Hard Real-Time Computing Systems

Real-Time Systems Series

Series Editor

John A. Stankovic
University of Virginia, Virginia, USA

For further volumes:
http://www.springer.com/series/6941

Giorgio C. Buttazzo

Hard Real-Time Computing Systems

Predictable Scheduling Algorithms and Applications

Third Edition

Springer

Giorgio C. Buttazzo
RETIS Lab
Scuola Superiore Sant'Anna
Pisa
Italy
g.buttazzo@sssup.it

ISSN 1867-321X e-ISSN 1867-3228
ISBN 978-1-4614-0675-4 e-ISBN 978-1-4614-0676-1
DOI 10.1007/978-1-4614-0676-1
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2011937234

CONTENTS

PREFACE

Real-time computing plays a crucial role in our society since an increasing number of complex systems rely, in part or completely, on computer control. Examples of applications that require real-time computing include nuclear power plants, railway switching systems, automotive and avionic systems, air traffic control, telecommunications, robotics, and military systems. In the last several years, real-time computing has been required in new applications areas, such as medical equipments, consumer electronics, multimedia systems, flight simulation systems, virtual reality, and interactive games.

Despite this large application domain, most of the current real-time systems are still designed and implemented using low-level programming and empirical techniques, without the support of a scientific methodology. This approach results in a lack of reliability, which in critical applications may cause serious environmental damage or even loss of life.

This book is a basic treatise on real-time computing, with particular emphasis on predictable scheduling algorithms. The main objectives of the book are to introduce the basic concepts of real-time computing, illustrate the most significant results in the field, and provide the basic methodologies for designing predictable computing systems useful in supporting critical control applications.

This book is written for instructional use and is organized to enable readers without a strong knowledge of the subject matter to quickly grasp the material. Technical concepts are clearly defined at the beginning of each chapter, and algorithm descriptions are corroborated through concrete examples, illustrations, and tables.

Contents of the chapters

Chapter 1 presents a general introduction to real-time computing and real-time operating systems. It introduces the basic terminology and concepts used in the book, discusses the typical application domains, and clearly illustrates the main characteristics that distinguish real-time processing from other types of computing.

Chapter 2 introduces the general problem of scheduling a set of tasks on a uniprocessor system. Objectives, performance metrics, and hypotheses are clearly presented, and the scheduling problem is precisely formalized. The different algorithms proposed in the literature are then classified in a taxonomy, which provides a useful reference framework for understanding the different approaches. At the end of the chapter, a number of scheduling anomalies are illustrated to show that real-time computing is not equivalent to fast computing.

The rest of the book is dedicated to specific scheduling algorithms, which are presented as a function of the task characteristics.

Chapter 3 introduces a number of real-time scheduling algorithms for handling aperiodic tasks with explicit deadlines. Each algorithm is examined in regard to the task set assumptions, formal properties, performance, and implementation complexity.

Chapter 4 treats the problem of scheduling a set of real-time tasks with periodic activation requirements. In particular, three classical algorithms are presented in detail: Rate Monotonic, Earliest Deadline First, and Deadline Monotonic. A schedulability test is derived for each algorithm.

Chapter 5 deals with the problem of scheduling hybrid sets consisting of hard periodic and soft aperiodic tasks, in the context of fixed-priority assignments. Several algorithms proposed in the literature are analyzed in detail. Each algorithm is compared with respect to the assumptions made on the task set, its formal properties, its performance, and its implementation complexity.

Chapter 6 considers the same problem addressed in Chapter 5, but in the context of a dynamic priority assignment. Performance results and comparisons are presented at the end of the chapter.

Chapter 7 introduces the problem of scheduling a set of real-time tasks that may interact through shared resources and hence have both time and resource constraints. Three important resource access protocols are described in detail: the Priority Inheritance Protocol, the Priority Ceiling Protocol, and the Stack Resource Policy. These protocols are essential for achieving predictable behavior, since they bound the max-

imum blocking time of a process when accessing shared resources. The latter two protocols also prevent deadlocks and chained blocking.

Chapter 8 is dedicated to non-preemptive and limited preemptive scheduling, often used in industrial applications to make task execution more predictable and reduce the run time overhead introduced by arbitrary preemptions. Different solutions are presented, analyzed, and compared in terms of implementation complexity, predictability, and efficacy.

Chapter 9 deals with the problem of real-time scheduling during overload conditions; that is, those situations in which the total processor demand exceeds the available processing time. These conditions are critical for real-time systems, since not all tasks can complete within their timing constraints. This chapter introduces new metrics for evaluating the performance of a system and presents a new class of scheduling algorithms capable of achieving graceful degradation in overload conditions.

Chapter 10 describes some basic guidelines that should be considered during the design and the development of a hard real-time kernel for critical control applications. An example of a small real-time kernel is presented. The problem of time predictable inter-task communication is also discussed, and a particular communication mechanism for exchanging asynchronous messages among periodic tasks is illustrated. The final section shows how the runtime overhead of the kernel can be evaluated and taken into account in the guarantee tests.

Chapter 11 discusses some important issues related to the design of real-time applications. A robot control system is considered as a specific example for illustrating why control applications need real-time computing and how time constraints can be derived from the application requirements, even though they are not explicitly specified by the user. Finally, the basic set of kernel primitives presented in Chapter 9 is used to illustrate a concrete programming example of real-time tasks for sensory processing and control activities.

Chapter 12 concludes the book by presenting a number of real-time operating systems, including standard interfaces (like RT-Posix, APEX, OSEK, and Micro-ITRON), commercial operating systems (like VxWorks, QNX, OSE), and open source kernels (like Shark, Erika, Marte, and Linux real-time extensions).

Difference with the second edition

This book contains several changes and additions with respect to the previous edition. Several parts have been added to illustrate the most recent methods proposed in the real-time literature, mistakes and typos have been corrected, and some concepts have been further clarified, based on the observations received from the readers.

The most significant additions are the following:

- In Chapter 1, the introduction has been extended by presenting new applications domains. A description of the Ariane 5 accident has been added to explain the importance of analyzing the characteristic of the system and the environment. The list of desirable features for a real-time system has been revised and expanded. Additional notes on the cache behavior have been included in the section about predictability.
- Chapter 2 has been enhanced in several parts. The section on resource constraints has been extended by a concrete example that illustrates the importance of using semaphores to guarantee data consistency when accessing shared resources. Other examples of scheduling anomalies have been added at the end of the chapter to highlight critical situations that can occur when running an application at different speeds and when self-suspending a task using a delay primitive.
- In Chapter 4, the schedulability analysis of fixed priority tasks has been extended by introducing the workload-based test, which in several conditions is more efficient than the response time test.
- In Chapter 5, the analysis of fixed priority servers has been also extended under the Hyperbolic Bound and the Response Time Analysis and a procedure for dimensioning the server parameters has been included.
- Given the popularity that the CBS algorithm received in the real-time community, Chapter 6 has been extended by introducing a section on how to determine the CBS parameters for minimizing the average response time of the served tasks.
- Chapter 7 has been substantially revised. Two protocols, Non-Preemptive Protocol and Highest Locker Priority, have been described and analyzed, as they are often used in legacy applications to deal with shared resources. The second protocol, also known as Immediate Priority Ceiling, is specified in the OSEK standard for the development of automotive systems. Finally, a new section has been added at the end of the chapter to show how schedulability tests can be extended in the presence of blocking terms.

- A new chapter (8) on Limited Preemptive Scheduling has been added, describing a set of scheduling methods that can reduce the overhead introduced by preemptions. Limited preemptive techniques are very effective in practical applications and represent a solution to increase the predictability and the efficiency of real-time systems.
- Chapter 9 has been substantially restructured. The concepts of overload and overrun have been formally defined. An example has been added to explain the rejection strategy of the RED algorithm. The section on Resource Reservation has been expanded, discussing how to perform schedulability analysis, bandwidth adaptation, and resource sharing. Job skipping and elastic scheduling have also been revisited and expanded with examples, considerations, and implementation issues.
- Chapter 12 has been significantly revised and updated by adding the most recent developments achieved in the Linux community and in the research laboratories. The AUTOSAR specification has been included in the section on standards and a new section has been added on the development tools for the analysis and the simulation of real-time systems.
- New exercises has been added.
- The bibliography has been updated with more recent references.

Acknowledgments

This work is the result of 20 years of research and teaching activity in the field of real-time systems. The majority of the material presented in this book is based on class notes for an operating systems course taught at the University of Pisa, at the University of Pavia, and at the Scuola Superiore Sant'Anna of Pisa.

Though this book carries the name of a single author, it has been positively influenced by a number of people to whom I am indebted. Foremost, I would like to thank all my students, who have directly and indirectly contributed to improve its readability and clarity.

A personal note of appreciation goes to Paolo Ancilotti, who gave me the opportunity to teach these topics. Moreover, I would like to acknowledge the contributions of John Stankovic, Krithi Ramamritham, Herman Kopetz, John Lehoczky, Gerard Le Lann, Alan Burns, Gerhard Fohler, Sanjoy Baruah, and Lui Sha. Their inputs enhanced the overall quality of this work. I would also like to thank the Springer editorial staff for the support I received during the preparation of the manuscript.

Special appreciation goes to Marco Spuri and Fabrizio Sensini, who provided a substantial contribution to the development of dynamic scheduling algorithms for aperiodic service; Benedetto Allotta, who worked with me in approaching some problems related to control theory and robotics applications; Luca Abeni, for his contribution on resource reservation; Giuseppe Lipari and Marco Caccamo, for their work on resource sharing in dynamic scheduling; and Enrico Bini, for his novel approach on the analysis of fixed priority systems.

I also wish to thank Marko Bertogna, for his a valuable support in revising the chapter on limited preemptive scheduling, and Claudio Scordino for his contribution in the description of Linux related kernels.

A very special thanks goes to Paolo Gai, who developed the SHARK operating system and the ERIKA kernel (currently used as an educational kernels in several real-time courses around the world), and to all the people who are now involved in their maintenance (Tullio Facchinetti for SHARK, Mauro Marinoni and Gianluca Franchino for ERIKA).

Giorgio C. Buttazzo is Full Professor of Computer Engineering at the Scuola Superiore Sant'Anna of Pisa (Italy), where he teaches courses on Real-Time Systems and Computer Architectures. His main research interests include real-time operating systems, dynamic scheduling algorithms, quality of service control, multimedia systems, advanced robotics applications, and neural networks.

He graduated in Electrical Engineering at the University of Pisa in 1985, received a Master in Computer Science at the University of Pennsylvania in 1987, and a Ph.D. in Computer Engineering at the Scuola Superiore Sant'Anna of Pisa in 1991. During 1987, he worked on active perception and real-time control at the G.R.A.S.P. Laboratory of the University of Pennsylvania, in Philadelphia. From 1991 to 1998, he held a position of Assistant Professor at the Scuola Superiore Sant'Anna of Pisa, where he founded and coordinated the RETIS Laboratory, one of the world leading research groups on real-time systems. From 1998 to 2005, he held a position of Associate Professor at the University of Pavia, where he directed the He was a co-founder of Evidence s.r.l. (http://www.evidence.eu.com), a spin-off company of the Scuola Superiore Sant'Anna providing software solutions for real-time embedded systems.

Prof. Buttazzo has been Program Chair and General Chair of the major international conferences on real-time systems. He is Editor-in-Chief of the Journal of Real-Time Systems (Springer), the major journal on real-time computing, Associate Editor of the IEEE Transactions on Industrial Informatics, and Chair of the IEEE Technical Committee on Real-Time Systems.

He has authored 6 books on real-time systems and over 200 papers in the field of real-time systems, robotics, and neural networks.