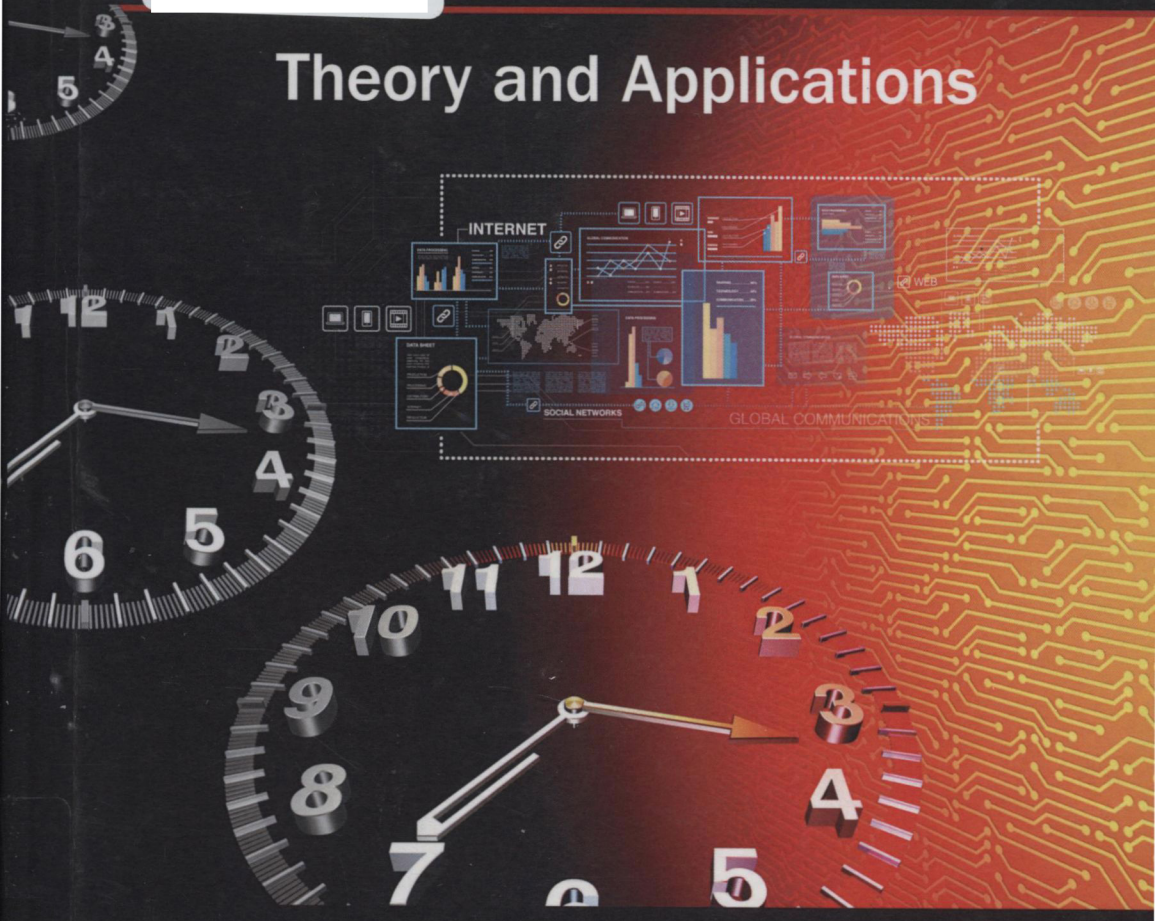
Real-Time and Distributed Real-Time Systems

Theory and Applications



Amitava Gupta
Anil Kumar Chandra
Peter Luksch

CRC Press
Taylor & Francis Group

Real-Time and Distributed Real-Time Systems

Theory and Applications

Amitava Gupta
Anil Kumar Ghandra
Peter Luksch

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **Informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20160108

International Standard Book Number-13: 978-1-4665-9847-8 (Hardback)

Library of Congress Cataloging-in-Publication Data

Names: Gupta, Amitava (Computer scientist), author. | Chandra, Anil Kumar, author. | Luksch, Peter, author.
Title: Real-time and distributed real-time systems : theory and applications / Amitava Gupta, Anil Kumar Chandra, and Peter Luksch.
Description: Boca Raton : Taylor & Francis, CRC Press, 2016. | Includes bibliographical references and index.
Identifiers: LCCN 2015043872 | ISBN 9781466598478 (alk. paper)
Subjects: LCSH: Electronic data processing--Distributed processing. | Real-time data processing.
Classification: LCC QA76.54 .G875 2016 | DDC 004/.33--dc23
LC record available at http://lccn.loc.gov/2015043872

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Real-Time and Distributed Real-Time Systems

Theory and Applications

In memory of Prof. D. Popovic, former professor of the University of Bremen, who has been an architect of an effective Indo–German collaboration in the realm of technology, a teacher for many Indian students in Germany, and a true friend of India.

Preface

The advent of digital computers during the late twentieth century has not only revolutionized the manner in which computations are carried out, but also the way in which computers can be used to control systems in real life and has given birth to a whole new paradigm termed as *real-time* (RT) systems. The inputs to such a system are usually from the real world and the system processes these inputs to generate outputs that affect the real world within a *finite* time irrespective of the computational load on the processing computer. Further, the massive advancements in the domain of communications have now made it possible to have RT systems performing an action in a coordinated manner over a communication interface, which has modified the RT paradigm further with the evolution of distributed RT systems or DRTS. As an example, one could consider a spacecraft headed toward the moon, the trajectory control for which involves the firing of thruster rockets with specific time deadlines. Missing a deadline in such a case might lead to the spacecraft being lost forever in deep space. Clearly, the requirements needed for the activation of thruster rockets on a spacecraft following a command from the mission control on Earth a hundred thousand kilometers away to meeting a deadline poses a many challenges on the computing resources and the communication channels to ensure timeliness. Necessity being the mother of invention, the requirements of the DRTS applications have paved the way for a whole gamut of techniques to represent, analyze, implement, and finally verify and validate such systems. This book introduces some of these aspects in six chapters that are found between its two covers.

The basis of a DRTS is an RT system. Thus, Chapter 1 begins with the basic concepts related to RT systems, namely, a *task* and its attributes, and the techniques to analyze the schedulability of a set of tasks for achieving a certain RT functionality; it then introduces the principles behind their synchronized functioning. The distributed processing attribute is then added to the RT functionality to cover a scenario where a set of RT systems delivers coordinated functionality through the scheduling of tasks and messages.

Chapter 2 extends the discussion of RT systems presented in Chapter 1 to DRTS by introducing the basic concepts starting with the topology of interconnection, architecture, and properties of a DRTS, and progressively introduces the concepts of time and time synchronization.

The second element in the constitution of a DRTS is an interconnecting network and the communication protocols that can be used to drive it. DRTS applications have led to the evolution of a large number of real-time communication protocols, and a few common protocols are presented and analyzed in detail in Chapter 3. Chapter 3 also presents advancements over the standard switched Ethernet like the multi-streamed TCP/IP, which

delivers the reliable communication required, for example, for safety critical DRTS applications. The chapter ends with protocol analysis using a protocol analyzer.

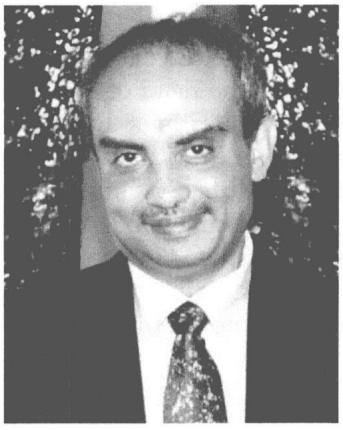
Designing DRTS applications using standard techniques is the theme of Chapter 4. The methodology presented in this chapter is based on a finite state machine (FSM) representation of an RT system. The basic concepts of automata theory are introduced first and then extended to model RT systems using standard FSM representation like the Mealy and Moore machines with practical systems like the coke vending machine. The basic concepts are extended to timed and hybrid automata and finally MATLAB® Stateflow is introduced as a tool for easily reproducing the FSMs as SIMULINK® models.

Chapter 5 illustrates how MATLAB can be used to develop real-time applications and integrate them over a communication network to develop a DRTS. The use of MATLAB allows development of DRTS applications in an easily reproducible manner. The methodology is illustrated using a representative application involving communication between the Earth, a lunar orbiter, and a lander to develop a DRTS. This simulates communication in a deep-space environment with delays taking into consideration the distance and radio-visibility of the communicating components. The chapter ends with an introduction to TrueTime, a MATLAB add-on, which can be used for simulation of protocols and hence DRTS applications in a MATLAB environment.

Finally, Chapter 6 presents the classification of DRTS applications in terms of the criticality and severity of their failures, the metrics in terms of safety and integrity levels, the different stages in their development life cycle, and their verification and validation techniques.

The individual chapters are supplemented by numerical and analytical problems or simulation exercises, which allow better understanding of the concepts. This supports a course format in which lectures are supplemented with tutorials and practical assignments. Such a format is quite common in German and Indian universities. A semester long course may also include a mini project using one of the simulation exercises given in Chapter 5.

Authors



Amitava Gupta is a professor in the Department of Power Engineering at Jadavpur University, Kolkata, India. A graduate in electrical engineering from Jadavpur University, Gupta earned his MTech from the Indian Institute of Technology, Kanpur and his PhD from Jadavpur University. He has served as an engineer in the Control and Instrumentation Division of the Nuclear Power Corporation of India Ltd. and as a member of the high-performance computing group of the Centre for Development of Advanced Computing, Bangalore before switching to academics. He also served as a *Gastwissenscaftler* (visiting scientist) at Technische Universität München, Germany, and as a DAAD *Gastdozent* (visiting professor) at the University of Rostock, Germany and Technische Universität München. His research interests include distributed simulation, distributed real-time systems, networked control systems, and control of nuclear reactors.

Anil Kumar Chandra graduated with a degree in electrical engineering from the Indian Institute of Technology, Delhi. He joined the Bhabha Atomic Research Center, Mumbai, and finally retired as a distinguished scientist and executive director (research and development) of the Nuclear Power Corporation of India Ltd., Mumbai. He worked on the modernization of control rooms and associated instrumentation for forthcoming Indian nuclear power plants. Chandra structured the required systems as distributed architectures with considerable standardization of hardware, software, and human–computer interaction. He also worked on obsolescence management of instrumentation in the older power plants. He contributed to devising the safety guide for review of digital instrumentation and control. He is a senior life member of the Computer Society of India and life member of the Indian Nuclear Society.

Peter Luksch is the chair of Distributed High Performance Computing at the Institute of Computing at the University of Rostock, Germany. Professor Luksch obtained his diploma in computer science from the Technische Universität München, Germany, followed by his doctoral degree and his habilitation. He served as the head of the Parallel and Distributed Applications Research Group of the Institut fuer Informaik of the Technische Universität München and as a visiting scholar at Emory University (Atlanta, Georgia) before joining the University of Rostock. Luksch's research primarily encompasses parallel and distributed computing.

Contents

1

Introduction to Real-Time Systems

Real-time (RT) systems belong to a class of computer systems that receive inputs from the external world, process them, and generate outputs so as to influence the external world, within a finite time. The inputs to such a system are usually *events* that occur in the real world, and the time interval between the instant at which such an event occurs and the instant at which the corresponding outputs are generated is termed the *response time* for the system. Thus, for an RT system, the response time is deterministic having a definite upper bound.

The deterministic response time of RT systems makes them suitable for automation-related applications. For example, if one considers a scenario where a thruster rocket is used to control the trajectory of a spacecraft used for planetary explorations, the necessity of deterministic response time becomes clear. The event in this case could be a command to fire the rocket based on the measurement data related to the trajectory of the spacecraft at any time instant (t), and the response time in this case would be the time between the instant at which the command is issued and the instant at which the rocket is fired. Understandably, this interval must have a definite upper bound or else the trajectory of the spacecraft would be completely different. It must, however, be borne in mind that the RT response of a system is associated with the deterministic nature of the response time and not with the magnitude of the response time. It is quite possible that a non-RT system produces a response within a much shorter time span compared to an RT system. For example, if one considers a MATLAB®/SIMULINK® [1] model that simulates the output of a dynamic system due to an input for T s, say, in the non-RT mode, the actual simulation run time may be T', $T' \ll T$ s, while the simulation time in the RT mode should be exactly T s in the ideal case and lie within a deterministic interval $[T - \Delta T, T + \Delta T]$ in the worst case.

As stated earlier, the inputs to an RT system are events that may be asynchronous with respect to the system, that is, the system may not be able to predict the instants at which these occur. In the worst case, the events may be completely unrelated without any *temporal cohesion* or mutual exclusion. Two events are said to possess a temporal cohesion if there exists a defined precedence law for their occurrence. Further, their processing may be completely different with the processing modules having no *functional cohesion* between them. Two processing modules are said to be functionally cohesive if they perform the same or a related set of functions. If one associates an entity

called a *task*, which is a sequence of functionally cohesive modules arising out of an event, then it is clear that such tasks, in the extreme case, can be completely independent of each other, without any precedence relationship between them. Thus, an RT system is inherently *multitasking* and RT systems form a special class of multitasking systems.

The rest of the chapter is organized as follows. Section 1.1 introduces the different types of RT systems. Sections 1.2 and 1.3 explain the key concepts associated with RT systems. Finally, Section 1.4 presents numerical and analytical problems.

1.1 Types of Real-Time (RT) Systems

RT systems may be broadly classified into two groups: *hard* RT systems and *soft* RT systems. The terminologies *hard* and *soft* are used to qualify the system based on the consequences of a nondeterministic response. A hard RT system is one in which the output loses its significance if it is not produced within a given response time. A typical example is a fly-by-wire control system controlling projectiles and aircrafts. On the other hand, for a soft RT system, a failure to meet response time requirements results in a degraded response and the *softness* of the system is a measure of its tardiness. A typical example is an RT streaming video display system. For such systems, the importance of the output generated by the system reduces with increased tardiness. The focus of this chapter is mostly hard RT systems, as soft RT systems can be viewed as a special case of hard RT systems with relaxed constraints on response time, and as such the concepts developed for hard RT systems can be easily extended to cover soft RT systems as well.

1.1.1 Embedded Systems

Embedded systems form a class of RT systems that are characterized by a short response time, compact architecture involving a processor with peripherals integrated in a single board often with small usable memory, enhanced robustness and exception handling capabilities, and in most cases low power consumption. To understand the significance of the term *embedded*, the difference between a normal (nonembedded) system architecture and an embedded one should be examined (see Figure 1.1).

As shown in Figure 1.1, in the strict classical sense of the term, an embedded system has the relevant operating system modules that an embedded application uses, embedded within the address space of the application itself. Hence the name embedded system. This arrangement substantially reduces the latency and the memory requirement. In most cases, an embedded