is assumed 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 modeled as a Discrete Time Markov Chain (DTMC). The choice of DTMC makes it possible to simply arrange the readily available probabilities from the schedulability analysis. System depiction as states replicates the switching of the real-time system between high the low criticality modes. The transitions between those states can be labeled with the probability of it being chosen. Moreover, DTMC allows modeling of a probabilistically distributed system subject to its mathematical foundation. This is unlike a linear system, like Petri net or automata, where discrete actions form a complex explorable tree. DTMC is subject to formal model checking in which path properties or the probability of reaching certain states can be formally checked. This is useful to know the probability of a path taken by the system. A DTMC M is defined as a set of states S and state transitions given by a Q-matrix Q, $M = (S, Q)$ [121]. In the following are given the basics to build DTMC for mode changes in MC pRTSs and the properties which are formally verified with PRISM Model Checker [87].

System criticality mode modeling. It is presented here the DTMC model for problem **P1**, $M^{crit} = (S^{crit}, Q^{crit})$;
To build M^{crit} it is required to consider only the HI-criticality jobs in this DTMC. It is because only HI-criticality jobs contribute to decide the system criticality. For each job $J^{\text{HI}}_{ij} \in \Gamma^{\text{HI}}$, the set of states $S_{ij} = \{J^{\text{HI}}_{ij}, HC_{ij}, LC_{ij}\}$ is defined. State HC_{ij} represents execution of J^{HI}_{ij} in high criticality mode ($[l_{ij}, \infty)$) and state LC_{ij} represents execution of J^{HI}_{ij} in low criticality mode ($[0, l_{ij})$). The state J^{HI}_{ij} 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^{crit} for the whole system is defined as an union of the sets S_{ij}, $\forall i, j$ such that job J_{ij} is a HI-criticality job: $S^{crit} \overset{def}{=} (\bigcup_{i,j} S_{ij}) : J_{ij} \in \Gamma^{\text{HI}}$. S^{crit} 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 DTMC does not represent the scheduling but only the criticality configurations. The set of states and transitions in M^{crit} is shown in Figure 4.10 for an example task set Γ. There are unidirectional transitions $J^{\text{HI}}_{ij} -> HC_{ij}$, $J^{\text{HI}}_{ij} -> LC_{ij}$, $HC_{ij} -> J_{ab}$, $LC_{ij} -> J_{ab}$; such that $p_{ab} > p_{ij}$ and the priority difference $|p_{ij} - p_{ab}|$ is minimum $\forall i, j, a, b$, $J_{ab}, J_{ij} \in \Gamma^{\text{HI}}$. The initial state is a J_{ab} such that p_{ab} is minimum and $J_{ab} \in \Gamma^{\text{HI}}$.
The state transitions are labeled such that the sum of the probabilities of all the outgoing transitions is equal to one. Each transition emanating from a state J_{ij} to HC_{ij} is labeled with the probability P^{HI}_{ij}. This implies, each transition emanating from a state J_{ij} to LC_{ij} is labeled with the probability $1 - P^{\text{HI}}_{ij}$. The transition matrix Q^{crit} defined as:

$$\left[\begin{array}{cccccccc}
 & J_{11} & HC_{11} & LC_{11} & \dots & J_{m^{\text{HI}}n^{\text{HI}}} & HC_{m^{\text{HI}}n^{\text{HI}}} & LC_{m^{\text{HI}}n^{\text{HI}}} \\
J_{11} & 0 & P_{11} & 1-P_{11} & \dots & 0 & 0 & 0 \\
HC_{11} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
LC_{11} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
J_{12} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
HC_{12} & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
LC_{12} & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
J_{m^{\text{HI}}n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & P_{m^{\text{HI}}n^{\text{HI}}} & 1-P_{m^{\text{HI}}n^{\text{HI}}} \\
HC_{m^{\text{HI}}n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
LC_{m^{\text{HI}}n^{\text{HI}}} & 0 & 0 & 0 & \dots & 0 & 0 & 0
\end{array}\right] \tag{4.48}$$

To refer to an element of the matrix, $Q^{crit}(State_r, State_c)$ 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_{11} to HC_{11} is referred as $Q_{DTMC}(J_{11}, HC_{11})$, equal to P_{11}.
The DTMC models construction is quite straightforward without involving any mathematical operations like convolution or upper-bounding of any kind. It is simply arranging the probabilities obtained from the schedulability analysis into formally verifiable DTMC structure. The safety of this construction comes from a safe schedulability analysis.

Criticality analysis

In this section, it is detailed the proposition to tackle with problem **P1**. With DTMC $M^{crit} = (S^{crit}, Q^{crit})$ Matrix (4.48), it is defined the meaning of system criticality from the criticality levels of the jobs. Then, it is quantified the probability of the system criticality by exploring M^{crit} with formal model checking. This analysis is named 'criticality analysis' (*crit*). The definition of system criticality level translates into paths within the DTMC taken by the system through certain states. Each path has a probability of being taken at runtime. This is a probability for the system entering the high criticality mode.

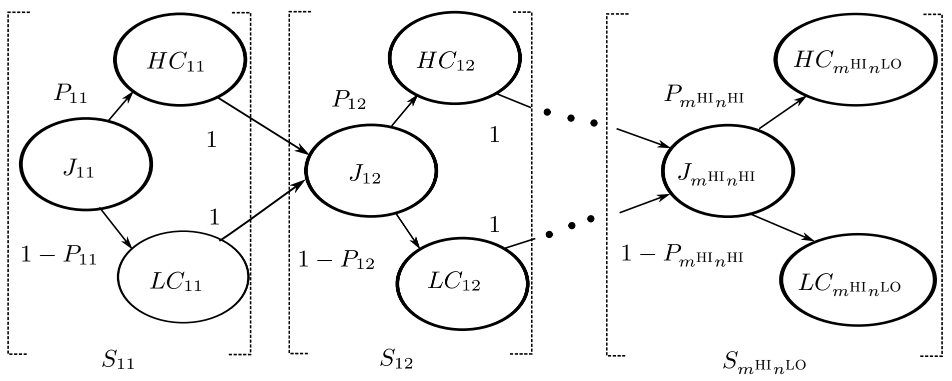


Figure 4.10: System DTMC model M^{crit}, assuming $J_{11}, J_{12}, \ldots, J_{m^{\text{HI}}n^{\text{HI}}} \in \Gamma^{\text{HI}}$ and such that $p_{11} \leq p_{12} \leq \ldots \leq p_{m^{\text{HI}}n^{\text{HI}}}$.

A path D in M^{crit} represents the trace of the jobs taking high or low criticality modes at runtime. It is an ordered set of states $D \overset{def}{=} [State_1, State_2, ...State_{n^{\text{HI}}}]$, such that $Q^{crit}(State_k, State_{k+1}) > 0$ 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 DTMC 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.

For M^{crit}, $2^{n^{\text{HI}}}$ are possible paths which start from the initial state. Examples of a path are: $D_1 = [J_{11}, HC_{11}, J_{12}, LC_{12} \ldots, LC_{m^{\text{HI}}n^{\text{HI}}}]$, $D_2 = [J_{11}, LC_{11}, J_{12}, HC_{12}, \ldots, HC_{m^{\text{HI}}n^{\text{HI}}}]$.

(k,n) system criticality. Here, it is recalled Definition 4.2.10 that system is said to be in high criticality mode if at least k^{HI} out of m^{HI} jobs enter high criticality mode. The system entered 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 numbers, which is left for future discussions. Now, using Definition 4.2.10 system entered 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 entered high criticality region is:

$$P^{kn} \overset{def}{=} Pr(D_1^{kn}) + \ldots + Pr(D_{q_k}^{kn}) = \sum_{q=0}^{q_k} Pr(D_q^{kn}). \tag{4.49}$$

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 jobs is in high criticality mode: $P^{1n} \overset{def}{=} Pr(D_1^{1n}) + \ldots + Pr(D_{2^n-1}^{1n}) = \sum_{q=1}^{2^n-1} Pr(D_q^{1n})$. Where D^{1n} are the paths taken through at least one high criticality states. (n,n) is to say that the system criticality level as high whenever all the HI-criticality jobs are in high criticality mode: $P^{nn} \overset{def}{=} Pr(D_1^{nn})$. Where D_1^{nn} is the path taken through all the high criticality states. This concludes the solution to problem **P1**.

This part focuses on the deadline miss probability of HI-criticality jobs. The problem **P2** can be stated as: *the deadline miss probability of a job J_{ij} should be less than or equal to certain 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. Solving **P2** is to define a MC scheduling algorithm that is able to improve the probability of deadline miss of jobs of choice.

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 [22]. Instead, the scheduling algorithms proposed 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 job 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. One such method is presented below.

Job strategy

The job strategy is the scheduling algorithm proposed 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 times depends on the execution of the previously executed jobs. The strategy to choose needs to ensure optimality as well as the safety of the resulting schedule. 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 is presented 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 whenever system enters high criticality mode [22].

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

$$\overline{J}^{over}(J_{ij}^{\text{HI}}) \overset{def}{=} \{J_{gh} : p_{gh} < p_{ij}, d_{gh} > a_{ij}, J_{gh} \in \Gamma^{\text{LO}}\}. \tag{4.50}$$

The jobs in this set directly impose a probabilistic delay/backlog in the execution of J_{ij}^{HI}, as depicted in the Figure 4.11 by jobs J_{rt}^{LO} and J_{kj}^{LO}. All the jobs in $\overline{J}^{over}(J_{ij}^{\text{HI}})$ impose an indirect backlog to J_{ij}^{HI}. The indirect backlog is passing a certain amount of backlog in their order of priority and thus indirectly to the job J_{ij}^{HI}. The term 'interference isolation' refers to the separation of J_{ij}^{HI} from the indirect backlog.

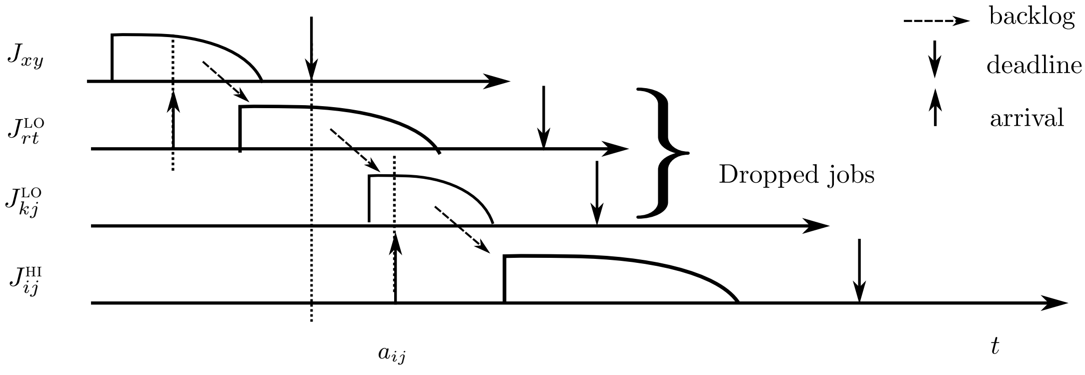


Figure 4.11: The backlog to the HI-criticality job J_{ij}^{HI} reduces to zero by dropping jobs J_{rt}^{LO} and J_{kj}^{LO}.

Lemma 4.2.11 (Backlog isolation) *The backlog for a* HI*-criticality job J_{ij}^{HI} reduces by the maximum amount if all the* LO*-criticality jobs in $\overline{J}^{over}(J_{ij}^{\text{HI}})$ from Equation (4.50) are dropped, given that the jobs are suspended at their respective deadlines.*

The lemma is proven in [202, 204].

The job-level scheduling strategy proposed reduces the probability of deadline miss of a HI-criticality job by dropping all the LO-criticality jobs in $\overline{J}^{over}(J_{ij}^{\text{HI}})$. Lemma 4.2.11 proves that dropping all those jobs ensures the maximum possible deadline miss probability reduction for J_{ij}^{HI}. Thus, the strategy is: 1) identifying $\overline{J}^{over}(J_{ij}^{\text{HI}})$, 2) dropping all the jobs in $\overline{J}^{over}(J_{ij}^{\text{HI}})$. Referring to the Figure 4.11, the backlog to the HI-criticality job J^{HI} minimizes by dropping jobs J_{rt}^{LO} and J_{kj}^{LO}: there is no affect from the job J_{xy} to the job J_{ij}^{HI} because J_{xy} suspends at the deadline. The HI-criticality job in observation still retains its own execution after dropping the jobs in the set $\overline{J}^{over}(J_{ij}^{\text{HI}})$, that is after removing maximum interference. The P_{ij}^{dm} obtained once dropping all the jobs in $\overline{J}^{over}(J_{ij}^{\text{HI}})$, is the best (minimum) deadline miss probability that can be achieved for J_{ij}^{HI}. If it is not enough to meet the constraint $P_{ij}^{dm,max}$, P_{ij}^{dm} is still larger than $P_{ij}^{dm,max}$, the problem for this job is unsolvable.

This work has been recently accepted to the MC workshop WMC at RTSS 2018 [204]. In it, explicative examples and a test case applied for validating the dropping algorithm. Future work is already under development in collaboration with Jasdeep Singh, with Prof. Giuseppe Lipari in Lille, and Prof. Laura Carnevali in Florence. With it, the effort is for releasing the constraining hypothesis of independence and taking into account multiple source of dependence between tasks.

4.2.7 Some conclusions on probabilistic approaches to mixed criticality

On the criticality of pWCET models. In [193] there are the basis for the most flexible probabilistic task model that accounts for all the modes and mode combinations. The complexity from such MC representations is not considered here, preferring at this stage to investigate the benefits of a detailed probabilistic model to task MC behaviors. A case study is in [193] for validating the probabilistic models and illustrating its effectiveness in modeling different conditions/criticality.
Such fine grained representation, coupled with probabilistic MC scheduler will allow taking more accurate decisions depending on the actual execution conditions. Future work will apply the representations into MC schedulability and MC probabilistic sensitivity analysis to better explore the flexibility of the probabilistic representations.

Probabilistic sensitivity analysis for mixed criticality. [191] defines probabilistic sensitivity analysis for MC problems. The notion of parameter β is just introduced, but its envisioned potential for the MC problem deserves more investigation.
Future works will exploit the use of β, the same for the whole task set or each task with a different β_i, to develop MC real-time systems. Also, there will be developed strategies to change resource or application parameters. The strategies are for better exploring the probabilistic regions, for defining more trade-offs, and propose changing strategies. All those notions will be applied to MC problems.

Open problems with probabilistic sensitivity analysis. [184] has been proposed to the open problems workshop RTSOPS at ECRTS 2018. The abstract is an attempt to show how sensitivity analysis can help with the intrinsic complexity of probabilistic [MC] scheduling. Exploring the abstract representations (pCspace or probabilistic (α, Δ)-space) can avoid applying complex analytic expression for the probabilistic schedulability; the trade-off definition and exploration are for simplifying the definition and development of MC real-time embedded systems.
Future works will develop probabilistic sensitivity analysis for probabilistic MC scheduling with the objective that remains to reduce the complexity of MC scheduling. Also, the trade-offs, parametrized with probability, will be explored with sensitivity analysis for smart/efficient MC scheduling decisions based on probabilities.

Fault-aware sensitivity analysis for probabilistic real-time systems. [194] is an example of faults and fault effects included into probabilistic task models, and then applied into schedulability analysis. The notion of safety can be seen as equivalent to criticality level, as it is used to distinguish execution scenarios different than the absolute worst-case. Probabilistic sensitivity analysis applies to evaluate the impact that faults have on the scheduling of real-time applications.
Future work is necessary for integrating fault effects into timing analysis approaches for task pWCET which is more realistic and less pessimistic. Also, schedulability analysis will profit of more accurate task models with faults, in order to study the impact that faults have, or could have, on scheduling decisions. Natural will be the application of the results in [194] to the MC problem.
In those future contributions, the modeling with faults will parameterize the execution scenarios, and criticality levels, with faults models. Instead, the sensitivity analysis will exploit such parameters and relate scheduling conditions to fault conditions.

Probabilistic scheduling for mixed criticality. [165] proposes a MC scheduling algorithm which mixes probabilistic task models for LO-criticality modes and deterministic WCET for HI-criticality modes. The algorithm apply probability information into scheduling decisions in order to reduce the pessimism and increase resource usage with respect to classical deterministic MC scheduling.
The solution proposed has limitations in the server implementation, applieddue to the need to react at each valuable time instant (arrival, deadline, period, etc.) in order to be effective. From that, the solution proposed with servers of period 1. In future work, the server period assumption of 1 unit of time will be released by applying adaptivity to resource reservation inspired by [185, 206]. With the analysis of the deadline and task periods, it would be possible to implement realistic servers which adapt their period and budget to the MC-scheduler needs, while leaving the system predictable at any time interval. Also, the sensitivity analysis will help releasing such constraints and extend the algorithm applicability to real implementations.

Formal methods and mixed criticality. Formal methods, with their solid mathematical background, have already proven to be able to model complex systems. They can be very helpful in representing pRTSs and in taking efficient scheduling decisions. Also, with formal methods, validation becomes much easier and provable with model checkers.
Once explored in detail, formal methods will be an alternative way to model MC scheduling. Scheduling decisions can be characterized and validated, with formal methods. Specifically, formal methods and probabilities could

help taking the decision that best cope with the actual execution conditions. Efficient resource usage can be enforced with probabilities and accurate task execution models.
Currently, Jasdeep Singh is investigating other formal methods to be applied to the MC problem. He is also enhancing already existing solutions [202, 204].

Chapter 5

Concluding remarks, lessons learned, and future perspectives on mixed criticality

This concluding chapter comments my contributions on MC, and outlines possible and necessary progresses for it. The research perspectives are mainly necessary for the challenges that multi-core and many-core implementations are posing and will continue to pose for real-time systems. There will be always considered both theoretical and industrial problems, and the solutions proposed will always get inspired from practical and academic experience.

MC is one of the real-time problems under heavy scrutiny from research groups and industry. It has been largely investigated with single-core platforms. Nowadays, the focus is more on multi-core implementations and the challenges from *partitioning and safe resource sharing.*

The MC problem affects task models: more flexible representations imposes with the goal of reducing pessimism. Tasks worst-case models parametrized with criticality is a good starting but the multiple execution conditions possible with multi-core and many-core have to be better represented with MC models than just with WCET thresholds. Also, the numerous source of interference have to be identified and embedded into MC task models in order to increase the knowledge of system dynamics, improve system models, and embed them into more fine grained task models.

The MC problem affects also schedulability analysis. The largely available resource with multi-core and many-core implementations require better management: functionality increase demands for efficiency and safety. More effective resource partitioning have to be defined, while partitioning tools have to be verified. Safe resource sharing have to be guaranteed with scheduling algorithms that are efficient.

As at today, I consider two main perspectives for the MC problem:

- The *academic perspective* takes the mixed criticality literally by studying how mixing together applications with different criticality levels. In it, criticality levels are combined and there are developed scheduling algorithms that are able to guarantee such mixture of tasks. The classical guarantee offered in MC is such that: all the HI-criticality tasks are assured execution first, and then the LO-criticality tasks if there is some resource left. Combining together mixed critical real-time applications aims at an efficient resource usage.
  In the academic perspective, multiple WCETs are possible to represent task executions i.e., Vestal's model Equation (2.1), there exist the notion of criticality mode, and the classical scheduling algorithms are deterministic. The tasks that can switch to higher criticality modes whenever their lower WCET threshold is overcome, which make scheduling guarantees complex. It is left to MC scheduling algorithms the guarantee criticality modes (and mode transitions), and the respect of timing constraints in each mode. The claim of the academic approach to MC is that mixing applications of criticality levels would allow for more efficient resource utilization, while the complexity is for the scheduling algorithms.

- the *industry perspective* to the MC problem is more partitioning oriented. Temporal and spatial isolation between different criticality are defined, and are guaranteed with partitioning mechanisms such as OS and hypervisors. The partitions are for separating criticality levels in order to limit the interactions across criticality levels. The scheduling within the partitions can apply classical scheduling algorithms, both on-line and off-line. Task execute in single mode and criticality is more about importance or safety level of the functionality. The main claim of the industry approach to MC is that separating applications is safer for certification purposes or system development; the complexity is on guaranteeing the separation, especially with multi- and many-core implementations.

With both academic and industry perspective, the challenge is to safely make use of multi-core and many-core, and efficiently apply shared resources. For that, there is still plenty of room for investigating the MC problem as both academic and industry perspectives demands for further studies.

One of the main effort of the research project here defined is to make the two perspectives converge, such that:

- Similar requirements and certification arguments are defined. An interesting question is: *why certification arguments for SW talks about probabilities/failure rates for any criticality level, but safety critical guarantees are never probabilistic?*. Here the effort would be about probabilistic approaches and make them more trustable/reliable, such that the industry can rely on them. Also, schedulability analysis should approach schedulability guarantees from a perspective close to certification requirements.

- Similar languages and understandings appears from the research project to share MC notions among the two perspectives. Certification and certification arguments could help with that; also, timing models and schedulability analysis should approach the MC problem from a common background and similar techniques.

- Same (at least similar) techniques for timing guarantees and efficient resource usage to be developed: *trading partitioning with safe sharing* would allow efficient resource usage with comparable safety guarantees.

On the other end, it is not always possible to propose solutions that would work well for academic and for industry. There will be proposed adhoc solutions which best approach peculiar challenges that either academic or industry have with respect to MC. Multi-core implementations, and upcoming many-core platforms, are creating plenty of opportunity to keep developing efficient solutions which apply to one or the other perspective.

5.1 Lessons learned on mixed criticality

The interest to MC problem is motivated by the same trend that is ongoing with real-time: real-time embedded system are becoming more and more mixed critical, thus real-time has more and more to consider MC timing models and MC schedulability analysis. This is especially true with multi-core and many-core implementations and the demand for safe resource sharing and guaranteed partitioning. MC tackles with both, thus the interest for it.

In the last three years, some works have been made on MC. Earlier than two years ago, it has not been considered the MC problem because the interest was about understanding multi-core platforms and their challenges for real-time guarantees with more classical approaches i.e., pre mixed criticality problem. Since 2015, the MC problem is seen as crucial designing and developing real-time embedded systems; it is seen the need for timing modeling and for guaranteeing timing constraints in presence of different criticality levels. This comes from working on safety critical and mixed critical systems such as avionic and space as a researcher at the ONERA.

The research project proposed is for investigating MC. In particular:

- it is for deriving more accurate and flexible timing models for the multiple requirements that MC applications have;

- it is for developing scheduling algorithms that can safely mix tasks with different needs;

- it is for exploring less pessimistic resource partitioning and safer resource sharing.

The research project is also motivated from professional and research experiences. Hence for, both academic and industry perspectives to MC will be considered with shared and adhoc solutions.

All the contributions made, and that will continue to be made, apply modeling and analysis tools developed since the PhD. In particular, probabilistic timing models that will contribute with more flexible task timing models coping with the different criticality modes, and deterministic as well as probabilistic schedulability analysis to improve already existing solutions. All of the contributions aim at reducing pessimism and more efficient resource utilization.

So far, the contributions made tackled with timing models and schedulability analysis such that:

- The MBPTA approach and the quality of the input measurements are investigated in details in order to improve the quality of the pWCET estimates [159, 163, 192, 193]. Those works are not presented in Chapter 4, but they affect MC timing models as multiple execution conditions are taken into account.

- Formal deterministic bounds to task behaviors, deterministic schedulability analysis, and deterministic sensitivity analysis are applied to MC. The bounds are able to account for different MC modes and mode combinations between tasks [190, 198]. These works involve mainly the industry perspective to MC, since task applications are partitioned and within the partitions classical scheduling paradigms are applied.
- It is proposed a deterministic scheduling algorithm that releases some non-realistic hypotheses of the classical MC scheduling [166]. With it, the pessimism of deterministic MC scheduling algorithm can be reduced.
- Probabilistic sensitivity analyses have been developed and applied for academic and industrial MC problems [184, 191, 194]. The abstract representations considered are the pC-space and the probabilistic (α, Δ)-space. The sensitivity analysis, equivalently parametric analysis, relates small changes of WCETs or resource provisioning to schedulability conditions. Initial trade-off resource-schedulability-criticality are studied, and some changing strategies for efficient resource usage are proposed.
- Timing analysis and schedulability analysis for MC with fault and fault effects have been marginally approached [194]. In this work, the faults are abstract and considered as extra execution time; probabilistic timing models are derived to describe fault effects. Different fault models have been studied and modeled with safety levels, equivalent to criticality levels. Probabilistic sensitivity analysis with the pC-space is applied to quantify fault effect into schedulability conditions;
- It is proposed a probabilistic scheduling algorithm which is able to leverage probability into scheduling decisions [165]. This work has some limitations, but it represent the first schedulability analysis in which probabilities are actively applied into scheduling decisions.
- Some early works on formal methods for modeling MC behavior and defining formal MC scheduling decisions have been proposed [202, 204]. Formal methods are to model task executions and mode changes. The scheduling strategies are off-line and based on probability information; they act selecting the LO-criticality jobs to drop in order to maximize the resource utilization and minimize the probability of system HI-criticality mode.

Figure 4.1 recaps the contributions made so far to MC which have been applied to avionic, space, and robotic real-time applications. The lack of contributions with real-time calculus and its probabilistic version pCalculus is because so far the research activities insisted more on bounding functions i.e., demand bound function and level-i workload function, both deterministic and probabilistic. Future work will consider also the real-time calculus for MC applications, as the group of Prof. Lothar Thiele is actually doing at the ETH.

In the following, the contributions are commented with the lessons learned and future directions for those. There are presented first probabilistic approaches to timing analysis, then deterministic approaches for schedulability analysis, and in the end probabilistic approaches to schedulability analysis.

5.1.1 Probabilistic timing models for mixed criticality

The purpose of pWCET estimates is to give more insight on what happens to execution times smaller than the deterministic WCET estimate. Such more flexible probabilistic worst-case models have multiple possible WCET thresholds each with the confidence associated i.e., the probability of being overcome. With the new developments [148, 149, 159, 162, 163, 180, 182, 192], it is possible to have an accurate and confident pWCET distribution per execution scenario. To each scenario there can be applied the notion of safety or criticality level since it represents specific execution conditions. To stress that a pWCET distribution is the worst-case for only the scenario considered: they are relative pWCET, which fits well with MC.

Probabilistic representations can be applied to the MC problem such that the task model of a HI-criticality task changes into different possibilities, each with a probability associated.

There can exist the "probabilistic LO-criticality model" for HI-criticality tasks:

$$\tau_i \overset{def}{=} ([\langle C_i(\text{LO}), p_i(\text{LO})\rangle, C_i(\text{HI})], T_i, \chi_i), \tag{5.1}$$

where $C_i(\text{LO})$ is associated with the probability $p_i(\text{LO})$ as the probability (equivalently confidence) that tasks has of overcoming $C_i(\text{LO})$ while executing. $p_i(\text{LO})$ represents the probability of mode change [165]. The HI-criticality mode remains deterministic and impossible to overcome.

There can also be the "pWCET model" for HI-criticality tasks:

$$\tau_i \overset{def}{=} (\mathcal{C}_i, T_i, \chi_i), \tag{5.2}$$

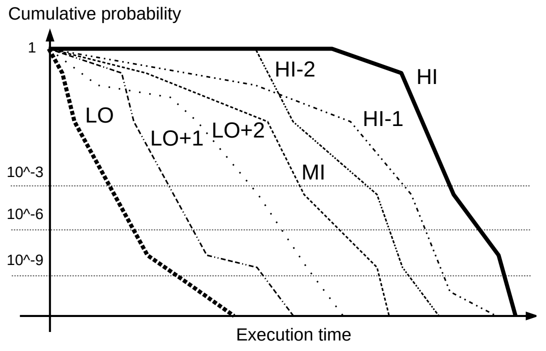


Figure 5.1: Multiple pWCET distribution for a MC task with the inverse cumulative distribution representation. Each distribution represent a criticality level or equivalently an execution condition; WCET thresholds can be extracted from each distribution at different probability which are represented by horizontal lines.

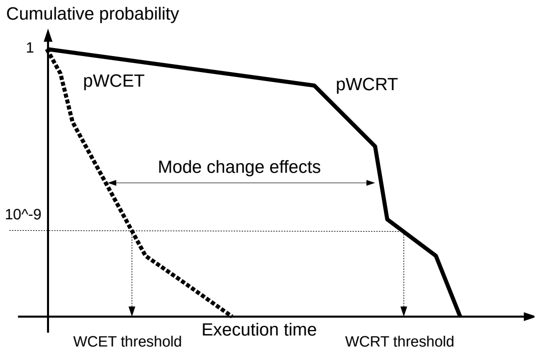


Figure 5.2: pWCET and pWCRT representation for mode change. HI-criticality and LO-criticality levels can be represented with WCET threshold and WCRT thresholds extracted respectively from pWCET and pWCRT distributions.

where the pWCET distribution $\mathcal{C}_i$ describes all the modes possible to τ_i, each with a probability associated [194]. With discrete it is possible to associate as many criticality levels as the distribution domain; the probabilities would depend on the granularity of the distribution support. With continuous distributions, the criticality levels to be associated with any possible probability are potentially infinite.

In between Equation (5.1) and Equation (5.2), there could be the "probabilistic thresholds model" for HI-criticality tasks:

$$\tau_i \overset{def}{=} ([\langle C_i(\text{LO}), p_i(\text{LO})\rangle, \langle C_i(\text{HI}), p_i(\text{HI})\rangle], T_i, \chi_i), \tag{5.3}$$

in which both LO and HI modes are described with probabilistic WCET thresholds; each has a probability associated, representing the probability/confidence of overcoming it at runtime [110]. $p_i(\text{HI}) \leq p_i(\text{LO})$; all $p_i(\cdot)$ are cumulative probabilities.

$C_i(\text{LO})$ and $C_i(\text{HI})$ can come from the same pWCET estimates, at different probabilities as it is the case of Equation (5.2) and Equation (5.3), or two different pWCET estimates with different conditions. In the latter case, it can be inferred the "pWCETs model" for HI-criticality tasks such as:

$$\tau_i \overset{def}{=} ([\mathcal{C}_i(\text{LO}), \mathcal{C}_i(\text{HI})], T_i, \chi_i), \tag{5.4}$$

as an extension of Equation (5.2) with two different pWCET distributions describing task behaviors in the two possible criticality modes.

Figure 5.1 illustrates multiple pWCET estimates, one per criticality mode that the task can have. LO-criticality mode, LO+1-criticality mode, ..., HI-1-criticality mode, and HI-critcality mode are represented with examples of possible pWCET distributions that can overlap for high cumulative probabilities. Each pWCET distribution represent the worst-case model for the specific mode, thus execution condition (less critical, highly critical), considered. Figure 5.1 illustrates also how to infer criticality modes from a pWCET estimate i.e., selecting WCET thresholds at different probabilities. This could be the case for obtaining the models in Equation (5.1), Equation (5.2), and Equation (5.3) from the same pWCET representation. Instead, the two distributions in Equation (5.4) could come from two timing analysis tools, two separate criticality conditions required for timing analysis, or two execution conditions accounted for. This is also represented in Figure 5.1 with different distribution, one per tool considered.

For LO-criticality tasks, the "pWCET model" could be such that:

$$\tau_i \overset{def}{=} (\mathcal{C}_i, T_i, \chi_i), \tag{5.5}$$

where $\mathcal{C}_i$ represents the only possible LO-criticality behavior of the task.

Also, the "probabilistic threshold model":

$$\tau_i \overset{def}{=} (\langle C_i(\text{LO}), p_i(\text{LO})\rangle, T_i, \chi_i), \tag{5.6}$$

with only $C_i(\text{LO})$ representing the task behavior; $p_i(\text{LO})$ is the confidence associated to $C_i(\text{LO})$ bound, equivalently the confidence of not overcoming it.

In all those models, χ_{ij} parameter is the job criticality level, the same as χ_i or different in case each job could have its own behavior. In the probabilistic paradigm it has associated a probability representing the probability of mode change from LO to HI criticality mode. In a generalization effort, probabilities can be associated to any criticality level, included the HI-criticality mode.

All those probabilistic models would benefit MC models with reduced pessimism, and more flexibility; probabilities distributions and probability thresholds would help with more precise models that can represent better task behaviors. This would especially important with less critical levels.

Probabilistic criticality mode. Normally, χ is defined from pWCET models. By doing this, only the mode change effects that belongs to task conditions are included in such representation: the mode change depends on the task and other effect embedded into worst-case models for execution time: *the criticality level depends on the task*. Currently, there are under investigation other representations for probabilistic mode changes and χ parameters. For example, in [202, 204] it is studied the possibility of modeling mode changes from pWCRTs. The advantages of such representations can be significant, since in the pWCRT can be embedded more effects on mode changes, e.g. interference from concurrent tasks, preemption overheads, dependence. Another advantage with mode characterized with pWCRT is that *probabilities can be actively applied into scheduling decisions*: response time changes by dropping LO-criticality tasks or with a different job ordering [202, 204], thus the mode change probability can change accordingly: *the criticality level depends on the application*. In this case, dependence effects between concurrent tasks can be represented and taken into account into scheduling decisions. On the other end, the complexity with mode change representation from pWCRT consists of removing from pWCRT representations the preemption time and other conditions that do not actively impact mode change: the thresholds imposed have to take into account such problems. Figure 5.2 details the difference between pWCET and pWCRT representations for mode changes; χ parameter can be inferred from both with different impact, and the differences among the two representations are from the different effects that can be embedded. Figure 5.12 gives another example of pWCRT, continuous or discrete distribution, used for mode characterization.

In this manuscript are not discussed the motivation nor the validity of the possible probabilistic models proposed. It is only noted that probabilistic representations can potentially increase the amount of information in the task model. Such flexibility will reduce pessimism of task models and will be applied into schedulability analysis.

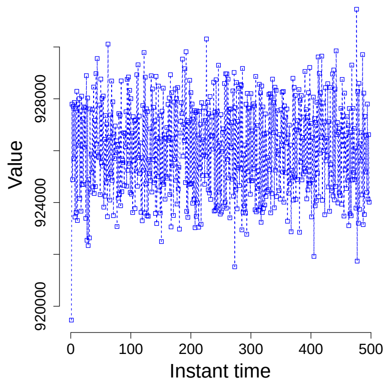
It is also true that probabilistic representations negatively impact the complexity of probabilistic MC schedulability. There are works made in the past for reducing the complexity from probabilistic representations [178, 179, 189] by re-sampling the distributions. Here the trade-off become in terms of flexibility and complexity, and efficient choices can be continued to be developed in the future. Future researches will focus also on reducing complexity of probabilistic schedulability analysis side in order to effectively apply probabilistic approaches to MC problems.

On the criticality of pWCET models

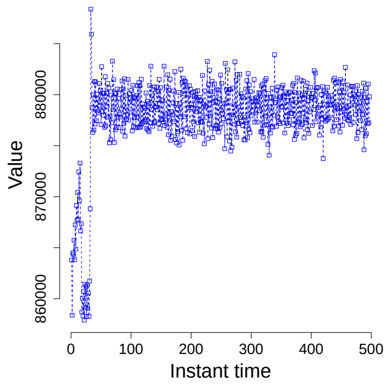
In [193] are the basis for the most flexible probabilistic task model that accounts for multiple modes and mode combinations possible as well as multiple bounding probabilities. The complexity from such MC representations is not considered here, preferring at this stage to investigate the benefits of a detailed probabilistic model to task MC behaviors.

Figure 5.3 presents the two traces of measurements and a comparison between pWCET estimates from the measurements in case of HI-criticality scenario e.g. most conservative and pessimistic described by $\mathcal{C}^{\text{HI}}$, and in case of LO-criticality scenario described by $\mathcal{C}^{\text{LO}}$. In Figure 5.3(a) and Figure 5.3(b), the two traces of measurements which represents the two modes for task; they are *trace2_1* for the *lms* Malardalen benchmark in LO-criticality mode, and *trace2_2* for the same benchmark in HI-criticality mode. Figure 5.3(c) compares the pWCETs illustrating the partial ordering between the two criticality level with *trace2_2* dominating *trace2_2*; it it, the pWCET is represented with ICDFs. The case study applied is the *lms* task from the Malardalen benchmark
`http://www.mrtc.mdh.se/projects/wcet/benchmarks.html` running on a multi-core platform [157].

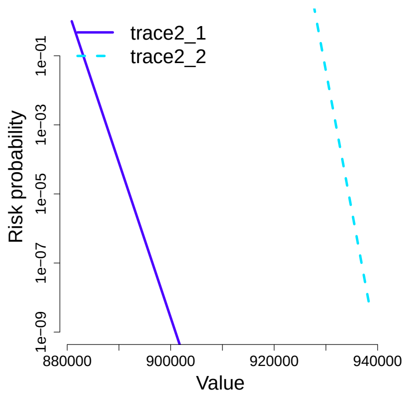
[193] has been presented to SETTA conference in 2017 as an alternative representation for MC task worst-case models. Such fine grained representation, coupled with probabilistic MC scheduler, will allow taking more accurate decisions depending on the actual execution conditions.



(a) *lms* HI-criticality trace; the execution time in ordinate are in CPU cycles



(b) *lms* LO-criticality trace; the execution time in ordinate are in CPU cycles



(c) icdf pWCET with execution time in abscissa are in CPU cycles; ordinate has log scale

Figure 5.3: *lms* benchmark with two traces of execution time measurements *trace2_1* (*lms* in LO-criticality mode) and *trace2_2* (*lms* in HI-criticality mode).

pWCET challenges

Probabilistic timing analysis and probabilistic models are able to represent accurately task mixed critical behavior. Parameterizing WCET bounds with probabilities, i.e., pWCETs, allows to better cope with criticality levels and the different safety requirements. This is how pWCET reduce the pessimism of the task models.

The main problem with probabilistic timing analysis is the strong dependence with hypotheses and conditions assumed for the measurements or the system behavior i.e., hypotheses assumed. This reflects into the notion of worst-case specific to the hypothesis/condition considered, i.e., relative pWCET $\mathcal{C}^{s^j}$ specific to scenario s^j versus the absolute pWCET which upper bound them all. To note that this is exactly the same problem happening to static timing analysis with multi-core processors [106]. With such platforms, the complexity is such that timing analysis is decomposed with hypotheses and WCETs are estimated according to those: there exist the WCET in isolation where the task execute in isolation within a core, and other WCET are estimated adding hypotheses to the isolation case. To note that the trend of separating hypotheses and execution conditions copes well with the MC problem and timing models which differentiates among criticality levels. This makes probabilistic models such as pWCET handy for representing tasks with multiple criticality as well as for multi-core and many-core implementations.

Static probabilistic timing analysis. The papers on SPTA [154, 158] have been applied to real-time architectures with randomized memories. At this stage, SPTA applies only with randomized replacement in cache memories. In [158], it has tackled with convolution of distributions and preemption effects on cache memories. In it, the MC problem is left for future contributions: multiple execution conditions will be considered with different preemption effect (preemption costs). As such, different pWCET estimates will be computed, each representing a specific execution condition, and thus a criticality level.

In [154], it has been applied the Markov chain formalization to represent randomized cache behaviors. The cost of cache hit and misses are embedded into the pWCET estimates with Markov chains models. In this work, different permanent and transient fault conditions can be applied, and pWCET estimates are computed for each. Implicitly, the MC problem has been considered in [154], since more conservative fault models can be associated with HI-criticality modes, while less conservative fault models can be associated with LO-criticality modes. Future work will continue applying formal methods for SPTA extending probabilistic models to effects other than faults.

Measurement-based probabilistic timing analysis. MBPTA does not rely on traditional analytical system models as static timing analysis. Instead, it empirically builds the models from the measurements, and that is a quality used for reducing complexity and increasing applicability of timing analysis. Unfortunately, MBPTA suffers from the challenge of guaranteeing the worst-case conditions with just measurements.
Contributions have been made to distinguish between the notion of absolute pWCET and relative pWCET [163, 192]. There are formalized models that keeps all the possible input scenarios i.e., "Worst-case group, or those which upper bounds every possible relative pWCET i.e., "Envelope". Also, works that begin considering the input measurements representativity problem [159] exist. For the moment, that contribution provides few observations to define measurements representativity, and how representativity impacts pWCET estimates.
The MC problem is touched by [159, 163, 192], in which the dependence to the input translates into task modes

and different criticality level models to task executions.
MBPTA and EVT in their classical formalization applies only if the measurements satisfy the "independence and identical distribution hypotheses". This makes difficult to use with realistic real-time embedded systems, in which there manifest dependence effects from one execution to the next. In [148, 149, 159, 162, 180, 182, 192], it has been formalized the generic definition of EVT which applies in case of weak dependence: with certain degree of dependence (local and extreme independence), the EVT can be applied and still provide safe worst-case representations. Under weak dependence, MBPTA can apply to realistic platforms, as proved by the test cases in the contributions made.
The production on measurement-based approaches is [148, 149, 153, 154, 156, 158, 159, 162, 163, 177, 180, 182, 192], while the projects related are PROARTIS, IRHEDO2, R2D2, MAMA, CAPACITES, and CAPHCA. Fabrice Guet PhD Thesis has largely contributed to this subject, including the tool diagXtrm[1] that he has developed and the ONERA is using in projects. Moreover, the probabilistic models have been objects of works with Master students Alessandra Melani, Corentin Damman, Gregory Edison, Gregor Vindry, and Julien Durand that have faced multi-core platforms. Within project CAPHCA, the collaboration with Postdoc Guillaume Phavorin is to extend diagXtrmapplicability to realistic multi-core real-time systems.

SPTA and MBPTA are relatively young approaches to task timing analysis and pWCET estimates: further development with those are required. With respect to SPTA, research activities are necessary to reduce the pessimism, and to extend its applicability to other randomized systems. For MBPTA, the need is to do research for increasing the reliability of the MBPTA methodology in each of its composing steps, and to define representativity of the measurements to be linked to pWCET estimates confidence.

Hybrid approach to timing analysis. Most important future work will be on developing the "hybrid approach" to timing analysis which can take advantage of the strong points of [deterministic] static timing analysis and measurement-based approaches. At one end, the effectiveness of MBPTA, which requires only measurements of execution times, can be applied to evaluate the quality of deterministic WCET bounds and the assumptions made to derive them. MBPTA can help static timing analysis to reduce his pessimism, especially with multi-core implementations. On the other end, an improved knowledge of the system via static timing analysis, can improve the safety of the MBPTA with more guarantees to have explored the wort-case scenarios with measurements. Mixing the two approaches will allow reducing the pessimism of task worst-case execution time estimates, thus benefit MC modeling for every criticality level. The development of the hybrid approach is in collaboration with IRIT Toulouse, Prof. Christine Rochange and Prof. Hugues Casse.

5.1.2 Deterministic schedulability approaches to mixed criticality

Some are the contributions with deterministic approaches made for the MC problem are [166, 190, 198]; they all relate to schedulability analysis applied to mixed criticality. In [190, 198], the deterministic sensitivity analysis has been applied to partitioned real-time embedded systems and MC trask sets; in [166] it has been released an important hypothesis about MC and system mode definition and the result obtained are less pessimistic.

Sensitivity analysis for MC

In [190, 198] it is formalized the deterministic sensitivity analysis through which explore trade-off between computational resource, criticality levels, and schedulability. The effect of changes on resource availability or on different modes are quantified with the abstract representation of the deterministic (α, Δ)-space.

The two works differentiate from the case studies and from the strategies developed. Multiple practical case studies are applied for validating the deterministic sensitivity analysis and outlining the advantage to design and develop MC real-time embedded systems.

The MC perspective adopted here is more the industrial one: no online mode changes are considered, multiple possible modes and mode combinations between tasks, instead what can change is the resource provisioning. The task sets are partitioned resembling to multi-core partitioned scheduling problem. For each partition classical EDF or FP schedulers are applied.

In the following, some results from the MC version of the ROSACE case study[2], named ROSACE++ [190]. The test case is a real MC real-time applications which requires partitioning to run on multi-core platform. In those, it is possible to see how sensitivity analysis works for defining better system design improving resource usage for the criticality level to be guaranteed.

[1] `https://forge.onera.fr/projects/diagxtrm` as open source tool form probabilistic timing analysis.
[2]ROSACE case study developed at the ONERA, `http://sites.onera.fr/schedmcore/ROSACE` and `https://svn.onera.fr/schedmcore/branches/schedmcore-RTAS2014/Case_Study_RTAS/`.

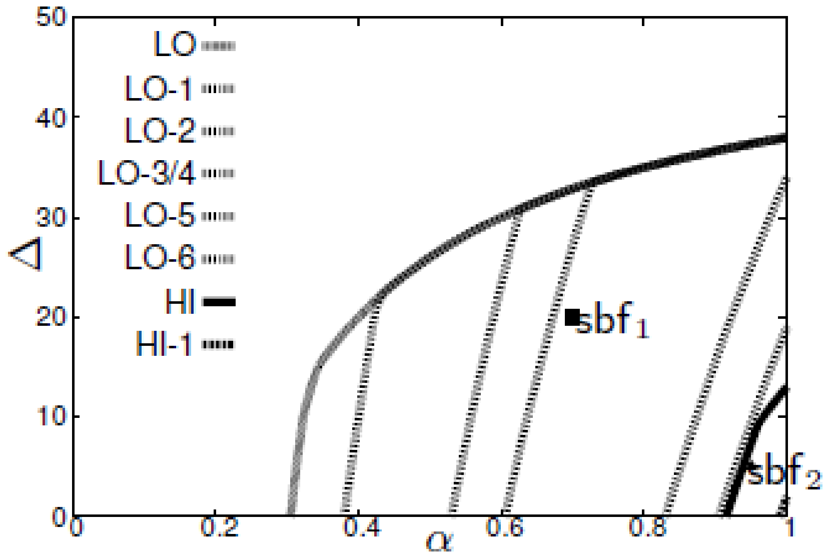
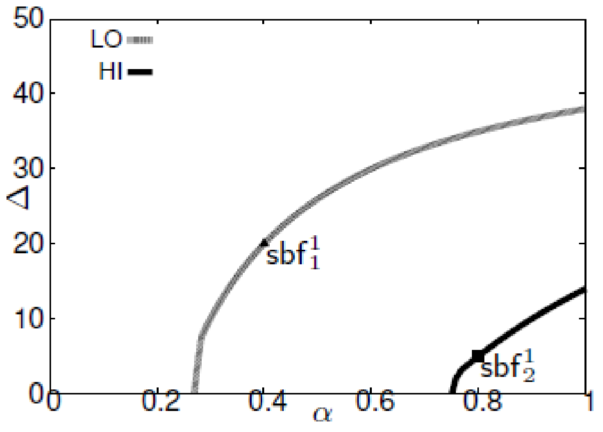
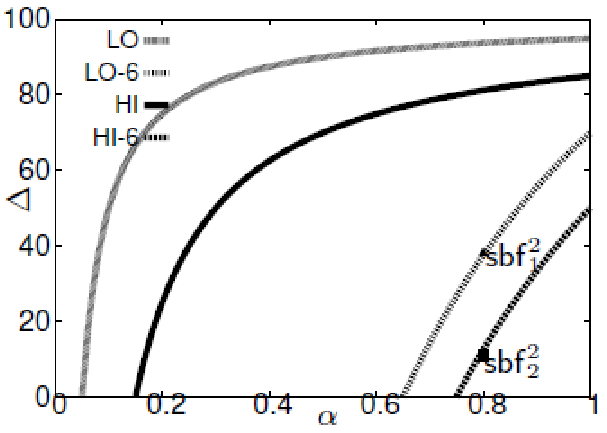


Figure 5.4: (α, Δ)-space and EDF feasibility regions with MC models for ROSACE++.



(a) Partition $P1$ with some bounds represented



(b) Partition $P2$ with some bounds represented

Figure 5.5: (α, Δ)-space and EDF with MC models for ROSACE++.

In this system design, the application of 14 tasks (11 HI-criticality tasks, and 3 LO-criticality tasks). A first analysis is without application partitioning. A resource provisioning of sbf_1 is able to guarantee schedulability of all HI-criticality tasks in LO-criticality mode plus any combination of the LO-criticality tasks, even all the tasks together. Instead, only a resource of $\mathsf{sbf}_2 = (0.95, 5)$ is able to guarantee the schedulability of HI-criticality tasks in HI-criticality mode. This comes with a cost of resource change of $\delta\alpha = 0.95-0.7 = 0.2$ and $\delta\Delta = 5-20 = -15$. A negative $\delta\Delta$ means a reduction of idle time, thus more resource to be provided. Figure 5.4 illustrates the case.

A second analysis is carried out partitioning the application with a first partition $P1 = \{\tau_1, \tau_4, \tau_5, \tau_6, \tau_7, \tau_9, \tau_{12}\}$, and a second partition $P2 = \{\tau_2, \tau_3, \tau_8, \tau_{10}, \tau_{11}, \tau_{13}, \tau_{14}\}$. In terms of utilization, $P1$ is such that $U_{\text{HI}}^{1,\text{HI}-1} = 188/200$, when the HI-criticality tasks are in HI-criticality mode; $P2$ is such that $U_{\text{HI}}^{2,\text{HI}-2} = 135/200$ when HI-criticality tasks are in HI-criticality mode. In terms of resource, per partition there are two possible resource availability: $\mathsf{sbf}_1^1 = (0.4, 20)$ and $\mathsf{sbf}_2^1 = (0.8, 5)$ for P1, $\mathsf{sbf}_1^2 = (0.8, 38)$, $\mathsf{sbf}_2^2 = (0.8, 11)$ for P2; Figure 5.5 illustrates the partitions, each with the available resources.

With P1 and sbf_1^1 available, it is possible to guarantee up to $\mathsf{dbf}_{\text{HI}}^{\text{LO}-12} = \mathsf{dbf}_{\text{HI}}^{\text{LO}} + \mathsf{dbf}_{\text{LO},12}$ (LO). All the other dbfs $\mathsf{dbf}_{\text{HI}}^{\text{HI}-k}$ are larger than the available resource. With sbf_2^1, it is possible to guarantee all the tasks in P1, see Figure 5.5(a) where only some of the bounds are represented. In it, HI stands for all the HI-criticality tasks in HI-criticality mode plus τ_{12}, while LO stands for the all the HI-criticality tasks in LO-criticality mode plus τ_{12}. The cost for the full schedulability within P1, guaranteed by sbf_2^1, is $\delta\alpha = 0.4$ and $\delta\Delta = -15$, see Figure 5.5(a) in which some regions are presented with the possible the supply bound function sbf. We remind that full schedulability means that LO-criticality tasks can execute concurrently with HI-criticality tasks without jeopardizing any timing constraint.

With P2, sbf_1^2 guarantees all the tasks, but with the HI-criticality in LO-criticality mode (LO-6 case in Figure 5.5(b)). Instead, with sbf_1^2, it is also possible to schedule all the HI-criticality tasks in P2 in HI-criticality modes; in that case, τ_{13} and τ_{14} have to be dropped. By increasing the resource to sbf_2^1, it is possible to guarantee all the tasks in P2 in HI-criticality mode (HI-6), see Figure 5.5(b). The cost for the full schedulability within P2 is $\delta\alpha = 0$ and $\delta\Delta = -27$. The partitioned solution can be implemented with a dual core platform.

Figure 5.5(a) and Figure 5.5(b) present some of the regions together with two possible sbfper partition. From that, the guarantees that can be provided with each sbfcan be inferred, while possible other changes can apply in order to increase the schedulability guarantees.

So far, resource availability and trade-offs are only quantified. Future works will develop strategies to decide optimal or sub-optimal changes for resource or system parameter i.e., task modes. Also, future contributions will consider better partitioning strategies in order to minimize resource pessimism.

MC scheduling with limited HI-criticality behaviors

Classical MC scheduling algorithms assume that once a HI-criticality task moves to HI-criticality mode, all the HI-criticality tasks are in HI-criticality modes. This is a pessimistic hypothesis which requires a lot of computational resource; it is also a non-realistic hypothesis since it does not reflect actual task conditions.

The deterministic scheduling contribution [166] releases that hypothesis since it is assumed that not all the HI-criticality tasks move to HI-criticality at the same time. This work elaborate on the (k, n) definition of criticality mode: $k \leq n$ HI-criticality tasks in HI-criticality mode which is an intermediate step toward MC scheduling where each HI-criticality task has to be treated separately.

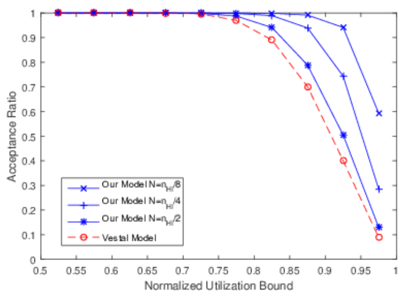


Figure 5.6: Schedulability ratio comparison of our proposed model and the classical Vestal's model under various N's, with $n_{\text{HI}} = 16$.

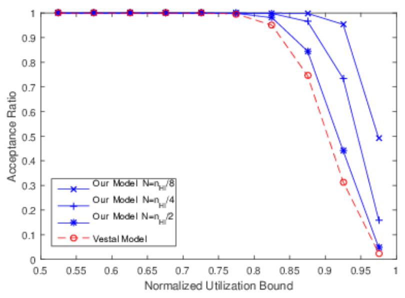


Figure 5.7: Schedulability ratio comparison of our proposed model and the classical Vestal's model under various N's, with $n_{\text{HI}} = 32$.

Figure 5.6 and Figure 5.7 show how beneficial is the work made in order to reduce pessimism. In particular, they demonstrate the effectiveness of the new model ("Our model") along with the corresponding EDF-VD schedulability test under various settings of numbers of HI-criticality tasks n_{HI} (16 and 32) and sizes of N (i.e., number of HI-criticality tasks that can simultaneously exceed LO-WCETs. It is natural that the acceptance ratios will drop when system is more heavily loaded (with higher utilization). However, we notice that our methods maintains relatively higher acceptance ratio even when normalized utilization gets close to 1.

The collaboration with Prof. Zhishan Guo and Prof. Kecheng Yang will continue investigating the complexity and pessimism of MC scheduling; more realistic hypotheses will be presented. The approaches that continue exploring (k, n) definition of system criticality mode, or those that apply the notion of graceful degradation will be investigated in future research. It has already planned to contribute to fixed-priority schedulers applying proposed model. The results will also be extended (at a measurable cost) into multi-processor and/or multi-criticality-level cases.

5.1.3 Probabilistic schedulability approaches to mixed criticality

Probabilistic approaches to schedulability analysis makes use of probabilistic task models. Scheduling conditions get parametrized with criticality levels and probabilities in order to cover the multiple cases that can happen at runtime. the objective of that is to reduce the pessimism of schedulability results; the probabilistic conditions will apply to criticality levels and the different guarantees that each criticality level could require.

There are developed probabilistic approaches to MC: i) some of them define a fine grained probabilistic representation which account for all the conditions/modes task can experience; ii) some of them define and apply probabilistic sensitivity analysis to MC problems; iii) others enhance task models with fault representations and apply such models to probabilistic schedulability analysis with sensitivity; iv) others define scheduling algorithms that effectively leverage probability into scheduling decisions; v) the rest apply formal methods for scheduling representation and scheduling strategies. The contributions are presented in this order.

Probabilistic sensitivity analysis for mixed criticality

The MC problem for real-time embedded systems requires for continue exploration of complexity issues, and for developing new solutions that can manage such complexity. The sensitivity analysis has proved to be effective in representing schedulability parameters and system resources with abstract representations such as the pC-space and the probabilistic (α, Δ)-space [184, 191]. Also, it proves to be effective in developing strategies to explore trade-offs between computational resource, schedulability and criticality levels. The trade-offs are i) *helpful in designing and guaranteeing the multiple conditions that MC real-time systems can run into*; ii) *developing changing strategies parametrized with criticality levels and resource usage.*

The research on probabilistic sensitivity analysis is not yet complete. The abstract representations have been formalized and initial thoughts to the β parameter have been laid out. See Figure 4.5 and Figure 4.6 for insight on investigation of β.

Future work will apply the representations into MC scheduling and MC probabilistic sensitivity analysis to better exploring the flexibility of the probabilistic representations. Future contributions are necessary to propose meaningful parameters for designing efficient MC real-time systems. In particular, it will be exploited the use of β, the same for the whole task set or each task with a different β_i $\overline{\beta} = \{\beta_i\}$, and how β can help developing MC real-time systems.

Also, future contributions are necessary to fully exploit the benefits of sensitivity analysis in reducing the complexity of MC schedulability analysis. With abstract representations there will be developed strategies to change resource or application parameters. The strategies are for better exploring the probabilistic regions and defining trade-off between schedulability and criticality levels that apply probabilities. The notion of probabilistic trade-off will be defined and applied to MC problems so that probabilities can have an active role into scheduling policies. Such active role have to be better defined, but can be inspired by [165, 204].

Fault-aware sensitivity analysis for probabilistic real-time systems

In order to be an upper bound, the WCET has to account for any condition, including the highly improbable pathological cases such as faults. This could lead to extremely pessimistic worst-case execution models. For these, pWCETs represent a valid alternative to classical deterministic models able to reduce pessimism with multiple WCET values.

A brief state of the art for fault modeling and real-time in order to motivate some contributions and possible perspectives.

Fault modeling and fault management intertwines with timing analysis as faults introduce latency to the task execution behavior which have to be embedded into the task models, [40]. As examples, in [116] backups are executed for fault tolerance and recovering form task errors caused by hardware or software faults; in [144] it is proposed an algorithm to abort and restart a task in case of conflicts in shared resources accesses.

Faults and fault effects have to be accounted into MC task models. Various methods applies to perform fault diagnosis [15, 76]; they can be combined with adaptivity and reliability analysis for safety critical systems [16, 64, 88].

Very few are the works which integrate fault effects into real-time analyses. [194] is an example of faults and fault effects included into probabilistic task models, and then applied into schedulability analysis. The notion of safety applied in the paper is equivalent to criticality as it is used to distinguish execution scenarios different than the absolute worst-case. Besides, [194] applies the probabilistic sensitivity analysis to evaluate the impact that faults have on the scheduling of real-time applications.

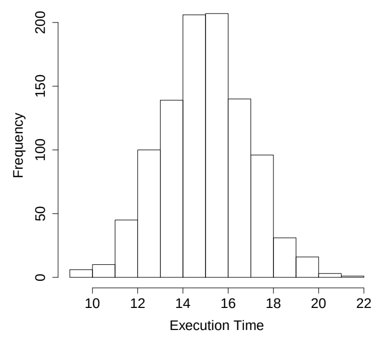
In [194], it is presented an example of industrial case study of a multi-core avionic application, also applied in [192]. The two papers consider real-time tasks executing under different scenarios and the MBPTA is applied to the traces for inferring pWCET estimates for the possible scenarios. The results obtained are available in the two a fore mentioned papers. This is The contribution of the paper with respect to timing analysis; Figure 5.8 depicts how the task models can be affected by fault models.

[193, 194] are two initial works for exploring the impact of hypotheses and execution scenarios on pWCET estimates. With different possible execution conditions it is possible to propose multiple pWCET estimates, each representing a worst case for the condition considered. This reflects today's trend with STA and the use of hypotheses to distinguish conditions and WCETs.

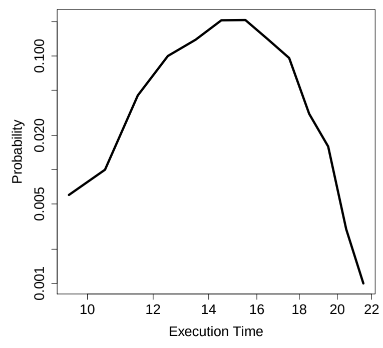
Future work is necessary for continuing integrating fault effects into timing analysis approaches and derive task pWCETs which are more realistic and pessimistic less. Natural will be the application of the results in [194] to the MC problem, since the idea behind Vestal's MC task model "for each criticality level there could be a WCET model that best fit the task behavior under this case", matches fault effects and the need for including those into MC timing models.

In [194], with fault probabilistic task it is applied probabilistic sensitivity analysis. Here the idea is to have schedulability analysis that can profit of more accurate task models in order to study the impact that faults have, or could have, on scheduling decisions. Better strategies and trade-offs will be proposed in the future, as well as scheduling algorithms that can effectively apply probabilistic models.

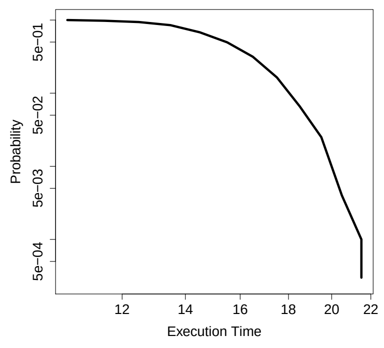
Future work is necessary to study more general coupling of fault effects, probabilistic models, and MC, along the line of [154, 194]. In those, the modeling with faults will parameterize the execution scenarios, and criticality levels, with faults models.



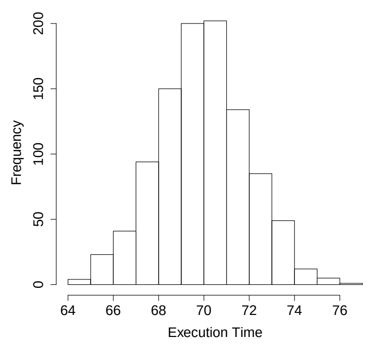
(a) Second scenario *no-fault*: histogram



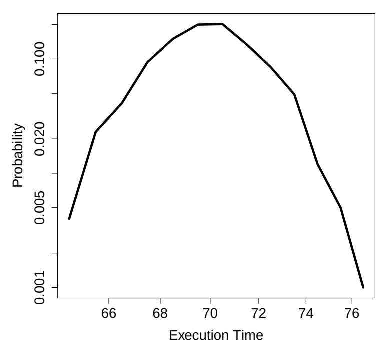
(b) Second scenario *no-fault*: PDF



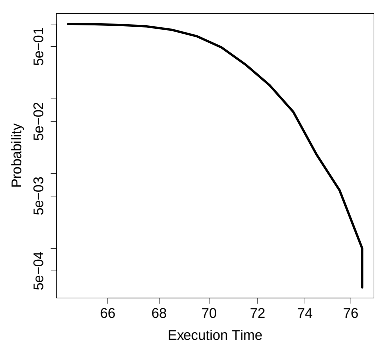
(c) Second scenario *no-fault*: CDF



(d) First scenario *faults*: histogram



(e) First scenario *faults*: PDF



(f) First scenario *faults*: CDF

Figure 5.8: Two execution scenarios (without faults and with faults) with histogram, PDF, and CDF representations. The two scenarios are empirically constructed from 1000 measurements following two different Gaussian distributions.

In Figure 5.8 are presented two execution scenarios *no-fault* and *faults*. The two scenarios are empirically constructed from 1000 measurements following two different Gaussian distributions, but they can represent realistic cases: one where the execution scenario does not considered fault conditions, the second in which faults are included into the measurements. The difference among the two is the fault effect on task executions.

Also, in [194] only the formalization of abstract representations probabilities and faults models are defined; in particular, abstract representation applied is the probabilistic C-space. Future work will apply the probabilistic (α, Δ)-space as well as sensitivity analysis strategies for enhanced trade-off evaluations.

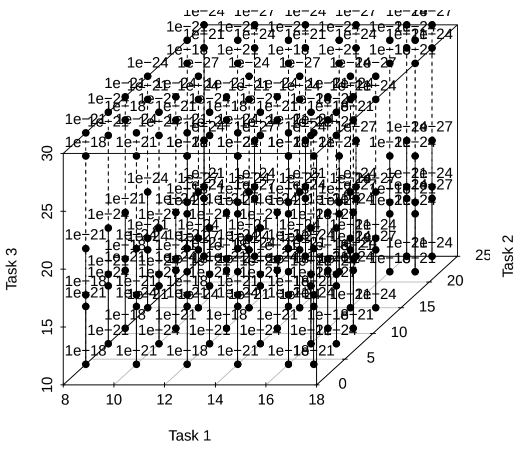


Figure 5.9: Discrete points represented with probabilities associated to each point.

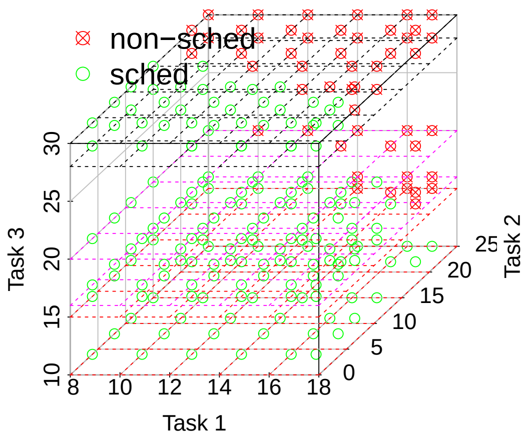


Figure 5.10: pC-space with discrete points and safety levels for task τ_3.

Figure 5.9 and Figure 5.10 illustrate the pC-space representation for the test case in [194]. In particular, Figure 5.9 represents the space with probabilities for each point in it; the probabilities are cumulative and come from scenario combinations; Figure 5.10 represents the same space where each point is judged as schedulable or not schedulable. From the pC-space, sensitivity analysis can be developed in order to explore the enormous flexibility of probabilistic representations.

Future contribution will be devoted to proficiently make use of probabilistic MC task models (with different fault effects) into probabilistic schedulability analysis. This will contribute building the MC schedulability analysis in presence of faults. Future enhancements to sensitivity analysis will exploit different parameter to fault models and relate scheduling conditions to fault conditions. Trade-offs will be developed to explore regions, resource needs and guarantees that can be provided for each fault model.

Probabilistic scheduling for MC

[165] proposes a MC scheduling algorithm which mixes probabilistic task models for LO-criticality modes and deterministic WCET for HI-criticality modes. Even if this is a probabilistic framework, for HI-criticality tasks the constraints remains hard real-time – deterministic model. The scheduling algorithm applies probability information into scheduling decisions in order to reduce the pessimism and increase resource usage with respect to classical deterministic MC scheduling.

The algorithm proposed is the LFF-clustering algorithm; it implements a runtime strategy with servers that executed every time units. The strategy is such that: at any time instant that the server is executing, the active HI-criticality job, if any, with earliest deadline is executed; if there is no such job, the current job of the server is dropped. All jobs including the server are scheduled and executed in EDF order, and a job is dropped at its deadline if it is not completed by then.

Note that although it is introduced a server task with period of 1, preemption does not necessarily happen that often. The goal of the sever task with utilization Δ is to preserve a "bandwidth" of at least Δ for HI-criticality jobs if the HI-criticality ready queue is not empty. There are three situations to be considered:

Figure 5.11 compares LFF-clustering algorithm with more classical MC scheduling. As it can be seen, by applying probability into scheduling decisions it is possible to reduce the pessimism and increase resource usage with respect to classical deterministic MC scheduling. The solution proposed has limitations in the server

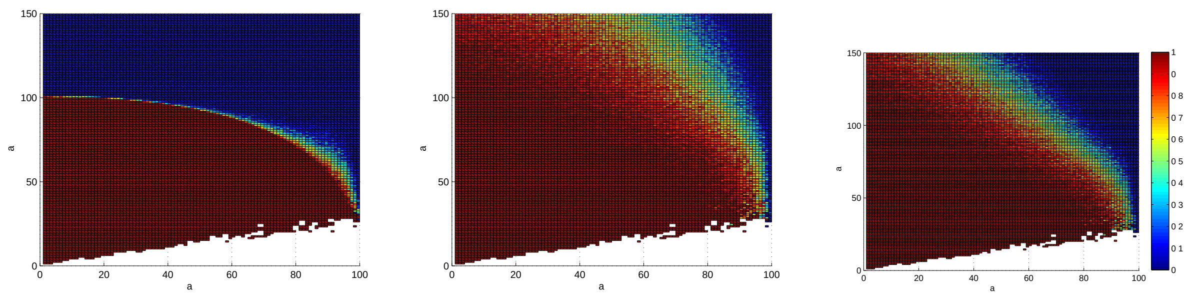


Figure 5.11: Schedulability comparison of the EDF-VD algorithm (left), the algorithm when returning partial correct or correct (middle), and it returning only correct (right), where color of each block represents the percentage of schedulable sets within certain utilization ranges.

implementation, due to the need to react at each valuable time instant (arrival, deadline, period, etc.) in order to be effective. From that, the solution proposed with servers of period 1.

In future work, the server period assumption of 1 unit of time will be released by applying adaptivity to resource reservation [185, 206]. With the analysis of the deadline and task periods, it would be possible to implement realistic servers which adapt their period and budget to the MC-scheduler needs, while leaving the system predictable at any time interval. Also, the sensitivity analysis will help releasing such constraints and extend the algorithm applicability to real implementations.

Formal methods and MC

The idea of applying formal methods and model checking into schedulability analysis is for benefiting of the formal mathematical support of such approach, in order to develop models which can be validated/checked. The models are used to describe task and scheduling behaviors. The aim with formal methods is to *formally and correctly representing the system dynamics allows also for more efficient scheduling decisions.*

Formal methods and schedulability. With the rising complexity of schedulability analysis, alternative approaches are appearing to manage that issue. The thesis Jasdeep Singh at the ONERA is to explore some alternatives, in particular the approaches that apply formal methods for probabilistic scheduling problems.
Formal methods which could cope with the requirements of non-determinism and probability of choice are probabilistic timed automata, stochastic timed automata, stochastic Petri net, stochastic model checking and Markov chains.
Through probabilistic timed automata, many state transitions can emanate from a state; each transition has an associated probability [132]. With pWCETs there is no question of choice of what the task does, since it can only execute after it is released once the computational resource is made available. PTA cannot model the time spent in executing (time staying in a state), as pWCET would represent.
Another formal method which could be used with pRTSs is stochastic model checking [43]. With this, performing Monte Carlo simulation requires a source of randomness. Unfortunately, safety of the source of randomness being used in real-time system analysis is an open question.
Stochastic Petri nets can model probabilistic task executions through stochastic time triggers. Existing work on stochastic Petri nets [152] can analyze probabilistic task set at a high computational cost. On the other hand, it provides exact result of probability of deadline miss along with the trace representing the probability of task being executed at a certain time.
The Markov chain a set of states and transitions in which each transition is labeled with the probability of being chosen [115]. Markov Chain possesses the memoryless property in which the determination of future state depends only on the present state. Some works apply MC to model cache memories with random replacement policies [39]. They are SPTA approaches, and use MC to compute discrete pWCETs. Another set of works has applied MC with schedulability analysis [48]. In there, the Markov chain is only to represent the system utilization and study its stability in the hyperperiod; the Markov chain does not model task executions nor task interactions as it is planned to do with CTMC.
In [152, 201, 203] the initial effort of the thesis has been to apply continuous time Markov chains and stochastic Petri nets to schedulability analysis of real-time systems. Although the use of formal methods makes execution models and scheduler more formal and easier to validate, problems to be solved still exist. Future work with formal methods and probabilistic scheduling developed with formal methods will show the strong points of

formal methods applied to complex scheduling problems. It will also apply formal to the MC problem where the probabilities will allow defining efficient scheduling decisions.
To mention that Jasdeep Singh developed a tool `https://forge.onera.fr/projects/probscheduling` to model probabilistic real-time schedulers with CTMCs. In it there is the interface with model checkers allows to formally check scheduler models and do probabilistic schedulability analysis from the models.

Formal methods and MC. Recent works from Jasdeep Singh thesis attempt to use formal methods in order to represent probabilistic MC problems with formal methods [200, 202]. The effort is to tackle with the extreme complexity of the MC schedulability with the use of easier to validate formal methods and model checkers. Markov decision process and Discrete Time Markov Chain (DTMC) are the tool applied with MC problems; with them are defined and investigated path among job states as sequence of task MC modes. With path exploration it is possible to quantify the probabilities of system criticality level across the paths; also, a more generic (and more realistic) definition of (k, n) system criticality can be applied. Most of all, with Markov decision process and DTMC there are developed "smart" LO-criticality job dropping strategies that select the least number of LO-criticality jobs to drop executing in order to reduce the probability of HI-criticality mode. The MC system models are built and formally validated in order to increase the confidence on the correctness of the models as well as of the analysis developed on top of them.

Figure 5.12 shows the example in which the criticality mode is defined from pWCRT continuous or discrete distribution. There are depicted HI- and LO-criticality regions: if the job finishing time is in $[l_{ij}, \infty)$, the job is considered to execute in high criticality mode, otherwise the job executes in low criticality mode, $[0, l_{ij})$. The probability is associated to mode transition from LO-criticality mode to HI-criticality mode. l_{ij} is the criticality WCRT threshold which is used to distinguish between a LO-criticality behavior (mode) and a HI-criticality behavior of a task; the job criticality level is χ_{ij}.

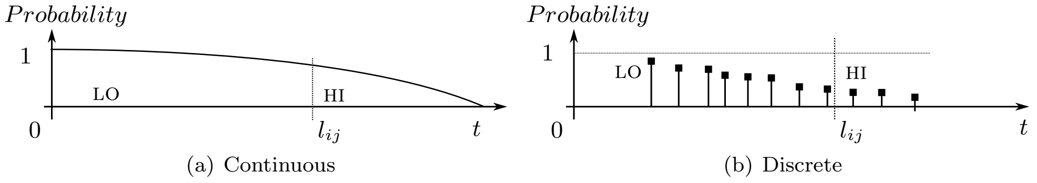


Figure 5.12: pWCRT of job J_{ij} in its ICDF in (a) continuous and (b) discrete form . High and low criticality regions are separated by l_{ij}. Low and High criticality zones denoted as LO and HI.

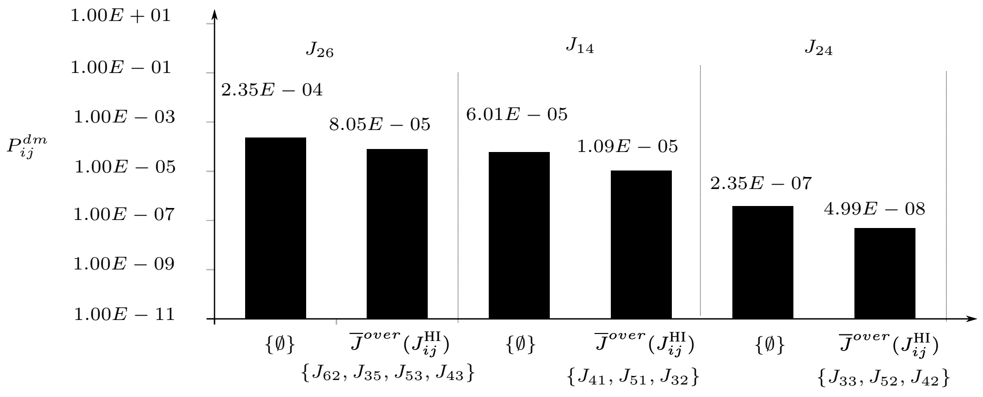


Figure 5.13: Probability of deadline miss of the jobs J_{26}, J_{14}, and J_{24} vs the LO-criticality jobs dropped.

The scheduling strategy proposed consists of dropping LO-criticality jobs in case some objectives are not achieved. The probability of moving to HI-criticality system mode is among the objectives. Figure 5.13 details some results on the dropping strategy implemented. It illustrates the probability of deadline miss for the jobs J_{26}, J_{14}, J_{24} before and after dropping corresponding LO-criticality jobs. Sensible modification applies because the job dropping changes job response time. The percent change in P^{dm} of the HI-criticality jobs from dropping LO-ciriticality jobs is also shown. This effect can be applied also to control probability of mode change, e.g. job passing to HI-criticality mode, whenever the pWCRT modeling of modes is applied.
However, from the discussion in [202, 204], a best case can be ensured if all the jobs in the set $\overline{J}^{over}(J_{ij}^{\mathrm{HI}})$ are dropped. In all the other cases, the reduction cannot be ensured to be the best for the job or the whole system.

Current works assumes that task execution times are independent. This is an unrealistic assumption, but it will be weakened in future work. Also, future work is under development to apply graph models and graph theory to represent probabilistic MC scheduling. Future extensions will keep applying Markov decision processes and DTMC, but will also explore other formal methods for reducing the complexity of the MC scheduling problem and for developing more efficient schedulers.

5.2 On the future of mixed criticality

In the MC framework, timing analysis and schedulability analysis demands for more work with respect to MC. The issues addressed in the research project are representativity, flexibility, complexity, and pessimism. They instantiate for multi-core and many-core implementations with the challenges due to the enormous resource availability, the contention to it, and the predictability guarantees.

The guidelines for future contributions are derived from four keywords: representativity, flexibility, complexity, pessimism

- *Representativity* is for timing models that have to represent well the MC behavior: they have to be able to consider multiple modes and/or multiple execution scenarios, they have to include well represented interference from concurrent executions or shared resources with multiple cores available, etc.. Representativity for schedulability analysis means that the hypotheses considered are representative of the system real behavior: they are not too pessimistic, nor too far away from nominal behavior.
- *Flexibility* is the main quality that timing models and schedulability analysis have whenever embedding multiple system behaviors. Flexibility for timing analysis is considering the worst-case model for hard guarantees (HI-criticality tasks) and also different worst-case thresholds for lower criticality levels. Probabilistic models or complete deterministic models have to be applied in order to obtain flexible task models. Flexibility is also for schedulability analysis, and it means including all the thresholds into multiple scheduling conditions: the timing constraints of all the criticality levels have to be guaranteed depending on the requirements of each criticality level.
- *Complexity* is key for conceiving and validating MC systems with multi-core and many-core. Nowadays MC scheduling algorithms are complex and not that performing; complexity is reduced with more efficient deterministic scheduling approaches, and by developing alternative scheduling approaches like formal methods, with the use of probabilities, and with sensitivity analysis.
- Today's MC timing models and MC schedulability analysis are *pessimistic*; such pessimism is not affordable anymore, even with multi-core and many-core implementations. The computational resource has to be well managed since it is limited and MC applications are very demanding. With more representativity, with the use of sensitivity analysis, and applying probabilities to timing models and schedulability analysis more efficient resource usage is guaranteed and the pessimism is reduced.

Special attention will be given: i) *to develop accurate statistical modeling analysis for studying interference, independence, and partitioning properties of multi-core and many-core platforms*; ii) *to develop a timing analysis that combines measurement-based and static in order to reduce pessimism and increase applicability for multi-core and many-core real-time systems*; iii) *to develop deterministic schedulability analysis processes which are more realistic, and thus less pessimistic*; iv) *to develop probabilistic schedulability analysis efficient, with reduced complexity, and effective in applying probabilities into scheduling decisions.*

As for the previous chapters, future work on MC is presented with timing analysis first, and schedulability analysis next.

In all those future works with timing analysis and schedulability analysis, it will be applied both the academic and industry perspectives. Moreover, the focus will be on multi-core and many-core implementations with the challenges they pose for MC.

In Figure 4.1 are depicted previous contributions and present works for the MC problem. It recaps some contributions made for the MC problem in terms of timing analysis approaches as well as in terms of scheduling analysis approaches. But what has been done so far is just an initial step; Further works and projects are already under development and will be more specific to multi-core platforms and the challenges they present. The lack of contributions with real-time calculus and its probabilistic version pCalculus is evident in the figure. This is because so far the research activities insisted more on bounding functions i.e., demand bound function and level-i workload function, both deterministic and probabilistic. Future work will consider also the real-time calculus for MC schedulability analysis, as the group of Prof. Lothar Thiele is actually doing at the ETH, and the pCalculus for more flexible representations and efficient schedulability decisions depending on probabilities.

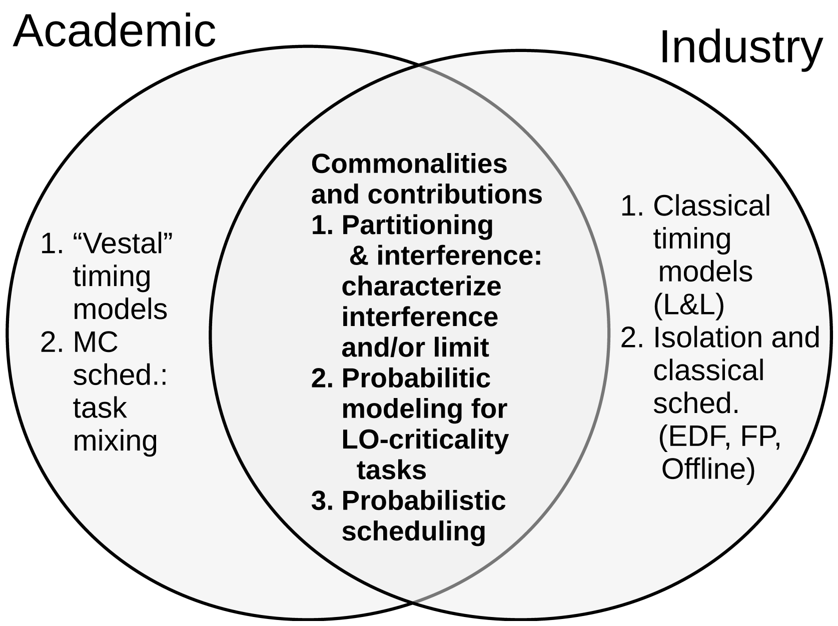

Academic
Industry
1. "Vestal" timing models
2. MC sched.: task mixing
Commonalities and contributions
1. Partitioning & interference: characterize interference and/or limit
2. Probabilitic modeling for LO-criticality tasks
3. Probabilistic scheduling
1. Classical timing models (L&L)
2. Isolation and classical sched. (EDF, FP, Offline)

Figure 5.14: MC problem for academics and industry with commonalities and differences.

Figure 5.14 illustrates some challenges related to MC for both industry and academics. In it, it is briefly presented what could be considered the differences among them with impact on timing models (Vestal's task model vs Liu and Layland task model – L&L) and schedulability analysis (mixing tasks vs separating tasks – partitioning). Instead, at the intersection there are commonalities and potential contributions to both industry and academic. Such contributions recaps what has previously discussed and will defines the some important directions for the research project. Those will be approached to close the gap between academic and industry perspectives as well as to provide adhoc solutions which would benefit one or the other. In the intersection, the picture presents what I consider to be the main issues to be addressed with academic and industry perspectives, for multi-core and many-core; they recaps present and future contributions previously discussed.

1. Partitioning problem: guaranteeing partitions and characterizing interference – Partitions are for isolating MC tasks and avoid that hazardous phenomena propagates or infect other HI-criticality tasks. They will be investigated with statistical analyses and analytical approaches. Partitions properties, guarantees, and their impact to MC models are evaluated. This is mostly helpful to the industry perspective to the MC problem.

2. Need for enhanced probabilistic models – There will be proposed probabilistic models for lower criticality tasks that can account for different execution conditions and embed different confidence level according to the criticality required. The flexibility from probabilities will reduce pessimism of timing models. This will contribute to both academic and industry perspectives.

3. More effective scheduling – The contributions relate to deterministic and probabilistic schedulability analysis. With deterministic approaches, the releasing non-realistic hypotheses and trade-offs are proposed. Academic and industry perspectives will benefit thanks to the sensitivity analysis and deterministic approaches. Probabilities are for reducing pessimism and for taking better scheduling decisions that allow for more efficient resource usage. Complexity and MC system design are explored with probabilistic sensitivity analysis. Academic and industry perspectives will be impacted from use of probabilities within the partitions and across them.

Timing analysis and schedulability analysis further investigation will consider and revise some of the classical assumptions to MC made with academic contributions. Special attention will be given to the MC task modeling where the multiple execution time threshold will be generalized with probabilistic task models. The multiple pWCET applied will result from MBPTA and the hybrid timing analysis approach applied to different execution conditions or to different interference scenarios. Beside, it will be investigated the tools for task modeling i.e., MBPTA, the SPTA and the hybrid, for their degree of independence and to which criticality level is better apply one or the others.

Also, special attention will be given to MC schedulability analysis approaches in order to enforce efficient resource usage with improved scheduling algorithms. At one end, it will be applied and verified more realistic execution conditions with deterministic schedulability analysis. On the other end, the probabilistic schedulability analysis will apply actively into scheduling decisions to leverage the different criticality requirements into smarter scheduling decisions.

In future contributions it will be explored the difference in modeling and analysis between ”task” criticality levels and ”application” criticality levels; motivations of the two different notions will be explored while the results obtained will be compared. This will continue the work started with Jasdeep Singh thesis for exploring differences between pWCET representations for MC tasks (task criticality) and the pWCRT representations for MC tasks (application criticality).

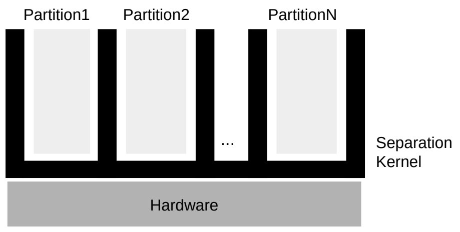


Figure 5.15: Partitioning problem: guaranteeing isolation and separation of criticality.

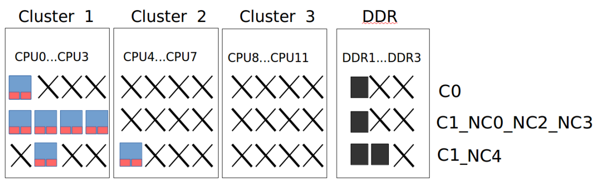


Figure 5.16: Execution scenarios with task placement onto different partitions.

Timing analysis for mixed criticality

The timing analysis defines the possible interpretations of predictability with MC. Task timing models are for clarifying ontological and epistemological hypotheses of system behavior. With multi-core and many-core implementations, epistemic variability increases enormously, while ontological hypotheses are necessary to simplify system behavior.

Timing models have to be built depending on the criticality requirements. That is achievable with models specific to well identified execution conditions that can happen within the systems; among those conditions, worst-cases have to be included. Research on timing models aims for the best possible representation that can minimize pessimism and reduce complexity: good models have to be produced for specific modes (representativity).

Probabilistic models and probabilistic timing analysis will be effective in modeling the variability that exist in today’s multi-core and many-core embedded systems. Also, they will apply to study interference that can happen at runtime and that can affect task executions. Interference are evaluated in order to identify interference channels according to certification standards, e.g. CAST-32A; partitioning mechanisms are evaluated to see their capability of temporal and spatial isolation; mechanisms/execution conditions will be applied to mitigate interference effects or limit anomalies propagation.

It is chosen to investigate the probabilistic timing analysis approaches because probabilistic models have proved flexibility and accuracy [159, 163, 192, 193]. Probabilistic models will allow better representing the multiple requirements that each criticality demands i.e., more flexibly and less pessimistically. At one end, the possible multiple execution scenarios will be modeled with probabilistic models; on the other end, interference effects will be characterized with probabilistic representations for nominal and worst-case behaviors.

Particular attention will be given to the lower criticality tasks and their representation with probabilities; this is to associate a confidence degree (probability) to their worst-case models of lower criticality modes, while the HI-criticality tasks remain deterministic in their representations. The mixture of deterministic and probabilistic representations will allow to apply probabilities for guaranteeing both hard and soft real-time constraints: probabilities reduces pessimism of lower criticality modes, deterministic models enforce safety with HI-criticality modes.

For MC timing analysis, the new collaboration with Polytechnic of Milan (Prof. Wiliam Fornaciari and PhD Student Federico Reghenzani), and the ongoing collaboration with Prof. Iain Bate (University of York), PhD Liliana Cucu Grosjean (INRIA Paris), and PhD Francisco Cazorla (Barcelona Supercomputing Center) will allow improving all the aspect above mentioned. In particular, with Polytechnic of Milan, the collaboration will produce results on:

- Enhancing the MBPTA methodology: more reliable and robust statistical tests are under investigation. Applied to the EVT, they will increase confidence on pWCET estimates especially for lower criticality modes. The statistical power is used for evaluating reliability of statistical tests: it is applied to find the best goodness-of-fit test for pWCET estimation. Also, it is applied sensitivity analysis to pWCET estimates parameters to seek for the best pWCET distribution according to multiple criteria. The quality of pWCET estimates will be enhanced, that in turn it will reduce pessimism of MC task models.
- Improving representativity of input measurements: it is under investigation the notion of *representativity* of input measurements. When and how is a trace of measurements representative enough of some specific execution conditions? When is it representative of the worst-case conditions? What does it mean that the system hypotheses are represented well enough? In particular, the maximum domain of attraction of the EVT is under exploration in order to identify which are the system hypotheses and see if the EVT embeds them well into pWCET estimates. Also, it will be studied the problem of input coverage, where representativity is used for assuring to have included the worst-cases among the inputs. With an increased input representativity, the quality of MC timing models will increase.
- System behavior monitoring: identify which are the statistics that allow to characterize task condition changes, and then from the changes derive the best worst-case model that needs less measurements (reduce cost). The monitor will apply in order to reduce complexity of MBPTA, and to focus pWCET estimates to specific modes/execution conditions: the model pessimism is reduced. To the MC problem, the monitor will help separating modes (thus better models for each mode), and it will contribute representing the dependence between modes and other runtime system behaviors.

Also, it will be investigated the combination of deterministic timing analysis approaches (static timing analysis) and probabilistic timing analysis into the so called *hybrid timing analysis* approach. In the hybrid approach, probabilistic methods like MBPTA will allow evaluating the accuracy of static timing analysis: measuring are for validating static timing analysis hypotheses. The pessimism will be reduced with hypotheses that better cope with the behavior of the system. Besides, with a more precise system model from static timing analysis, input representativity for MBPTA will improve, and more accurate pWCET estimates would be computed. The coupling between MBPTA and static timing analysis is for increasing confidence on probabilistic models (MBPTA) and to reduce pessimism of deterministic models (static timing analysis).

The hybrid approach will also apply to characterize MC applications. Each tool will be used for a specific criticality level i.e., MBPTA for lower criticality modes and tasks, static timing analysis for higher criticality models.

The collaboration with IRIT Toulouse and Prof. Hugues Casse and Prof. Christine Rochange with the OTAWA static timing analysis tool developed by them is for developing the hybrid tool and explore its benefits.

Schedulability analysis for mixed criticality

With respect to schedulability analysis, the plan is to continue the investigation of deterministic approaches to MC schedulability. Collaborations are planned in order to continue releasing pessimistic and non-realistic hypotheses e.g. reduced number of HI-criticality tasks in HI-criticality mode. Also, new deterministic schedulability analysis are proposed in order to improve the resource usage within multi-core platforms.

Realistic hypothesis on system criticality mode will be applied and compared in terms of resource efficiency. This will be coupled with multi-core and many-core implementations, and the deterministic scheduling algorithms proposed will explore the trade-off between resource partitioning, resource sharing, and realistic behavior.

Also, the complexity of deterministic MC scheduling will be investigated: efficient deterministic MC scheduling policies will be proposed for the specific conditions considered (execution scenarios) and for the specific task allocation within the multiple cores available. Under investigation are better priority assignment for reduced scheduling complexity.

The ongoing collaboration with with Prof. Zhishan Guo and Prof. Kecheng Yang (Texas State University) will focus on relaxing the classical MC hypothesis for which once a HI-criticality task goes in HImode, every HI-criticality tasks goes on HImode. By doing that, we plan to contribute with more realistic conditions and less pessimistic scheduling results. New (k, n) definitions of system criticality will be explored, together with their realism and efficient resource usage.

In parallel, probabilistic MC scheduling approaches will be investigated. New probabilistic schedulers will be proposed such that they will be able to leverage probabilities into scheduling decisions: better decisions will improve resource utilization. Those contribution will follow the work done in [165]; they will focus on reducing some of its limitations, and by continuing developing efficient scheduling decisions which actively apply probabilities. In particular, the limitations in [165] will be overcome with adaptive and efficient servers that are able to capture decision instants and convenient runtime scheduling decisions.

Other ways to include probabilities into scheduling decisions will be investigated: the probabilistic models, that better cope with runtime behavior, will be applied to foresee the next decision, e.g. by considering the most probable event. More efficient resource usage and reduced pessimism will be achieved with better decisions from probabilities.

Collaborations with Prof. Zhishan Guo, PhD Konstantinos Bletsas, Prof. Laurent George, and other will allow producing more results for probabilistic MC problems.

The schedulability analysis, both deterministic and probabilistic, will have a key role in investigating complex problems such as MC for multi-core and many-core. With sensitivity analysis applied to schedulability, the idea is to explore effects that parameter changes would have on schedulability or other criteria that can be defined.

Deterministic sensitivity analysis like [190, 198] will continue to be studied in order to explore the multiple mode combinations that can happen within the system. With a complete and flexible deterministic representation of the runtime behavior, it will be qualified/quantified the impact of scheduling decisions and resource changes on criticality guarantees.

Probabilistic sensitivity analysis will continue to be developed and applied to study impact of system parameters have with MC schedulability [184, 191, 194]. There will be evaluated multiple trade-offs, e.g. predictability-resource usage and probabilistic guarantees. Also, it will be proposed enhanced strategies for more efficient mode changes and/or resource changes.

Finally, formal methods have been shown promising in representing MC system behavior and in taking scheduling decisions with probabilities [202, 204]. Such investigation will continue with the help of PhD Student Jasdeep Singh and Prof. Laura Carnevali (University of Florence). In particular, it will be applied Markov decision process with the reward mechanisms to be associated with probabilities, and there will be developed new schedulers and less pessimistic scheduling decisions than classical ones with the help of probabilities.

Jasdeep Singh is now investigating other tools to represent MC task executions. In particular, he is studying graphs and how they can be handy representing the task ordering. From graphs, the idea is to develop first off-line scheduling strategies that define best execution paths in order to increase the resource utilization. Once on-line, the strategies are to promptly adapt to mode changes and apply sub optimal strategies with similar objectives. The thesis aim at making graph and graph theory a formal approach for MC schedulability analysis, where everything is proved and validated.

In the literature, tasks are always assumed fully dependent of fully independent with respect to criticality levels and mode changes. If one HI-criticality task/job changes mode, either all the HI-criticality task/jobs are considered to change (full dependence), or each mode change has to be evaluated independently (full independence). In reality, things are more complex than that. There can exist dependence relationship among task/jobs, and their investigation can and will help reducing pessimism of actual results on MC schedulability analysis.

With Jasdeep Singh, future work is also devoted releasing the independence hypothesis between tasks. Different causes of dependence will be introduced in the Markov chain formalization with probabilities representing dependence effects. With this new model, scheduling decisions that account for dependence effects can be made and embedded into the MC scheduler. The thesis is currently investigating how to plug dependence models into formal methods, included graphs, and applied to MC schedulability analysis.

Partitioning guarantees and interference analysis

Partitioning guarantees and interference analysis are crucial for MC timing analysis and MC schedulability analysis. Moreover, they affect both academic and industry studies.

Figure 5.15 and Figure 5.16 are for outlining the importance of the partitioning problem to MC. Figure 5.15 describes an abstract case of partitioned system in which time and space partitioning is guaranteed with a separation kernel like real-time OSs or hypervisors. Figure 5.16 is from an example with the NXP T4240 multi-core platform with PikeOS hypervisor for partitioning, currently under investigation at the ONERA. In it, the partitioning is by cores, while cores are grouped by clusters; `C0` is the execution scenario with just one task running in core/partition 0, `C1_NC0_NC2_NC3` is the scenario with the task under observation $C1$ running in core 1 and other tasks NC running in other cores within the same cluster, and `C1_NC0_NC2_NC3_NC4` is the scenario with the task under observation running in core 1 and other tasks running in cores within the same cluster and in other clusters. Guaranteeing the isolation is complex; interference have to be studied, represented, and limited or canceled; for that, it is planned to apply the statistical analysis with measurements. The probabilistic models derived from the statistical analysis are for characterizing how good are the partitioning tools applied, and eventually for improving their effectiveness with better placement or scheduling choices.

Average (nominal) models, worst-case models, and statistics to identify behaviors, trends, hypothesis testing, etc. are proposed. All of it will be applied to measurements of execution times or other system parameters. The statistical analysis will also apply to qualify the partitioning applied for separating mixed critical applications.

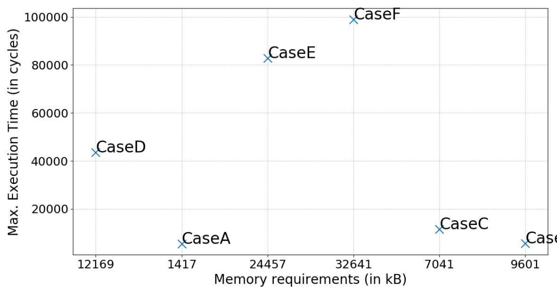


Figure 5.17: Interference effects from shared resources and different execution scenarios.

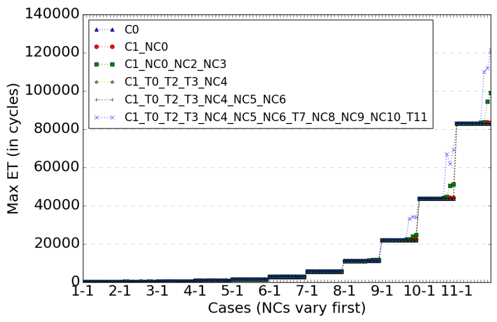


Figure 5.18: Maximum measured execution time under different resource requirements and under different execution scenarios.

Figure 5.17 motivates the need for interference modeling since it represents the effect on maximum measured execution time of a task under observation with respect to memory requirements of the task and its interfering tasks. Here the execution scenarios are named with CaseX. Interference from shared resources such as cache memories can impact 5 times task execution time (CaseF) with respect to the case where shared resources are not used (isolation case, CaseA), when resource usage increases only 3.5 times.

Figure 5.18, Figure 5.19, and Figure 5.20 illustrate some preliminary results of an on-going work for studying interference. The investigation is made with the NXP T4240 multi-core platform and PikeOS hypervisor. In it, different execution scenarios are considered; they are are different tasks and task placement within the cores and clusters which exercise interference with shared resources, e.g. `C1_NC0_NC2_NC3_NC4` is the scenario where the task under observation $C1$ is running in core 1, and other four tasks execute in four different cores (NC). Task executions are observed with measurements on execution time and the interference levels is parametrized since the tasks composing the application can change their memory requirements.

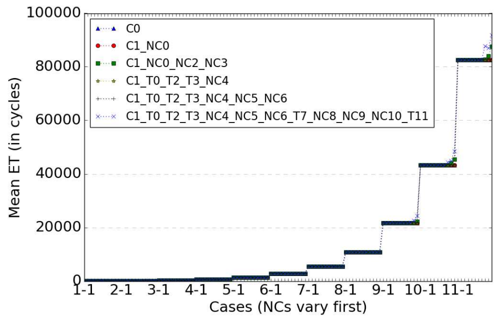


Figure 5.19: Average measured execution time from different execution scenarios and different resource requirements.

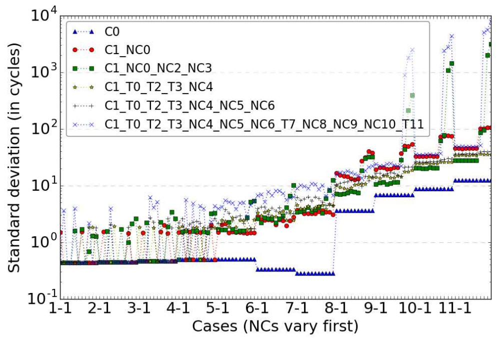


Figure 5.20: Standard deviation of measured execution time from different execution scenarios and different resource requirements.

The interference analysis is conducted with measurements and a formal statistical approach in which different metrics are applied to characterize the task execution times under different conditions. The statistical analysis is still under development, and applies for average and worst-case modeling of task execution times. It will be useful to qualify and quantify resource partitioning from hypervisor or operating systems mechanisms. Moreover, statistical analysis tackles with schedulability analysis and MC, since task placement and task ordering are among the parameters to be investigated.

In Figure 5.18 there are represented the maximum execution time of the task under observation with different memory requirement (in abscissa, with X-Y representing the memory requirements of the task under observation (X) and of the task interfering with it (Y)) at different scenarios. Figure 5.19 and Figure 5.20 illustrate other statistical metrics, respectively average and standard deviation, applied to characterize interference effects under different memory requirements and execution scenarios.

The statistical analysis will be coupled with task placement and scheduling policies to limit/remove interference and increase the predictability per criticality level. Such iterative approach will apply in order to evaluate placements/scheduling policies. The aim is proposing multiple policies and eventually converging toward optimal or sub-optimal solutions for the application or the implementations.

5.3 Future projects

There are hereby presented projects proposals under submission and future ideas around MC which will contribute to the research project defined. Among the project proposals, there are PhD thesis proposals, Postdoc offers, collaborations, and research or technology transfer. The tools formerly described, see Figure 4.1, are included in the comments since the idea is apply them in order to make contributions to the MC problem for multi-core and many-core implementations.

- An ANR "jeunes chercheuses et jeunes chercheurs" JCJC project proposal under submission. It is for investigating mixed critical system with multi-core and many-core implementations, as for the research project hereby presented. The project proposal is to define the research activity in the next four year around the MC problem. It tackles with timing analysis and schedulability analysis, with deterministic and probabilistic approaches, and for multi-core and many-core platforms.
  The JCJC project proposal is for building the collaboration network and propose breakthrough contributions for multi- and many-cores. The proposal includes industry and academic partners: Prof. Hugues Casse, Prof. Zhishan Guo, PhD Konstantinos Bletsas, Prof. Laura Canevali, and industrial partners such as AIRBUS, COBHAM Gaisler, ESA. It refines the research project described in this manuscript; steps and intermediate objectives with respect to timing analysis and schedulability analysis for MC are detailed. Timing analysis approaches and schedulers will be enhanced/newly developed for both academic and industry perspectives.
  The project proposal asks funding for a PhD thesis on probabilistic MC schedulability analysis with the use of sensitivity analysis techniques. The PhD candidate will investigate MC applications implemented with multi-core and many-core platforms and will apply sensitivity analysis in order to reduce pessimism and complexity with innovative scheduling algorithms.

- The ongoing collaboration with prof. Zhishan Guo, PhD Konstantinos Bletsas, and Prof. Kecheng Yang about scheduling problems for MC.
  The collaboration is for applying deterministic and probabilistic approaches with the objective of proposing alternative scheduling algorithms for more efficient resource utilization. Also, the Vestal's model [140] will be revised, and less pessimistic scheduling algorithms will be proposed. The collaboration will continue investigating MC scheduling with limited HI-criticality behaviors [166], and probabilistic MC scheduling algorithms like [165]. There is also under investigation a priority assignment problems for MC task scheduling, and how they can help defining less complex deterministic scheduling algorithms. The project will pursue only the academic perspective to MC.

- The third year of Thesis of PhD student Jasdeep Singh in 2019 is for further investigating the use of formal methods (and the advantages related to those) for more efficient probabilistic MC scheduling decisions. The collaboration with Prof. Laura Carnevali and Prof. Giuseppe Lipari (University of Sciences and Technologies of Lille) during this last year is for pursuing investigation and development of formal methods applied to MC scheduling. In particular, more realistic hypothesis will be applied to model and study dependent tasks. This project will pursue only the academic perspective to MC.

- The six years PEA DRONE project, 2019 – 2025. The project is for investigating different aspects of drone development and analysis. The PEA project is in collaboration with the ONERA, the ISAE and the ENAC in Toulouse.
  A work package of the project tackles with embedded system implementations for Drone applications, and in particular, it considers the use of GPU platforms. In the project there is financed a PhD thesis for studying GPU architecture for Drone MC applications. During the thesis the candidate will investigate safety problems and scheduling/placement problems related to MC applications. The PhD thesis is between the ONERA and the ISAE.
  Also, in the project there are the funding for a 18 months Postdoc for implementing MC applications with

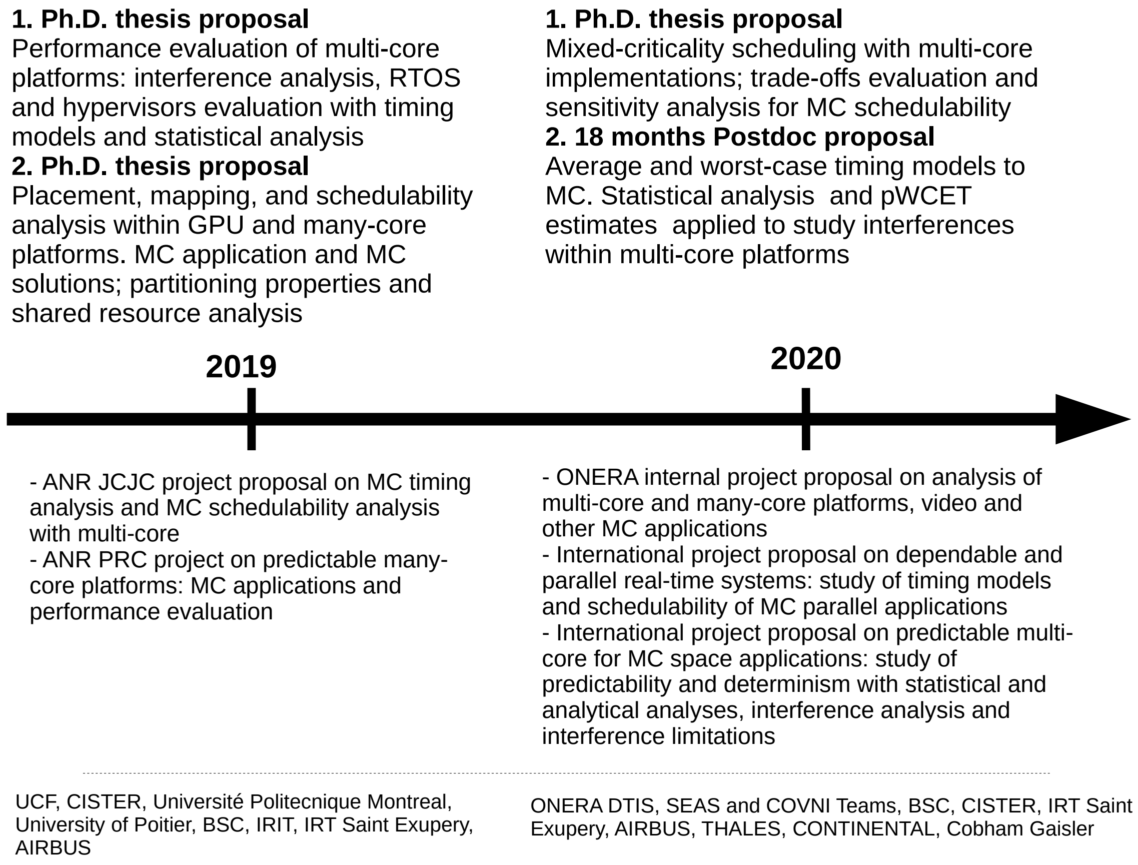
GPU platforms. During the implementation, it will be evaluated the determinism of GPU implementations with Drone MC applications; measurement-based approaches will be applied together with statistical analysis. Both PhD thesis and Postdoc have to finish by the project deadline in 2025; the PhD thesis goes first and then the Postdoc which is for validating thesis achievement. The project will pursue only the academic perspective to MC. Within the PEA, it will be close collaboration with PhD David Doose (ONERA), PhD Youcef Bouchebaba (ONERA), and PhD Jean-Baptiste Chaudron (ISAE Toulouse) on placement policies for MC applications, schedulability analysis of safety-critical as well as mixed critical applications, and determinism evaluation of MC applications on GPU platforms.

- A proposed PhD thesis starting from October 2019 to be financed ONERA-ISAE (shared research team ONERA-ISAE). The thesis if for studying interference of mixed critical applications within multi-core platforms. A clear understanding of system dynamics will help producing better timing models, and having better models is critical with today's and future multi-core implementations of MC real-time embedded systems. Here, representativity and flexibility issues are addressed in order to improve the representations of complex MC multi-core real-time embedded systems.
  The PhD candidate will apply statistical analysis based on measurements in order to identify and quantify interference effects on task execution times. Also, parameters like bus latency and cache misses will be applied for such characterization. The statistical analysis will embed all the parameters and will provide average (nominal) and worst-case models to interference and task execution times. The thesis will investigate both academic and industry perspectives to MC.

- An ANR project proposal "projet de recherche collaborative" PRC AAP 2019 on innovative parallel processors for real-time systems under submission. The proposal is for studying a many-core platform (Little Big Processor - LBP), its deterministic characteristics, and its performance with MC real-time applications.
  The project proposal is with academic partners from INRIA Lyon, INRIA Rennes, IRIT Toulouse, and University of Perpignan for studying and validating the determinism with such an innovative many-core architecture as the Little Big RPcessor. In particular, the ONERA is involved into the schedulability analysis of safety-critical and mixed critical applications, and performance evaluation for those.
  The proposal has funding for a 12 months Postdoc for performance evaluation. The interest for the project proposal and for proposing the Postdoc come from the need for studying and guaranteeing predictability with future implementations for real-time embedded systems such as many-core. MC applications will be studied, while sensitivity analysis will be applied to explore performance-predictability trade-offs. The project will pursue both academic and industry perspective to MC.

In the upcoming months other collaboration projects will be proposed in order to tackle with MC for multi-core and many-core platforms. With those, the many open problems with MC and the new architectures for real-time embedded systems will be explored with timing analysis and schedulability analysis tools. The pursuit of both industry and academic perspectives for MC will continue.

Figure 5.21 details some of the project proposal already mentioned with their topics. Also, are represented PhD thesis proposals, and Postdocs proposals with a brief description of interest for MC. PhD thesis and the Postdoc are for investigating more in details some of the topics in the projects: both timing analysis and schedulability analysis are approached by them. There are also enlisted existing and future collaborations related to such projects that will help working on MC with multi-core and many-core platforms.



1. Ph.D. thesis proposal
Performance evaluation of multi-core platforms: interference analysis, RTOS and hypervisors evaluation with timing models and statistical analysis
2. Ph.D. thesis proposal
Placement, mapping, and schedulability analysis within GPU and many-core platforms. MC application and MC solutions; partitioning properties and shared resource analysis

1. Ph.D. thesis proposal
Mixed-criticality scheduling with multi-core implementations; trade-offs evaluation and sensitivity analysis for MC schedulability
2. 18 months Postdoc proposal
Average and worst-case timing models to MC. Statistical analysis and pWCET estimates applied to study interferences within multi-core platforms

- ANR JCJC project proposal on MC timing analysis and MC schedulability analysis with multi-core
- ANR PRC project on predictable many-core platforms: MC applications and performance evaluation

- ONERA internal project proposal on analysis of multi-core and many-core platforms, video and other MC applications
- International project proposal on dependable and parallel real-time systems: study of timing models and schedulability of MC parallel applications
- International project proposal on predictable multi-core for MC space applications: study of predictability and determinism with statistical and analytical analyses, interference analysis and interference limitations

UCF, CISTER, Université Politecnique Montreal, University of Poitier, BSC, IRIT, IRT Saint Exupery, AIRBUS

ONERA DTIS, SEAS and COVNI Teams, BSC, CISTER, IRT Saint Exupery, AIRBUS, THALES, CONTINENTAL, Cobham Gaisler

Figure 5.21: Research perspective and projects under submission.

Bibliography

[1] IEC SC 65A. **Functional safety of electrical/electronic/programmable electronic safety-related systems**. Technical Report IEC 61508, The International Electrotechnical Commission, 1998.

[2] Jaume Abella, Eduardo Quiñones, Franck Wartel, Tullio Vardanega, and Francisco J. Cazorla. **Heart of Gold: Making the Improbable Happen to Increase Confidence in MBPTA**. In *26th Euromicro Conference on Real-Time Systems, (ECRTS)*, 2014.

[3] L. Abeni and G. Buttazzo. **QoS guarantee using probabilistic deadlines**. In *the 11th Euromicro Conference on Real-Time Systems (ECRTS'99)*, 1999.

[4] Luca Abeni and Giorgio Buttazzo. **Integrating Multimedia Applications in Hard Real-Time Systems**. In *Proceedings of the* 19th *IEEE Real-Time Systems Symposium*, pages 4–13, Madrid, Spain, dec 1998.

[5] Luca Abeni and Giorgio Buttazzo. **Adaptive Bandwidth Reservation for Multimedia Computing**. In *RTCSA '99: Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications*, page 70, Washington, DC, USA, 1999. IEEE Computer Society.

[6] Luca Abeni and Giorgio Buttazzo. **Hierarchical QoS Management for Time Sensitive Applications**. In *RTAS '01: Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01)*, page 63, Washington, DC, USA, 2001. IEEE Computer Society.

[7] Luca Abeni and Giorgio Buttazzo. **Resource Reservation in Dynamic Real-Time Systems**. *Real-Time Syst.*, **27**(2):123–167, 2004.

[8] Kunal Agrawal and Sanjoy Baruah. **Intractability Issues in Mixed-Criticality Scheduling**. In *ECRTS*, **106** of *LIPIcs*, pages 11:1–11:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[9] 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, November 2016.

[10] Bader Alahmad and Sathish Gopalakrishnan. **Risk-Aware Scheduling of Dual Criticality Job Systems Using Demand Distributions**. *LITES*, **5**(1):01:1–01:30, 2018.

[11] Sebastian Altmeyer, Liliana Cucu-Grosjean, and Robert I. Davis. **Static probabilistic timing analysis for real-time systems using random replacement caches**. *Real-Time Systems*, **51**(1):77–123, 2015.

[12] N. Audsley. **On priority assignment in fixed priority scheduling**. *Information Processing Letters*, **79**(1):39–44, 2001.

[13] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. **Applying New Scheduling Theory to Static Priority Pre-Emptive Scheduling**. *Software Engineering Journal*, **8**:284–292, 1993.

[14] N.C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. **Hard Real-Time Scheduling: The Deadline-Monotonic Approach**. In *in Proc. IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.

[15] Roberto Baldoni, Luca Montanari, and Marco Rizzuto. **On-line failure prediction in safety-critical systems**. *Future Generation Computer Systems*, **45**:123–132, 2015.

[16] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, Maurizio Peri, and Saverio Pezzini. **Fault-tolerant Platforms for Automotive Safety-critical Applications**. In *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, CASES '03, pages 170–177, New York, NY, USA, 2003. ACM.

[17] S. Baruah, A. Burns, and Z. Guo. **Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors**. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems*, 2016.

[18] S. Baruah, A. Easwaran, and Z. Guo. **MC-Fluid: Simplified and Optimally Quantified**. In *2015 IEEE Real-Time Systems Symposium*, pages 327–337, 2015.

[19] Sanjoy Baruah. **Mixed criticality schedulability analysis is highly intractable**, 2009.

[20] Sanjoy Baruah. **A scheduling model inspired by control theory**. In *Proceedings of the 6th International Real-Time Scheduling Open Problems Seminar*, 2015.

[21] Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Nicole Megow, and Leen Stougie. **Scheduling real-time mixed-criticality jobs**. In *Proceedings of the 35th international conference on Mathematical foundations of computer science*, MFCS, pages 90–101, 2010.

[22] Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne van der Ster, and Leen Stougie. **Preemptive Uniprocessor Scheduling of Mixed-Criticality Sporadic Task Systems**. *J. ACM*, **62**(2):14:1–14:33, 2015.

[23] Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Alberto Marchetti-Spaccamela, Suzanne van der Ster, and Leen Stougie. **Mixed-Criticality Scheduling of Sporadic Task Systems**. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 555–566, 2011.

[24] Sanjoy K. Baruah, Alan Burns, and Robert I. Davis. **Response-Time Analysis for Mixed Criticality Systems**. In *Proceedings of the 32nd IEEE Real-Time Systems Symposium, RTSS 2011, Vienna, Austria, November 29 - December 2, 2011*, pages 34–43, 2011.

[25] Sanjoy K. Baruah, Alan Burns, and Zhishan Guo. **Scheduling Mixed-Criticality Systems to Guarantee Some Service under All Non-erroneous Behaviors**. In *28th Euromicro Conference on Real-Time Systems, ECRTS*, pages 131–138, 2016.

[26] Sanjoy K. Baruah, Rodney R. Howell, and Louis E. Rosier. **Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor**. In *Real-Time System*, pages 301–324, 1990.

[27] Sanjoy K. Baruah, Haohan Li, and Leen Stougie. **Towards the Design of Certifiable Mixed-criticality Systems**. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010.

[28] Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. **Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor**. In *IEEE Real-Time Systems Symposium*, pages 182–190, 1990.

[29] Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. **Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor**. *Real-Time Systems*, **2**(4):301–324, 1990.

[30] Enrico Bini and Giorgio C. Buttazzo. **Schedulability Analysis of Periodic Fixed Priority Systems**. *IEEE Transactions on Computers 53 (11), pp. 1462-1473, November 2004*, pages 1462–1473, 2004.

[31] Enrico Bini and Giorgio C. Buttazzo. **The Space of EDF Feasible Deadlines**. In *19th Euromicro Conference on Real-Time Systems, ECRTS'07, 4-6 July 2007, Pisa, Italy, Proceedings*, pages 19–28, 2007.

[32] Enrico Bini, Marco Di Natale, and Giorgio C. Buttazzo. **Sensitivity analysis for fixed-priority real-time systems**. *Real-Time Systems*, **39**(1-3), 2008.

[33] P. Binns. **A robust high-performance time partitioning algorithm: the digital engine operating system (DEOS) approach**. In *20th DASC. 20th Digital Avionics Systems Conference (Cat. No.01CH37219)*, **1**, pages 1B6/1–1B6/12 vol.1, Oct 2001.

[34] James J. Buckley. **Fuzzy statistics: hypothesis testing**. *Soft Comput.*, **9**(7):512–518, 2005.

[35] A. Burns and R. Davis. **Mixed-criticality systems: A Review**. 2017. Available from: `http://www-users.cs.york.ac.uk/~burns/review.pdf+`.

[36] G. Buttazzo and L. Abeni. **Adaptive rate control through elastic scheduling**. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, **5**, pages 4883–4888 vol.5, 2000.

[37] Giorgio C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 1998.

[38] S. Chakraborty, S. Künzli, and L. Thiele. **A General Framework for Analysing System Properties in Platform-Based Embedded System Designs**. In *DATE*, pages 190–195, 2003.

[39] Chao Chen and Giovanni Beltrame. **An Adaptive Markov Model for the Timing Analysis of Probabilistic Caches**. *ACM Trans. Design Autom. Electr. Syst.*, **23**(1):12:1–12:24, 2017.

[40] Wei Chen, Sam Toueg, and Marcos Kawazoe Aguilera. **On the Quality of Service of Failure Detectors**. *IEEE Trans. Comput.*, **51**(5):561–580, May 2002.

[41] Tommaso Cucinotta, Luigi Palopoli, and Giuseppe Lipari. **FRESCOR Delivarable D-AQ2v2: Control Algorithms for Coordinated Resource-Level and Application-Level Adaptation v2**, 2008.

[42] Liliana Cucu-Grosjean. **Independence - a misunderstood property of and for (probabilistic) real-time systems**. Invited paper to the 60th birthday of A. Burns, 2013.

[43] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. **Uppaal SMC tutorial**. *International Journal on Software Tools for Technology Transfer*, **17**(4):397–415, Aug 2015.

[44] Robert I. Davis and Alan Burns. **Response Time Upper Bounds for Fixed Priority Real-Time Systems**. In *Proceedings of the 29th IEEE Real-Time Systems Symposium, RTSS 2008, Barcelona, Spain, 30 November - 3 December 2008*, pages 407–418, 2008.

[45] 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.

[46] D. de Niz, K. Lakshmanan, and R. Rajkumar. **On the Scheduling of Mixed-Criticality Real-Time Task Sets**. In *Real-Time Systems Symposium, RTSS. 30th IEEE*, pages 291 –300, 2009.

[47] Augusto Born de Oliveira, Eduardo Camponogara, and George Lima. **Dynamic Reconfiguration in Reservation-Based Scheduling: An Optimization Approach**. In *15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 173–182, 2009.

[48] José Luis Díaz, Daniel F. García, Kanghee Kim, Chang-Gun Lee, Lucia Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. **Stochastic Analysis of Periodic Real-Time Systems**. In *RTSS '02: Proceedings of the 23rd IEEE Real-Time Systems Symposium*, page 289, Washington, DC, USA, 2002. IEEE Computer Society.

[49] Stefan Draskovic, Pengcheng Huang, and Lothar Thiele. **On the Safety of Mixed-Criticality Scheduling**. In *WMC 2016*, Proceedings of WMC 2016, Porto, Portugal, November 2016.

[50] Arvind Easwaran. **Demand-Based Scheduling of Mixed-Criticality Sporadic Tasks on One Processor**. In *Proceedings of the IEEE 34th Real-Time Systems Symposium, RTSS 2013, Vancouver, BC, Canada, December 3-6, 2013*, pages 78–87, 2013.

[51] S. Edgar and A. Burns. **Statistical Analysis of WCET for Scheduling**. In *22nd of the IEEE Real-Time Systems Symposium*, 2001.

[52] Pontus Ekberg and Wang Yi. **Bounding and shaping the demand of generalized mixed-criticality sporadic task systems**. *Real-Time Systems*, **50**(1):48–86, 2014.

[53] P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling extremal events for insurance and finance*. Applications of mathematics. Springer, Berlin, Heidelberg, New York, 1997.

[54] Rolf Ernst, Alan Burns, Lothar Thiele, and Jimmy Le Rhun. **Mixed critical system design and analysis**. In *Proceedings of the 12th International Conference on Embedded Software, EMSOFT 2012, part of the Eighth Embedded Systems Week, ESWeek 2012, Tampere, Finland, October 7-12, 2012*, pages 247–248, 2012.

[55] Rolf Ernst and Marco Di Natale. **Mixed Criticality Systems - A History of Misconceptions?** *IEEE Design & Test*, **33**(5):65–74, 2016.

[56] X. Feng and Al. Mok. **A Model of Hierarchical Real-Time Virtual Resources**. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS 2002)*, pages 26–35, December 2002.

[57] Tom Fleming, Huang-Ming Huang, Alan Burns, Christopher D. Gill, Sanjoy Baruah, and Chenyang Lu. **Corrections to and Discussion of "Implementation and Evaluation of Mixed-criticality Scheduling Approaches for Sporadic Tasks"**. *ACM Trans. Embedded Comput. Syst.*, **16**(3):77:1–77:4, 2017. Available from: `http://doi.acm.org/10.1145/2974020`.

[58] Gerhard Fohler. **Changing Operational Modes in the Context of Pre Run-Time Scheduling (Special Issue on Responsive Computer Systems)**. *IEICE transactions on information and systems*, **76**(11):1333–1340, 1993.

[59] Radio Technical Commission for Aeronautics. **Software Considerations in Airborne Systems and Equipment Certification**, 1994.

[60] Radio Technical Commission for Aeronautics. **Software Considerations in Airborne Systems and Equipment Certification**, 2011.

[61] Lipari G. and E. Bini. **Resource Partitioning among Real-Time Applications**. In *ECRTS'03, IEEE Computer Society*, pages 151–158, 2003.

[62] Juan Carlos Figueroa García and José Jairo Soriano Méndez. **A Fuzzy Logic Approach to Test Statistical Hypothesis on Means**. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, 4th International Conference on Intelligent Computing, ICIC 2008, Shanghai, China, September 15-18, 2008, Proceedings*, pages 316–325, 2008.

[63] M. Gardner and J. Liu. **Analyzing stochastic fixed-priority real-time systems**. In *the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, 1999.

[64] Cijo George and Sathish S. Vadhiyar. **ADFT: An Adaptive Framework for Fault Tolerance on Large Scale Systems using Application Malleability**. *Procedia Computer Science*, **9**:166 – 175, 2012.

[65] L. George and J.F. Hermant. **Characterization of the Space of Feasible Worst-Case Execution Times for Earliest-Deadline-First scheduling**. *Journal of Aerospace Computing, Information and Communication (JACIC)*, **6**(11), 2009.

[66] Laurent George and Jean-François Hermant. **Characterization of the Space of Feasible Worst-Case Execution Times for Earliest-Deadline-First Scheduling**. *JACIC*, **6**(11):604–623, 2009.

[67] Oliver Gettings, Sophie Quinton, and Robert I. Davis. **Mixed criticality systems with weakly-hard constraints**. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, 2015.

[68] P. Graydon and I. Bate. **Safety Assurance Driven Problem Formulation for Mixed-Criticality Scheduling**. In *Proceedings of the Workshop on Mixed-Criticality Systems*, pages 19–24, 2013.

[69] Patrick J. Graydon and Iain Bate. **Realistic Safety Cases for the Timing of Systems**. *Comput. J.*, **57**(5):759–774, 2014.

[70] Chuancai Gu, Nan Guan, Qingxu Deng, and Wang Yi. **Improving OCBP-based scheduling for mixed-criticality sporadic task systems**. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2013, Taipei, Taiwan, August 19-21, 2013*, pages 247–256, 2013.

[71] Qian Guangming. **An earlier time for inserting and/or accelerating tasks**. *Real-Time Systems*, 2009.

[72] E.J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958.

[73] Zhishan Guo and Sanjoy Baruah. **Mixed-criticality scheduling upon varying-speed multiprocessors**. In *Dependable, Autonomic and Secure Computing (DASC), 12th International Conference on*, pages 237–244. IEEE, 2014.

[74] Zhishan Guo and Sanjoy Baruah. **The concurrent consideration of uncertainty in WCETs and processor speeds in mixed-criticality systems**. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems (RTNS)*, pages 247–256. ACM, 2015.

[75] Zhishan Guo, Sai Sruti, Bryan C. Ward, and Sanjoy Baruah. **Sustainability in Mixed-Criticality Scheduling**. In *2017 IEEE Real-Time Systems Symposium, RTSS 2017, Paris, France, December 5-8, 2017*, pages 24–33, 2017.

[76] Maggie Hamill. **Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system**. *Software Quality Journal*, **23**(2):229–265, 2015.

[77] J. Hansen, S. Hissam, , and G. Moreno. **Statistical- based WCET estimation and validation**. In *the 9th International Workshop on Worst-Case Execution Time*, 2009.

[78] J. Hansen, J. Lehoczky, H. Zhu, and R. Rajkumar. **Quantized EDF scheduling in a stochastic enviroment**. In *the 16th IEEE International Parallel and Distributed Processing Symposium (IPDPS'02)*, 2002.

[79] Michael Gonzalez Harbour, Daniel Sangorren, and Miguel Tellerea de Esteban. **FRESCOR Delivarable D-AT2: Schedulability analysis techniques for distributed systems**, 2009.

[80] Damien Hardy and Isabelle Puaut. **STATIC PROBABILISTIC WORST CASE EXECUTION TIME ESTIMATION FOR ARCHITECTURES WITH FAULTY INSTRUCTION CACHES**. In *International Conference on Real-Time Networks and Systems (RTNS)*, 2013.

[81] IEC. **Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems**.

[82] ISO. **Road vehicles – Functional safety**, 2011.

[83] Yuming Jiang. **A basic stochastic network calculus**. *SIGCOMM Comput. Commun. Rev.*, **36**(4), 2006.

[84] Y. Kato, M. Takahashi, R. Ohtsuki, and S. Yamaguchi. **A proposal of fuzzy test for statistical hypothesis**. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, **4**, pages 2929–2934 vol.4, 2000.

[85] Angeliki Kritikakou, Claire Pagetti, Olivier Baldellon, Matthieu Roy, and Christine Rochange. **Run-Time Control to Increase Task Parallelism In Mixed-Critical Systems**. In *26th Euromicro Conference on Real-Time Systems, ECRTS 2014, Madrid, Spain, July 8-11, 2014*, pages 119–128, 2014.

[86] M. Kwiatkowska, G. Norman, and D. Parker. **Stochastic Model Checking**. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation SFM*, **4486** of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.

[87] M. Kwiatkowska, G. Norman, and D. Parker. **PRISM 4.0: Verification of Probabilistic Real-time Systems**. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification CAV*, LNCS. Springer, 2011.

[88] Zhiling Lan and Yawei Li. **Adaptive Fault Management of Parallel Applications for High-Performance Computing**. *IEEE Trans. Comput.*, **57**(12):1647–1660, December 2008.

[89] J. Y. Le Boudec and P. Thiran. *Network calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer-Verlag New York, Inc., 2001.

[90] M Ross Leadbetter, Georg Lindgren, and Holger Rootzén. **Conditions for the convergence in distribution of maxima of stationary normal processes**. *Stochastic Processes and their Applications*, **8**(2):131–139, 1978.

[91] MR Leadbetter. **On a basis for Peaks over Threshold modeling**. *Statistics & Probability Letters*, **12**(4):357–362, 1991.

[92] Chang-Gun Lee, Joosun Hahn, Yang-Min Seo, Sang Lyul Min, Rhan Ha, Seongsoo Hong, Chang Yun Park, Minsuk Lee, and Chong Sang Kim. **Analysis of Cache-Related Preemption Delay in Fixed-Priority Preemptive Scheduling**. *IEEE Trans. Comput.*, **47**(6):700–713, June 1998.

[93] J. Lehoczky. **Real-time queueing theory**. In *the 17th IEEE Real-Time Systems Symposium (RTSS'96)*, 1996.

[94] John P. Lehoczky, Lui Sha, and Y. Ding. **The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior**. In *IEEE Real-Time Systems Symposium*, pages 166–171, 1989.

[95] John P. Lehoczky, Lui Sha, and Jay K. Strosnider. **Enhanced Aperiodic Responsiveness in Hard Real-Time Environments**. In *Proceedings of the IEEE Real-Time System Symposium (RTSS 1987)*, December 1997.

[96] J.P. Lehoczky. **Real-Time Queueing Theory**. In *10th of the IEEE Real-Time Systems Symposium (RTSS)*, 1996.

[97] J.P. Lehoczky, L. Sha, and Y. Ding. **The rate monotonic sheduling algorithm: exact characterization and average case bahavior**. *Proceedings of the IEEE Real-Time Systems Symposium*, 1989.

[98] Benjamin Lesage, David Griffin, Sebastian Altmeyer, and Robert I. Davis. **Static Probabilistic Timing Analysis for Multi-path Programs**. In *2015 IEEE Real-Time Systems Symposium, RTSS 2015, San Antonio, Texas, USA, December 1-4, 2015*, pages 361–372, 2015.

[99] J. Leung and J. W. Withehead. **On the complexity of fixed priority scheduling of periodic real-time tasks.** *Performance Evaluation*, **2**(4), 1982.

[100] J. Li, D. Ferry, S. Ahuja, K. Agrawal, C. Gill, and C. Lu. **Mixed-Criticality Federated Scheduling for Parallel Real-Time Tasks**. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–12, April 2016.

[101] 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.

[102] C. L. Liu and James W. Layland. **Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment**. *Journal of the ACM*, **20**(1):46–61, 1973.

[103] Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. **EDF-VD Scheduling of Mixed-Criticality Systems with Degraded Quality Guarantees**. In *Proceedings of the 37th IEEE Real-Time Systems Symposium*, 2016.

[104] José María López, José Luis Díaz, Joaquín Entrialgo, and Daniel F. García. **Stochastic analysis of real-time systems under preemptive priority-driven scheduling**. *Real-Time Systems*, **40**(2):180–207, 2008.

[105] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean. **A statistical response-time analysis of real-time embedded systems**. In *the 33rd IEEE Real-Time Systems Symposium (RTSS'12)*, 2012.

[106] Claire Maiza, Hamza Rihani, Juan M. Rivas, Joel Goossens, Sebastian Altmeyer, and Robert I. Davis. **A Survey of Timing Verification Techniques for Multi-Core Real-Time Systems**. VERIMAG Report TR-2018-9, Verimag, 2018.

[107] S. Manolache, P. Eles, and Z. Peng. **Schedulability analysis of applications with stochastic task execution times**. *ACM Transactions on Embedded Computing Systems (TECS)*, **3**(4):706–735, 2004.

[108] Alexander Maxiaguine, Simon Künzli, 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*, page 21040, Washington, DC, USA, 2004. IEEE Computer Society.

[109] D. Maxim and L. Cucu-Grosjean. **Response time analysis for fixed-priority tasks with multiple probabilistic parameters**. In *the 34th IEEE Real-Time Systems Symposium (RTSS)*, 2013.

[110] Dorin Maxim, Robert I. Davis, Liliana Cucu-Grosjean, and Arvind Easwaran. **Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling**. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS*, 2017.

[111] Dorin I Maxim, Robert Davis, Liliana Cucu-Grosjean, and Arvind Easwaran. **Probabilistic Analysis for Mixed Criticality Scheduling with SMC and AMC**. In *WMC 2016 - 4th International Workshop on Mixed Criticality Systems*, Porto, Portugal, November 2016.

[112] C. Mercer, R. Rajkumar, and J. Zelenka. **Temporal protection in real-time operating systems**. In *Real-Time Operating Systems and Software, 1994. RTOSS '94, Proceedings., 11th IEEE Workshop on*, pages 79–83, May 1994.

[113] Suzana Milutinovic, Jaume Abella, and Francisco J. Cazorla. **Modelling Probabilistic Cache Representativeness in the Presence of Arbitrary Access Patterns**. In *19th IEEE International Symposium on Real-Time Distributed Computing, ISORC 2016, York, United Kingdom, May 17-20, 2016*, pages 142–149, 2016.

[114] Vincent Nelis, Joel Goossens, and Bjorn Andersson. **Two Protocols for Scheduling Multi-mode Real-Time Systems upon Identical Multiprocessor Platforms**. In *ECRTS '09: Proceedings of the 2009 21st Euromicro Conference on Real-Time Systems*, pages 151–160, Washington, DC, USA, 2009. IEEE Computer Society.

[115] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[116] Risat Mahmud Pathan. **Fault-tolerant and real-time scheduling for mixed-criticality systems**. *Real-Time Systems*, **50**, 2014.

[117] P. Pedro and A. Burns. **Schedulability Analysis for Mode Changes in Flexible Real-Time Systems**. In *ECRTS*, pages 172–179, 1998.

[118] S. Perathoner, N. Stoimenov, and L. Thiele. **Reliable Mode Changes in Real-Time Systems with Fixed Priority or EDF Scheduling**. TIK Report 292, Computer Engineering and Networks Laboratory, ETH Zurich, ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-292.pdf, September 2008.

[119] Linh T. X. Phan, Insup Lee, and Oleg Sokolsky. **Compositional Analysis of Multi-mode Systems**. In *ECRTS '10: Proceedings of the 2010 22nd Euromicro Conference on Real-Time Systems*, pages 197–206, Washington, DC, USA, 2010. IEEE Computer Society.

[120] Miguel Piera-Martinez. *Modélisation des comportements extrêmes en ingénierie*. Theses, Université Paris Sud - Paris XI, September 2008.

[121] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1st edition, 1994.

[122] Eduardo Quinones, Emery D. Berger, Guillem Bernat, and Francisco J. Cazorla. **Using Randomized Caches in Probabilistic Real-Time Systems**. In *22nd Euromicro Conference on Real-Time Systems (ECRTS)*, pages 129–138. IEEE, 2009.

[123] S. Ramanathan and A. Easwaran. **MC-Fluid: rate assignment strategies**. In *WMC workshop at RTSS*, 2015.

[124] J. Real and A. Crespo. **Mode Change Protocols for Real-Time Systems: A Survey and a New Proposal**. *Real-Time Systems*, **26**(2):161–197, 2004.

[125] Indranil Saha, Sanjoy Baruah, and Rupak Majumdar. **Dynamic scheduling for networked control systems**. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC*. ACM Press, 2015.

[126] Carl Scarrott and Anna MacDonald. **A REVIEW OF EXTREME VALUE THRESHOLD ESTIMATION AND UNCERTAINTY QUANTIFICATION**. *REVSTAT–Statistical Journal*, **10**(1):33–60, 2012.

[127] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham. **Mode change protocols for priority-driven preemptive scheduling**. *Real-Time Systems*, **1**(3):243–264, 1989.

[128] Insik Shin and Insup Lee. **Periodic Resource Model for Compositional Real-Time Guarantees**. In *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, pages 2–13, Cancun, Mexico, December 2003.

[129] Insik Shin and Insup Lee. **Periodic Resource Model for Compositional Real-Time Guarantees**. In *RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium*, page 2, Washington, DC, USA, 2003. IEEE Computer Society.

[130] Youngsoo Shin, Daehong Kim, and Kiyoung Choi. **Schedulability-driven performance analysis of multiple mode embedded real-time systems**. In *DAC '00: Proceedings of the 37th Annual Design Automation Conference*, pages 495–500, New York, NY, USA, 2000. ACM.

[131] B. Sprunt, L. Sha, and J. P. Lehoczky. **Aperiodic Task Scheduling for Hard Real-Time Systems**. *Journal of Real-time Systems*, 1989.

[132] Marielle Stoelinga. **An introduction to probabilistic automata**. *Bulletin of the European Association for Theoretical Computer Science*, pages 176–198, 2002.

[133] N. Stoimenov, S. Perathoner, and L. Thiele. **Reliable Mode Changes in Real-Time Systems with Fixed Priority or EDF Scheduling**. In *DATE*, 2009.

[134] L. Thiele, S. Chakraborty, and M. Naedele. **Real-time calculus for scheduling hard real-time systems**. In *ISCAS*, **4**, pages 101–104, 2000.

[135] L. Thiele, E. Wandeler, and N. Stoimenov. **Real-Time Interfaces for Composing Real-Time Systems**. In *EMSOFT*, pages 34–43, 2006.

[136] T. Tia, Z. Deng, M. Storch, J. Sun, L. Wu, and J. Liu. **Probabilistic performance guarantee for real-time tasks with varying computation times**. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'95)*, 1995.

[137] T.S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.C. Wu, and J.S Liu. **Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times**. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 1995.

[138] K. W. Tindell, A. Burns, and A. J. Wellings. **Mode changes in priority pre-emptively scheduled systems**. In *RTSS*, pages 100–109, 1992.

[139] Marisol Garcia Valls, Alejandro Alonso, and Juan A. de la Puente. **Mode Change Protocols for Predictable Contract-Based Resource Management in Embedded Multimedia Systems**. *Embedded Software and Systems, Second International Conference on*, **0**:221–230, 2009.

[140] 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, pages 239–243. IEEE Computer Society, 2007.

[141] Ernesto Wandeler and Lothar Thiele. **Real-Time Calculus (RTC) Toolbox**. http://www.mpa.ethz.ch/Rtctoolbox, 2006. Available from: `http://www.mpa.ethz.ch/Rtctoolbox`.

[142] Franck Wartel, Leonidas Kosmidis, Code Lo, Benoit Triquet, Eduardo Quiñones, Jaume Abella, Adriana Gogonel, Andrea Baldovin, Enrico Mezzetti, Liliana Cucu, Tullio Vardanega, and Francisco J. Cazorla. **Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study**. In *8th IEEE International Symposium on Industrial Embedded Systems, SIES*, 2013.

[143] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. **The worst-case execution-time problem – overview of methods and survey of tools**. *ACM Transactions on Embedded Computing Systems (TECS)*, **7**(3):1–53, 2008.

[144] H. C. Wong and A. Burns. **Schedulability Analysis for the Abort-and-Restart (AR) Model**. In *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems (RTNS)*, 2014.

[145] Jing Xie and Yuming Jiang. **Stochastic Network Calculus Models under Max-Plus Algebra**. In *Proceedings of the Global Communications Conference (GLOBECOM)*, pages 1–6, 2009.

[146] S. Zhou. **An efficient simulation algorithm for cache of random replacement policy**. In *the 7th IFIP international conference on Network and parallel computing (NPC2010)*, pages 144–154, 2010.

[147] H. Zhu, J. Hansen, J. Lehoczky, and R. Rajkumar. **Optimal partitioning for quantized EDF scheduling**. In *the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*, 2002.

Own references

[148] Kostiantyn Berezovskyi, Fabrice Guet, Luca Santinelli, Konstantinos Bletsas, and Eduardo Tovar. Measurement-based probabilistic timing analysis for graphics processor units. In *Architecture of Computing Systems - ARCS 2016 - 29th International Conference, Nuremberg, Germany, April 4-7, 2016, Proceedings*, pages 223–236, 2016.

[149] Kostiantyn Berezovskyi, Luca Santinelli, Konstantinos Bletsas, and Eduardo Tovar. WCET measurement-based and extreme value theory characterisation of CUDA kernels. In *22nd International Conference on Real-Time Networks and Systems, RTNS '14, Versaille, France, October 8-10, 2014*, page 279, 2014.

[150] Marc Boyer, Guillaume Dufour, and Luca Santinelli. Continuity for network calculus. In *21st International Conference on Real-Time Networks and Systems, RTNS 2013, Sophia Antipolis, France, October 17-18, 2013*, pages 235–244, 2013.

[151] Giorgio C. Buttazzo and Luca Santinelli. Adaptive mechanisms for component-based real-time systems. In *2015 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2015, Montreal, QC, Canada, June 15-18, 2015*, pages 1–8, 2015.

[152] Laura Carnevali, Alessandra Melani, Luca Santinelli, and Giuseppe Lipari. Probabilistic deadline miss analysis of real-time systems using regenerative transient analysis. In *22nd International Conference on Real-Time Networks and Systems, RTNS '14, Versaille, France, October 8-10, 2014*, page 299, 2014.

[153] Francisco J. Cazorla, Eduardo Quiñones, Tullio Vardanega, Liliana Cucu, Benoit Triquet, Guillem Bernat, Emery D. Berger, Jaume Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. Proartis: Probabilistically analyzable real-time systems. *ACM Trans. Embedded Comput. Syst.*, 12(2s):94, 2013.

[154] Chao Chen, Luca Santinelli, Jérôme Hugues, and Giovanni Beltrame. Static probabilistic timing analysis in presence of faults. In *11th IEEE Symposium on Industrial Embedded Systems, SIES 2016, Krakow, Poland, May 23-25, 2016*, pages 149–158, 2016.

[155] Nastasi Christian, Marinoni Mauro, Santinelli Luca, Pagano Paolo, Lipari Giuseppe, and Franchino Gianluca. Baccarat: a dynamic real-time bandwidth allocationpolicy for ieee 802.15.4. In *Proceedings of IEEE Percom 2010, International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2010)*, Mannheim, Germany, 2010.

[156] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. Cazorla. Measurement-based probabilistic timing analysis for multi-path programs. In *the 24th Euromicro Conference on Real-Time Systems ECRTS*, 2012.

[157] Corentin Damman, Gregory Edison, Fabrice Guet, Eric Noulard, Luca Santinelli, and Jérôme Hugues. Architectural performance analysis of FPGA synthesized LEON processors. In *2016 International Symposium on Rapid System Prototyping, RSP 2016, Pittsburg, PA, USA, October 6-7, 2016*, pages 33–40, 2016.

[158] R. I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. *Proceedings of the 25th IEEE Euromicro Conference on Real-Time Systems (ECRTS'13)*, 2013.

[159] Samuel Jimenez Gil, Iain Bate, George Lima, Luca Santinelli, Adriana Gogonel, and Liliana Cucu-Grosjean. Open challenges for probabilistic measurement-based worst-case execution time. *Embedded Systems Letters*, 9(3):69–72, 2017.

[160] Nicolas Gobillot, David Doose, Charles Lesire, and Luca Santinelli. Periodic state-machine aware real-time analysis. In *20th IEEE Conference on Emerging Technologies & Factory Automation, ETFA 2015, Luxembourg, September 8-11, 2015*, pages 1–8, 2015.

[161] Nicolas Gobillot, Fabrice Guet, David Doose, Christophe Grand, Charles Lesire, and Luca Santinelli. Measurement-based real-time analysis of robotic software architectures. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 3306–3311, 2016.

[162] F. Guet, L. Santinelli, and J. Morio. On the reliability of the probabilistic worst-case execution time estimates. In *8th European Congress on Embedded Real Time Software and Systems (ERTS)*, 2016.

[163] Fabrice Guet, Luca Santinelli, and Jérôme Morio. On the representativity of execution time measurements: Studying dependence and multi-mode tasks. In *17th International Workshop on Worst-Case Execution Time Analysis, WCET 2017, June 27, 2017, Dubrovnik, Croatia*, pages 3:1–3:13, 2017.

[164] Fabrice Guet, Luca Santinelli, Jérôme Morio, Guillaume Phavorin, and Eric Jenn. Toward contention analysis for parallel executing real-time tasks. In *18th International Workshop on Worst-Case Execution Time Analysis, WCET 2018, July 3, 2018, Barcelona, Spain*, pages 4:1–4:13, 2018.

[165] Zhishan Guo, Luca Santinelli, and Kecheng Yang. EDF schedulability analysis on mixed-criticality systems with permitted failure probability. In *21th IEEE International Conference on Embedded and Real-Time Computing System and Applications*, 2015.

[166] Zhishan Guo, Luca Santinelli, and Kecheng Yang. Mixed-criticality scheduling with limited hi-criticality behaviors. In *Dependable Software Engineering. Theories, Tools, and Applications - 4th International Symposium, SETTA 2018, Beijing, China, September 4-6, 2018, Proceedings*, pages 187–199, 2018.

[167] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele, and Giorgio C. Buttazzio. Adaptive dynamic power management for hard real-time systems. In *the 30th IEEE Real-Time Systems Symposium (RTSS)*, pages 23–32, Washington D.C. U.S., 2009.

[168] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele, and Giorgio C. Buttazzio. Periodic power management schemes for real-time event streams. In *the 48th IEEE Conf. on Decision and Control (CDC)*, pages 6224–6231, Shanghai, China, 2009.

[169] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele, and Giorgio C. Buttazzio. Adaptive power management for real-time event streams. In *the 15th IEEE Conf. on Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 7–12, 2010.

[170] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele, and Giorgio C. Buttazzo. Adaptive dynamic power management for hard real-time streams. *Real-Time Systems Journal (submitted)*, 2010.

[171] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele, and Giorgio C. Buttazzo. Periodic power management schemes for real-time event streams. *EURASIP Journal on Embedded Systems (submitted)*, 2010.

[172] Dawood Khan, Bilel Nefzi, Luca Santinelli, and Ye-Qiong Song. Probabilistic bandwidth assignment in wireless sensor networks. In *Wireless Algorithms, Systems, and Applications - 7th International Conference, WASA 2012, Yellow Mountains, China, August 8-10, 2012. Proceedings*, pages 631–647, 2012.

[173] Dawood Khan, Luca Santinelli, and Liliana Cucu-Grosjean. Modeling uncertainties in safety-critical real-time systems: A probabilistic component-based analysis. In *7th IEEE International Symposium on Industrial Embedded Systems, SIES 2012, Karlsruhe, Germany, June 20-22, 2012*, pages 166–175, 2012.

[174] Tomasz Kloda, Bruno d'Ausbourg, and Luca Santinelli. EDF schedulability analysis for an extended timing definition language. In *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems, SIES 2014, Pisa, Italy, June 18-20, 2014*, pages 30–40, 2014.

[175] Tomasz Kloda, Bruno d'Ausbourg, and Luca Santinelli. Towards EDF schedulability analysis of an extended timing definition language. *SIGBED Review*, 11(3):44–49, 2014.

[176] Tomasz Kloda, Bruno d'Ausbourg, and Luca Santinelli. EDF schedulability test for the E-TDL time-triggered framework. In *11th IEEE Symposium on Industrial Embedded Systems, SIES 2016, Krakow, Poland, May 23-25, 2016*, pages 73–82, 2016.

[177] D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, and R. Davis. Optimal priority assignment algorithms for probabilistic real-time systems. In *the 19th International Conference on Real-Time and Networked Systems (RTNS'11)*, 2011.

[178] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean. Re-sampling for statistical timing analysis of real-time systems. In *the 20th International Conference on Real-Time and Network Systems (RTNS'12)*, 2012.

[179] D. Maxim, L. Santinelli, and Liliana Cucu-Grosjean. Improved sampling for statistical timing analysis of real-time systems. *Proceedings of the 4th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2010), joined to RTNS 2010*, 2010.

[180] A. Melani, E. Noulard, and L. Santinelli. Learning from probabilities: Dependences within real-time systems. In *the 8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'13)*, 2013.

[181] Anna Montesanto, Guido Tascini, Paola Baldassarri, and Luca Santinelli. Analysis of fingerprints through a reactive agent. In Gianfranco Minati, Eliano Pessa, and Mario Abram, editors, *Systemics of Emergence: Research and Development*, pages 93–104, Boston, MA, 2006. Springer US.

[182] L. Santinelli, J. Morio, G. Dufour, and D. Jacquemart. On the sustainability of the extreme value theory for WCET estimation. In *the 14th International Workshop on Worst-Case Execution Time Analysis (WCET'14)*, 2014.

[183] Luca Santinelli. Probabilistic component-based analysis for networks: invited paper. *SIGBED Review*, 13(3):65–72, 2016.

[184] Luca Santinelli. How effective is sensitivity analysis with probabilistic models? In *The 9th Real-Time Scheduling Open Problems Seminar at ECRTS*, 2018.

[185] Luca Santinelli, Giorgio Buttazzo, and Enrico Bini. Multi-moded resource reservation. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2011.

[186] Luca Santinelli, Giorgio C. Buttazzo, and Enrico Bini. Multi-moded resource reservations. In *17th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2011, Chicago, Illinois, USA, 11-14 April 2011*, pages 37–46, 2011.

[187] Luca Santinelli, Mangesh Chitnis, Christian Nastasi, Fabio Checconi, Giuseppe Lipari, and Paolo Pagano. A component-based architecture for adaptive bandwidth allocation in wireless sensor networks. In *IEEE Symposium on Industrial Embedded Systems (SIES)*, 2010.

[188] Luca Santinelli and Liliana Cucu-Grosjean. Toward probabilistic real-time calculus. In *3rd CRTS) with the 31st IEEE Real-Time Systems Symposium*, 2010.

[189] Luca Santinelli and Liliana Cucu-Grosjean. A probabilistic calculus for probabilistic real-time systems. *ACM Transactions on Embedded Computing Systems*, 14(3), 2015.

[190] Luca Santinelli, David Doose, Guy Durrieu, Frédéric Boniol, Charles Lesire-Cabaniols, and Christophe Grand. Schedulability analysis for mixed critical cyber physical systems. In *IEEE Industrial Cyber-Physical Systems, ICPS 2018, Saint Petersburg, Russia, May 15-18, 2018*, pages 297–303, 2018.

[191] 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.

[192] 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*, pages 199–208, 2017.

[193] Luca Santinelli and Zhishan Guo. On the criticality of probabilistic worst-case execution time models. In *Dependable Software Engineering. Theories, Tools, and Applications - Third International Symposium, SETTA 2017, Changsha, China, October 23-25, 2017, Proceedings*, pages 59–74, 2017.

[194] Luca Santinelli, Zhishan Guo, and Laurent George. Fault-aware sensitivity analysis for probabilistic real-time systems. In *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2016*, pages 69–74, 2016.

[195] Luca Santinelli, Giuseppe Lipari, and Paolo Ancilotti. Extending real-time calculus to hierarchical scheduling of real-time components. In *Proc. on 1st Workshop on Compositional Theory and Technology for Real-Time Embedded Systems CRTS'08*, 2008.

[196] Luca Santinelli, Mauro Marinoni, Francesco Prosperi, Francesco Esposito, Gianluca Franchino, and Giorgio Buttazzo. Energy-aware packet and task co-scheduling for embedded systems. In *International Conference On Embedded Software (EMSOFT)*, 2010.

[197] Luca Santinelli, Patrick Meumeu Yomsy, Dorin Maxim, and Liliana Cucu-Grosjean. A component-based framework for modeling and analysing probabilistic real-time systems. In *16th IEEE International Conference on Emerging Technologies and Factory Automation*, 2011.

[198] Luca Santinelli and Guo Zhishan. A sensitivity analysis for mixed criticality: Trading criticality with computational resource. In *the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2018.

[199] Andreas Schranzhofer, Jian-Jia Chen, Luca Santinelli, and Lothar Thiele. Dynamic and adaptive allocation of applications on mpsoc platforms. In *Proceedings of the 15th Asia South Pacific Design Automation Conference, ASP-DAC 2010, Taipei, Taiwan, January 18-21, 2010*, pages 885–890, 2010.

[200] Jasdeep Singh, Zhishan Guo, Luca Santinelli, Guillaume Infantes, David Doose, and Julien Brunel. Use of probabilities and formal methods to control system criticality levels. In *The 9th Real-Time Scheduling Open Problems Seminar at ECRTS*, 2018.

[201] Jasdeep Singh, Luca Santinelli, and Guillaume Infantes. Backlog estimation for suspended probabilistic tasks released synchronously. In *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2017.

[202] Jasdeep Singh, Luca Santinelli, and Guillaume Infantes. Rtprob - real time probabilsitic tool for probabilistic schedulability analysis using markov chain. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2018.

[203] Jasdeep Singh, Luca Santinelli, Guillaume Infantes, David Doose, and Julien Brunel. Markov chain modeling of probabilistic real-time systems. In *The 8th Real-Time Scheduling Open Problems Seminar at ECRTS*, 2017.

[204] Jasdeep Singh, Luca Santinelli, Guillaume Infantes, David Doose, and Julien Brunel. Mixed criticality probabilistic real-time systems analysis using discrete time markov chain. In *International Workshop on Mixed Criticality Systems (WMC)*, 2018.

[205] N. Stoimenov, L. Thiele, L. Santinelli, and G. Buttazzo. Resource adaptations with servers for hard real-time systems. TIK Report 292, Computer Engineering and Networks Laboratory, ETH Zurich, ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-320.pdf, September 2010.

[206] Nikolay Stoimenov, Lothar Thiele, Luca Santinelli, and Giorgio Buttazzo. Resource adaptations with servers for hard real-time systems. In *International Conference On Embedded Software (EMSOFT)*, 2010.

Mixed Criticality Modeling and Analysis Paradigms for Real-Time Embedded Systems

As research project, it has been chosen to investigate mixed criticality real-time embedded systems. The project aims at guaranteeing timing constraint and schedulability of applications with different requirements/criticality that are running together. Mixing criticality tries to reconcile efficient resource usage and safety assurance, thus it is critical with today's and future multi-core and many-core implementations for real-time embedded systems. It is a complex problem and has some interesting open problems that requires to be studied.
The project is presented with respect to works already made, and more importantly with perspectives that will be elaborated with future achievements. Previous research on non-mixed critical real-time embedded systems for timing analysis and schedulability analysis is also described; it is background work for mixed criticality achievements.

Keywords : MIXED-CRITICALITY;REAL-TIME EMBEDDED SYSTEMS

Modélisation et analyse de la criticité mixte pour le Systèmes embarqués temps réel

En tant que projet de recherche, il a été choisi d'étudier les systèmes embarqués en temps réel à criticité mixte. Le projet vise à garantir la contrainte de temps et l'ordonnancement des applications avec différentes exigences / criticité qui fonctionnent ensemble. La criticité mixte tente de concilier l'utilisation efficace des ressources et l'assurance de la sécurité, elle est donc essentielle avec les implémentations multicœurs et multicœurs actuelles et futures pour les systèmes embarqués en temps réel. Il s'agit d'un problème complexe qui présente des problèmes ouverts intéressants qui doivent être étudiés.
Le projet est présenté par rapport aux travaux déjà réalisés, et surtout avec des perspectives qui seront élaborées avec les réalisations futures. Des recherches antérieures sur les systèmes embarqués critiques en temps réel non mixtes pour l'analyse temporelle et l'analyse d'ordonnancement sont également décrites; c'est un travail de fond pour des réalisations de criticité mixte.

Mots-clés : CRITICITE MIXTE ; SYSTEME EMBARQUE TEMPS REEL