Towards the MaSHReC Manufacturing System under Real-Time Constraints

A Contribution to the Application of Real-Time System Advances to Production Control Systems

Dissertation

A thesis submitted to the
Department of Mathematics and Computer Science
of the
Paderborn University
in partial fulfillment of the requirements for the
degree of *Dr. rer. nat.*

by

Dania Adnan El-Kebbe

Paderborn
April 2002

Supervisors:

1. Prof. Dr. rer. nat. Franz-Joseph Rammig, University of Paderborn
2. Prof. Dr.-Ing. habil. Wilhelm Dangelmaier, University of Paderborn

◊

"The
time will come
when diligent re-
search over long periods
will bring to light things which
now lie hidden... And so this knowl-
edge will be unfolded only through long
successive ages. There will come
a time when our descendants
will be amazed that we did
not know things that
are so plain to
them."

◊

- Seneca, Natural Questions, Book 7, first century .

Acknowledgments

This work was supported by a scholarship from the graduate college "Parallel Computer Networks in the Production Technique", of Heinz-Nixdorf Institute (HNI), an interdisciplinary center of research and technology of the University of Paderborn, Germany.

Without the support of many people, this work would not have been possible. I am greatful to them all, not just for their technical support, but more importantly, for the experiences I shared with them.

First of all, I want to express my gratitude to Prof. Dr. Franz-Joseph Rammig and Prof. habil. Dr. Wilhelm Dangelmaier for supervising and vice-supervising this work respectively. From them, I learned, through many discussions, how to conduct research. I sincerely thank Prof. Rammig for the helpful advices and direction he has given me, for his support and constant guidance during this work.

Writing this thesis would not have been possible without the support of additional wonderful people. At the top of the list have to be my dear husband (Walid) and my lovely children (Ali and Roufayda). Walid always has a "go-getter" attitude about difficult tasks, which has influenced me. He urged to go when my patience ran short and confidence low. I have invested a lot of my free time writing this thesis. My family have been most understanding and encouraged me at times when they would have preferred we were doing other things together.

I am indebted to my colleagues for the nice environment we have in the work. Thanks to Christoph Bodba and Mohamed Hamadi for the many discussions we had. I am much obliged to Carsten Bke and Dr. Prasad for their detailed comments and for correcting a previous version of this thesis. I owe my sincere thanks to Dr. Ibrahim Kibbi from the American University of Beirut for the many extensive discussions we had during my stay in Beirut. Thank you also for Nathalie Naamai for correcting an earlier version of this thesis.

The closing but most important appreciations go to my parents. Without their continuous support and encouragement in my education, I would not have been able to achieve this endeavour.

Contents

Aknowledgments **V**

Contents **VII**

List of Figures **XI**

1 Introduction **1**

1.1 Motivation . . . 2
1.2 Manufacturing Systems . . . 2
1.3 Real-Time Systems . . . 3
1.4 Basic Definitions . . . 5
1.4.1 What is a Real-Time System? . . . 5
1.4.2 What is a Holonic Manufacturing System? . . . 6
1.5 Basic Approach . . . 7
1.6 Overview of the Thesis . . . 8
1.7 Organization and Special Features . . . 11
1.8 Relevant Publications . . . 11
Bibliography . . . 13

2 The MaSHReC Architecture **17**

2.1 State-of-the-Art . . . 18
2.2 The MaSHReC Architecture . . . 19
2.3 Motivations for the Use of the Holonic Approach . . . 20
2.4 Contributions of the Chapter . . . 22
2.5 Hypotheses Evaluation . . . 22
Bibliography . . . 25

3 The MaSHReC Design Methodology **27**

3.1 Requirements for Modeling Real-Time Systems . . . 28

3.1.1 Infrastructure modeling requirements 28
3.1.2 Behaviour modeling requirements 28
3.1.3 Structure modeling requirements 28
3.2 A Design Model for MaSHReC 29
3.2.1 Modeling the infrastructure of MaSHReC 29
3.2.2 Modeling the real-time behaviour of MaSHReC 30
3.2.3 Object interaction modeling 32
3.2.4 Object interaction modeling using collaboration diagrams 33
3.3 Contributions of the Chapter 40
3.4 Hypotheses Evaluation 40
Bibliography 41

4 A Local Scheduling Algorithm for MaSHReC 43
4.1 Taxonomy of Scheduling 44
4.2 Literature Survey 44
4.3 From RT Operating Systems to RT Manufacturing 45
4.4 System Characteristics 46
4.5 Computational Model for Preemptive Multitasking in Manufacturing Sytems 48
4.6 An Extended Schedulability Test for EDF 50
4.7 Contributions of the Chapter 52
4.8 Hypotheses Evaluation 52
Bibliography 55

5 A Distributed Scheduling Algorithm for MaSHReC 59
5.1 Traditional Scheduling Theory vs RT Scheduling Theory . . . 60
5.2 Related Work 60
5.3 The Structure and Behaviour of the Cell Coordinator Holon . 63
5.3.1 Bidder and local scheduler 64
5.3.2 The dispatcher 64
5.4 The Data Structure of the Cell Coordinator Holon 65
5.4.1 The system part table 65
5.4.2 Surplus 65
5.4.3 The guarantee routine 66
5.4.4 Considerations of time overheads in scheduling 66
5.5 Distributed Part Scheduling 67
5.5.1 The focused addressing scheme 67
5.5.2 The bidding scheme 68

5.5.3 Request for bids 68
5.5.4 Bidding in response to RFBs 70
5.5.5 Bid processing 72
5.5.6 Response to part award 73
5.6 Estimation Techniques 73
5.6.1 EST(Bid_wait) 74
5.6.2 EST(LS_wait) 74
5.6.3 EST(Response_time_for_RFBs) 74
5.6.4 EST(Production_time_local_between_ART_and_D) and EST(Production_time_bid_between_ART_and_D) . . . 75
5.6.5 EST(Part_processing_time_ratio) 77
5.7 Prototype Simulation 77
5.8 Performance Evaluation 78
5.9 Contributions of the Chapter 84
5.10 Hypotheses Evaluation 84
Bibliography 85

6 Predictable Monoprocessor Scheduling 93
6.1 Predicatble Aperiodic Scheduling 94
6.2 State-of-the-Art 95
6.3 Server Algorithms Evaluation 95
6.4 Assumptions and Terminology 100
6.5 The Total Bandwidth Server Algorithm 101
6.5.1 The fisrt version of the Total Bandwidth Server 101
6.5.2 Extension of the Total Bandwidth Server: the Robust Total Bandwidth algorithm 101
6.6 Adapting the TBS algorithm to the Monoprocessor Scheduling of a Production Stage 102
6.7 Adding Changeover Time Costs to TBS with Resource Reclaiming 102
6.8 Schedulability Analysis 103
6.9 Contributions of the Chapter 107
6.10 Hypotheses Evaluation 107
Bibliography 109

7 Predictable Multiprocessor Scheduling 121
7.1 Taxonomy of Multiprocessor Platforms 122
7.2 State-of-the-Art 122
7.3 The On-line Parallel Model 124

7.4 Total Bandwidth Server on Uniform Multiprocessors 127
7.5 Schedulability Analysis of Hybrid Task Systems on Uniform Multiprocessors 131
7.6 Practical Conclusions 135
7.7 Performance Evaluation 135
7.8 Contributions of the Chapter 139
7.9 Hypotheses Evaluation 139
Bibliography 141

8 Conclusions and Outlook 149
8.1 Summary 150
8.1.1 Chapter 1 "Introduction" 150
8.1.2 Chapter 2 "The MaSHReC Architecture" 150
8.1.3 Chapter 3 "The MaSHReC Design Model" 150
8.1.4 Chapter 4 "A Local Scheduling Algorithm for MaSHReC" 150
8.1.5 Chapter 5 "A Distributed Scheduling Algorithm for MaSHReC" 151
8.1.6 Chapter 6 "Predictable Monoprocessor Scheduling" . 151
8.1.7 Chapter 7 "Predictable Multiprocessor Scheduling" . . 151
8.2 Achievements of the Thesis 152
8.3 Suggestions for Future Research and Development 153

Bibliography 157

List of Figures

1.1 Workload considerations in a traditional production planning and control system 4
1.2 Workload considerations in a production planning and control system underlying real-time constraints 5
1.3 Structure of the thesis 10

2.1 The MaSHReC architecture 20

3.1 The class diagram of the MaSHReC architecture 31
3.2 The state machine diagram of the PPS Manager Holon 32
3.3 The state machine diagram of the Machine Holon 33
3.4 The state machine diagram of the Cell Coordinator Holon . . 34
3.5 The sequence diagram involved in initiating aperiodic orders . 35
3.6 The sequence diagram involved in initiating and producing pre-planned parts 36
3.7 The sequence diagram involved at the occurence of a machine trouble 37
3.8 The collaboration diagram involved in initiating and producing pre-planned parts 38
3.9 The collaboration diagram involved in initiating aperiodic orders 39
3.10 The collaboration diagram involved at the occurence of a machine trouble 39

4.1 Taxonomy of scheduling 45
4.2 The different perspectives of the scheduling problem in HMS and the different approaches to solve it 46
4.3 Example of an architecture for the on-line manufacturing system 47
4.4 Part characteritics specified by the programmer of the machine 49

5.1 Real-time distributed algorithms review 62
5.2 Illustration example of the estimation technique for local production time between ART and D 76
5.3 Prototype simulation 79
5.4 Changeover time versus production time 80
5.5 Performance evaluation of periodic tasks 81
5.6 Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 23,8% 81
5.7 Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 41,2,8% 82
5.8 Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 52,4% 82

6.1 Real-time uniprocessor algorithms evaluation 96
6.2 Real-time uniprocessor algorithms evaluation (continued) . . 97
6.3 Evaluation summary of dynamic priority server 99

7.1 Real-time multiprocessor algorithms evaluation 125
7.2 Changeover time versus production time 136
7.3 Performance evaluation of aperiodic tasks scheduled with TBS upon a uniform multiprocessor platform 136
7.4 Machine capacities variation of 3-machines platforms 137
7.5 Periodic utilization versus aperiodic server utilization upon 3-machines platforms 138

Chapter 1

Introduction

◊
"Almost all computer systems of the future will utilize real-time scientific principles and technology."
◊
John. A. Stankovic

A main object of this chapter is to present the basic hypotheses concluded from the different observations in several fields of Real-Time Systems and Production Planning and Control. The developed hypotheses structure the overview of the thesis. The evaluation of these hypotheses is the subject of the following seven chapters.

1.1 Motivation

From its title, it is obvious that this dissertation is about Real-Time Systems (RTS) and Production Planning and Control Systems (PPCS). Rather than pausing here to define both systems precisely, which will be done in section 1.4, the motivation of this thesis is first presented.

My involvement in the research fields RTS and PPCS - specifically the design of real-time systems, real-time modeling techniques, object-oriented design, real-time operating systems, real-time scheduling, schedulability analysis of real-time systems, production planning and control, and holonic manufacturing systems (HMS) - influenced this thesis. For this reason, the aspects of these fields, which are relevant for the motivation of the thesis, are considered in the following sections.

1.2 Manufacturing Systems

Since the industrial revolution, some 200 years ago, mathematicians, engineers, scientists and production managers have been trying to develop efficient factory scheduling and control procedures. In general, mathematicians and operation researchers have tried to solve the problem optimally, and have discovered that optimal analysis is very difficult. On the other hand, production managers and production engineers have tried to use heuristic procedures to order and organize production flow. Unfortunately, the results from these methodologies seem to be dependent on manufacturing system details and no general policies have been developed. The validation of these analyses has also been very difficult.

The advent of Numerical Control and Flexible Manufacturing Systems (FMS), including Holonic Manufacturing Systems (HMS), highlighted the past decade. Rapid static and hierarchical manufacturing systems have given way to systems that are more adaptable to rapid changes. Distributed control of various entities has taken place of centralized control. Thus, an effective coordination has been one key challenge for this generation of manufacturing systems. In a FMS, the ability to effectively schedule machines automatically has been highlighted. The system controller is responsible for scheduling decisions. However, selecting the proper scheduling rules in flexible automated systems is as difficult as it is in conventional manufacturing systems.

As more industries try to implement FMS, effective production control of these systems is needed to enable successful performance and safe operations.

Many control decisions in a manufacturing system are made in *real-time*, due to the dynamic nature of the environment. The manufacturing system should provide means to cope with various unexpected on-line requests in the presence of off-line preplanned requests.

To cope with various unexpected events in production planning and control, production engineers adopt a rescheduling policy (Nishi et al. 2000). Such rescheduling policies yields to the following drawbacks:

1. Rescheduling policies are feasible for small-sized and simple manufacturing systems. As manufacturing systems grow in size and complexity, a rescheduling policy becomes impracticable.
2. Additionally, no rescheduling policy is made on-line, in the sense that rescheduling policies are unfortunately executed at the end of a production shift.
3. Furthermore, no prediction can be made concerning unexpected arriving requests.

As an illustration, see Figure 1.1. It shows a manufacturing system consisting of four production stages (Broaching, Machining, Galvanic, and Assembly) designed according to the MFERT-model presented by Schneider (1996). The figure indicates a production stage to show that only preplanned tasks are schedulable using a production planning tool like OOPUS-DPS[1] upon a parallel uniform platform[2].

1.3 Real-Time Systems

As computer technologies advance, research activities and applications involving real-time systems have grown remarkably since the last few years. The field of real-time systems has expanded quickly towards new application areas, including telecommunication systems, multimedia computing, embedded systems, and wireless networks. Such new domains gave rise to new challenges and stimulated research in novel directions including quality of service control, energy aware computing, stochastic scheduling, and feedback based techniques for adaptive operating systems.

[1]OOPUS-DPS is an object-oriented planning tool developed by the workgroup of Prof. Dr. habil. W. Dangelmaier at the Heinz-Nixdorf Institute in Paderborn. For further information, please visit the web-page:
http://wwwhni.uni-paderborn.de/cim/projekte/oopus-dps.php3

[2]The parallel platform is only an example out of different platforms a production stage can comprise.

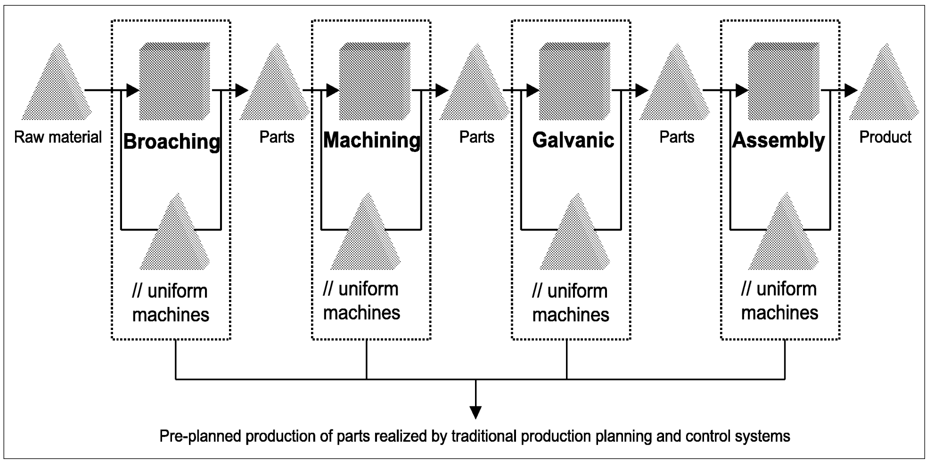


Figure 1.1: Workload considerations in a traditional production planning and control system

A novel application area of real-time systems, introduced in this thesis, is production control. The use of state-of-the-art real-time techniques in manufacturing planning and control is still rare. The lack of competence in real-time theory among production engineers, the lack of commercially available tools, and the lack of methodologies that consider real-time throughout the complete planning and control process are the major reasons for this.

The specific area of scheduling theory, which has been studied since more than two decades, involves different aspects of the real-time scheduling problem. Despite the leaping advances in real-time scheduling theory, more research effort is needed in order to put to use this theory in novel application areas.

Manufacturing systems underlying real-time constraints must be able to handle not only periodic tasks, but also aperiodic tasks. Periodic tasks are used to implement off-line pre-planned requests. While periodic tasks in real-time manufacturing systems have hard deadlines, aperiodic tasks may have soft, firm, hard or no deadlines at all. As an illustration, the Figure 1.2 presents the same production line as in Figure 1.1 but from a real-time perspective. Pre-planned orders are substituted by periodic orders. Furthermore, newly arriving aperiodic orders in a production shift are allowed to be produced.

The reader should note that a full implementation of a Manufacturing

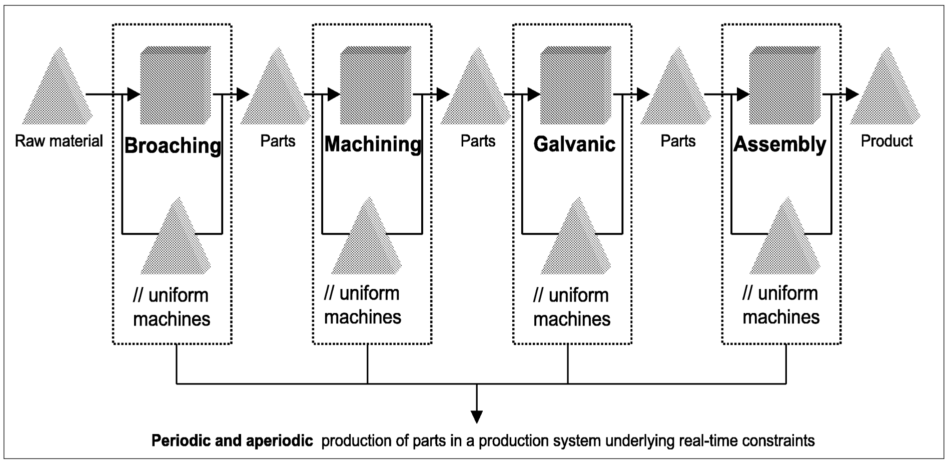


Figure 1.2: Workload considerations in a production planning and control system underlying real-time constraints

System under Real-Time Constraints (MaSHReC) would require years of research effort. In this thesis, a restricted structure and design of a Manufacturing System under Real-Time Constraints is clearly defined. A scheduling methodology for MaSHReC, especially the aperiodic scheduling of hard and firm tasks in the presence of hard periodic tasks, is provided. Schedulability tests and simulation studies of a prototyped manufacturing system are generated to ensure the predictability of the system and to evaluate the performance of the algorithms respectively.

1.4 Basic Definitions

Before proceeding further, it is helpful to establish the meaning of the terms "Real-Time System" and "Holonic Manufacturing System".

1.4.1 What is a Real-Time System?

In order to show the various ways in which real-time systems are defined, the following definitions of real-time systems are stated.

Young (1982) defines a real-time system to be:

"any information processing activity or system which has to respond to externally generated stimuli within a finite and specified period."

The Predictably Dependable Computer Systems Project (PDCS) project (Randell et al. 1995) gives the following definition:

"A real-time system is a system that is required to react to stimuli from the environment (including the passage of physical time) within time intervals dictated by the environment."

Kopetz (1997) referred to a real-time system as

"a computer system in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instant (deadline) *at which these results are produced."*

Stankovic and Ramamritham (1988a) used a quite similar definition to refer to *hard real-time systems.*

The definition of a real-time system, as stated by (Kopetz 1997), is used in this thesis. A real-time system is a system where the correct functioning of the system depends on the results produced by the system and the time at which these results are produced. Hard real-time sytems are characterized by the fact that severe and sometimes catastrophic consequences will result if logical as well as timing correctness properties of the system are not satisfied.

1.4.2 What is a Holonic Manufacturing System?

Holonic Manufacturing is a new paradigm originated in the framework of the Intelligent Manufacturing Systems (IMS) program (Kriz 1995), one of the largest research programs launched in manufacturing. Holonic systems are distributed multi-agent systems in which the entities, called holons, are both autonomous and cooperative. The Holonic Manufacturing System (HMS) is defined by the HMS consortium as follows (Valckenaers et al. 1997):

HMS is a holarchy that integrates the entire range of manufacturing activities from order booking through design, production and marketing to realize the agile manufacturing enterprise.

The Holonic Manufacturing System distinguishes itself from other distributed approaches by introducing hierarchy in the system. Compared to traditional hierarchical systems, this hierarchy should be flexible, dynamic and reconfigurable (Bongaerts 1998).

Central to the concept of HMS is the notion of a holon - a term introduced by the hungarian A. Koestler (Koestler 1967)- to mean simultaneously a whole and a part of a whole. The word "Holon" is a combination of the greek "holos" which means a whole, and the suffix "on", which as in proton or neutron suggests a particle or a part. Thus a holon can be made up of other holons. A holon is autonomous and cooperative. Each production unit can be a holon and these holons co-operate with each other - from planning and scheduling to physical production - to manufacture products.

1.5 Basic Approach

Due to the several observations provided in sections 1.2 and 1.3, different hypotheses may be identified. A procedure of hypotheses evaluation occurs at the end of each chapter.

Hypothesis 1 *Due to their autonomous and cooperative characteristics Holonic Manufacturing Systems provide a suitable basis to model the dynamic environment in which manufacturing has to take place. A real-time model supporting holonic features would benefit not only from holonic attributes, but also from predictable, logically and timely correct results.*

Hypothesis 2 *Deadline guarantee is the most important constraint in the manufacturing system. Thus, in opposite to soft real-time (RT) constraints which permit deadlines to be missed, the manufacturing activities should take place under hard real-time constraints.*

Hypothesis 3 *On-line requests in a production system under hard real-time constraints are aperiodic tasks to handle external events, which are usually unpredictable. Allowing additional orders in the manufacturing environment may lead to a high-level system utilization.*

Hypothesis 4 *The use of rigorous schedulability analysis coupled with predictable scheduling algorithms that attempt to capture the timing behavior of all tasks in a* ***multiprocessor*** *real-time system environment provides a high degree of schedulable resource utilization and* ***a priori*** *verification of timing correctness for all tasks in the system.*

Hypothesis 5 *The support of basic properties, including the following ones, is a major challenge in Production Planning and Control Systems underlying real-time constraints.*

1. ***Timeliness.*** *Results have to be correct not only in their value but also in the time domain.*

2. ***Overload support.*** *Real-time PPCS must not collapse when they are subject to peak-load conditions.*

3. ***Predictability.*** *One of the most important properties that differentiates real-time systems from other conventional systems. The system must be able to predict the consequences of any scheduling decisison. If some task cannot be guaranteed within its timing constraints, the system must notify this fact in advance, so that alternative actions can be planned in time to cope with the event.*

4. ***Maintainability.*** *In order to ensure that possible system modifications are easy to perform, the architecture of a real-time manufacturing system should be designed according to an object-oriented structure.*

1.6 Overview of the Thesis

In this section, the chapters of this thesis are put in a common perspective.

Chapter 2 *The MaSHReC Architecture* introduces the architecture of a Manufacturing System under Hard Real-Time Constraints. The goal from developing an architecture for MaSHReC is to ease the design and implementation of production control systems. The architecture described uses the holonic approach. Therefore, this chapter reviews state-of-the-art manufacturing architectures based on the holonic approach and it clears the motivation for the use of the holonic paradigm.

The modeling of MaSHReC is covered in **Chapter 3**, *The MaSHReC Design Model.* An object oriented (OO) model is developped based on the Unified Modeling Language (UML) modeling techniques. The different diagrams of the UML enables a graph based representation of MaSHReC. The OO model provides an overview of the information and functional structure of the manufacturing system. It allows also the behavioural representation of MaSHReC.

The local scheduling of a manufacturing cell in MaSHReC is the focus of **Chapter 4**, *A Local Scheduling Algorithm for MaSHReC.* The chapter 4 starts by providing a taxonomy which allows to classify the scheduling algorithms in manufacturing systems under consideration of real-time aspects. It reviews briefly the scheduling literature of Holonic Manufacturing systems. A computational model used for scheduling is built and the extended Earliest Deadline First algorithm and its schedulability test are developed for the local scheduling of MaSHReC.

The main purpose of **Chapter 5**, *A Distributed Scheduling Alogorithm for MaSHReC* is to develop a distributed scheduling algorithm that can be applied to the proposed model. The chapter starts with a brief survey of the variety of distributed scheduling algorithms. The result is an effective distributed scheduling algorithm, that satisfies the real-time requirements of MaSHReC. Various simulation experiments of the presented algorithm upon a prototyped manufactuting system underlying real-time constraints allow to study the performance of the system.

The goal of **Chapter 6**, *Predictable Monoprocessor Scheduling*, is to improve the results presented in the precedent chapter, especially those dealing with the unpredictability of heuristic mechanisms. To address the problem of *predictable* aperiodic scheduling in the presence of hard periodic tasks is an important challenge for MaSHReC. This is achieved upon the monoprocessor platform of a production stage by applying real-time server algorithms. A survey of real-time uniprocessor scheduling mechanisms including server algorithms is addressed. An evaluation of dynamic server algorithms assists to find the appropriate scheduling technique to be applied to MaSHReC. Furthermore, an extension of the algorithm to include changeover time costs is provided. The predictability of the system is proven through schedulability analysis techniques.

One step further than dealing with predictable real-time scheduling upon monoprocessor production stages in manufacturing systems is to predictably schedule aperiodic tasks upon multiprocessor production stages. This is the topic of **Chapter 7**, *Predictable Multiprocessor Scheduling.* First, a comparative study of state-of-the-art real-time multiprocessor scheduling schemes is provided. The Total Bandwidth Server algorithm is developed. The uniform multiprocessor scheduling algorithm is analyzed by considering its performance when it is allowed to run on faster machines. The predictability of the system is proven through schedulability analysis techniques. The performance of the multiprocessor control algorithm is analyzed through simulation studies.

In **Chapter 8**, *Conclusions and Outlook*, the thesis concludes with a summary of the results obtained and suggests promising directions for future research.

The Figure 1.3 illustrates the structure of the thesis including the affiliation of the different hypotheses.

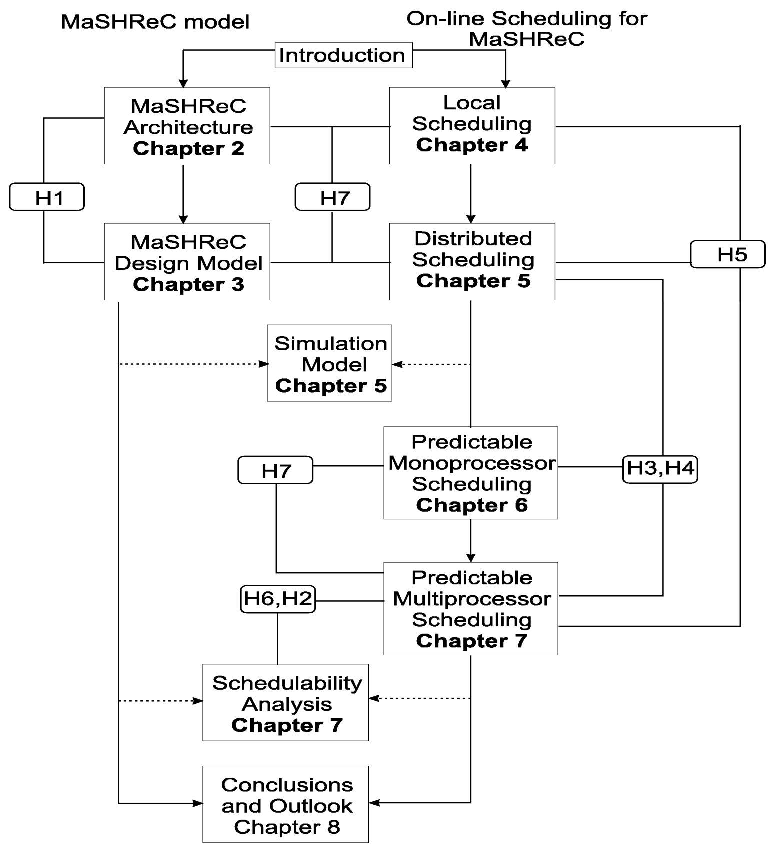

MaSHReC model
On-line Scheduling for MaSHReC
Introduction
MaSHReC Architecture Chapter 2
Local Scheduling Chapter 4
H1
H7
MaSHReC Design Model Chapter 3
Distributed Scheduling Chapter 5
H5
Simulation Model Chapter 5
H7
Predictable Monoprocessor Scheduling Chapter 6
H3,H4
H6,H2
Predictable Multiprocessor Scheduling Chapter 7
Schedulability Analysis Chapter 7
Conclusions and Outlook Chapter 8

Figure 1.3: Structure of the thesis

1.7 Organization and Special Features

- The thesis addresses topics in the fields of real-time systems, Unified Modeling Language, object-oriented design, production planning and control systems, and Flexible Manufacturing Systems (FMS) including Holonic Manufacturing Systems (HMS). It is assumed that the reader is familiar with the basic terms and concepts in these areas.
- A bibliography with short comments is presented at the end of each chapter. The complete bibliography is presented at the end of the thesis.
- The obtained scientific results listed in the last section of each chapter (Chapter 2 to 7) and in Section 8.2 are derived from the point of view of the author.
- Unless stated explicitly, the following words are used interchangeably: machine and processor; job, order and task; manufacturing and production.
- In section "Hypotheses Evaluation" of chapters 2 to 7, the evaluation of the hypotheses related to each chapter is to be deduced from the results presented in the chapter.

1.8 Relevant Publications

The most important results of this work were reported in the following publications: (El-Kebbe 2001c), (El-Kebbe 2001b), (El-Kebbe 2001d), (El-Kebbe 2001e), (El-Kebbe 2001a), (El-Kebbe 2000c), (El-Kebbe 2000b), (El-Kebbe 2000a), and (El-Kebbe 2002).

Bibliography

Bongaerts, L. (1998). *Integration of Scheduling and Control in Holonic Manufacturing Systems.* Ph. D. thesis, PMA/K.U. Leuven.
Develops shop floor control algorithms based on the notion of holonic manufacturing systems.

El-Kebbe, D. A. (2000a, May). Integration of On-Line and Off-Line Systems in Real-Time Manufacturing. In *Proc. of the Workshop of the Informatics Graduate Colleges*, Schloss Dagstuhl, Germany.
Introduces a methodology to integrate off-line and on-line production planning systems to achieve both flexibility and a guarantee for critical production tasks.

El-Kebbe, D. A. (2000b, October). Modeling the Manufacturing System under Hard Real-Time Constraints Using Real-Time UML. In *Workshop on Formal Design Techniques Using Real-Time UML*, York, UK.
Discusses modeling techniques for MaSHReC. Introduces an object oriented holonic model for a manufacturing system under real-time constraints using UML.

El-Kebbe, D. A. (2000c, September). Towards a Manufacturing System under Hard Real-Time Constraints. In *Informatik 2000: 30. Jahrestagung der Gesellschaft für Informatik*, Berlin.
Presents the basic concept of a manufacturing system under hard real-time constraints.

El-Kebbe, D. A. (2001a, April). A Real-Time Holon-Based Architecture for the Production Planning System: Further Results. In *Proc. of the Workshop on Agent Based Simulation II*, Passau, Germany.
Presents the architecture for the manufacturing system under hard real-time constraints (MaSHReC). Models the structure, bahavior and interactions of MaSHReC using UML.

El-Kebbe, D. A. (2001b, March). A UML Model for the MaSHReC Architecture. In *Proc. of the International Congress on Information Science Innovations in Intelligent Automated Manufacturing*, Dubai, United Arab Emirates.

Investigates current holonic manufacturing architectures. The MaSHReC model is designed using the Unified Modeling Language (UML). UML templates are provided to allow the design of the system structure, components, relations, data, functions and interactions.

El-Kebbe, D. A. (2001c, December). Aperiodic Scheduling in a Dynamic Real-Time Manufacturing System. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Addresses the problem of aperiodic scheduling of manufacturing systems under hard real-time constraints. Adopts the extended version of the Total Bandwidth Server algorithm and extends it to include changeover time costs.

El-Kebbe, D. A. (2001d, October). Findings from Adapting Real-Time Aperiodic Tasks to Production Planning Systems. In *Proc. of the IEEE Conference on Emerging Technologies and Factory Automation*, Antibes, France.

Presents some findings from adapting real-time operating systems scheduling theory to production planning and control. Concludes with the introduction of server mechanisms to the aperiodic scheduling of manufacturing systems with the novel feature of including changeover time costs.

El-Kebbe, D. A. (2001e, October). Scheduling of Manufacturing Systems under Hard Real-Time Constraints. In *Proc. of the IEEE Systems, Man and Cybernetics Conference*, Tucson, Arizona.

Presents a real-time distributed scheduling scheme for a manufacturing system underlying real-time constraints based on bidding and focused addressing.

El-Kebbe, D. A. (2002, April). Predictable Multiprocessor Scheduling in Manufacturing Systems underlying hard Real-Time Constraints. In *Proc. of the AIPS Workshop on On-line Planning and Scheduling*, Toulouse, France.

Presents the Total Bandwidth server algorithm and its schedulability analysis upon uniform multiprocessor platforms.

Koestler, A. (1967). *The Ghost in the Machine.* London: Hutchinson & Co. (Second Edition: Arkana Books, London, 1989)

Observes a dichotomy of wholeness and partness in living organisms and social organisations. Uses the "Janus Effect" as a metaphor for this dichotomy of wholeness and partness. Suggests a new term "holon" to describe the members of these systems.

Kopetz, H. (1997). *Real-Time Systems: Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers.

This book treats issues of hard real-time distributed systems with fault tolerance aspects.

Kriz, D. (1995). *Holonic Manufacturing Systems: Case study of an IMS Consortium.* http://hms.ifw.uni-hannover.de.

Presents motivational and implementation issues for Holonic Manufacturing Systems.

Nishi, T., A. Sakata, S. Hasebe, and I. Hashimoto (2000). Autonomous Decentralized Scheduling System for Just-in-Time Production. In *Proc. of the 7th International Symposium on Process System Engineering*, pp. 345–351.

Proposes an autonomous decentralized scheduling system for just-in-time production. The goal for each sub-system scheduling includes the storage costs for intermediate and final products in addition to the changeover costs and the due date penalties.

Randell, B., J.-C. Laprie, H. Kopetz, and B. Littlewood (1995). *Predictably Dependable Computing Systems.* Springer-Verlag.

Contains a selection of papers on main topics in Predictable Dependable Computing Systems: fault prevention, fault tolerance, fault removal, and fault forecasting.

Schneider, U. (1996). *Ein formales Modell und eine Klassifikation für die Fertigungssteuerung.* Ph. D. thesis, Heinz-Nixdorf Institut / Universität Paderborn.

Presents a methodology for the construction of production planning and control systems based on a classification of production control tasks and production control procedures.

Stankovic, J. and K. Ramamritham (1988). *Tutorial Hard Real-Time Systems.* IEEE Computer Society Press.

Valckenaers, P., H. V. Brussel, L. Bongaerts, and J. Wyns (1997). Holonic Manufacturing Sytems. *Integrated Computer-aided Engineering 4*(3), 191–201.

Young, S. (1982). *Real-Time Languages: Design and Development.* Ellis Horwood.

Chapter 2

The MaSHReC Architecture

Chapter 2 introduces the architecture of a Manufacturing System under Real-Time Constraints. The architecture described uses the holonic approach. Therefore, this chapter reviews state-of-the-art manufacturing architectures based on the holonic approach and it develops a section clearing the motivation for the use of the holonic paradigm.

2.1 State-of-the-Art

The HMS paradigm has been launched since less than a decade. Thus, few architectures were investigated in the literature. This paragraph gives a brief overview of these architectures.

Kouiss et al. (1997) presented a multi-agent system for dynamic scheduling in Flexible Manufacturing Systems. Each agent models a *Work Centre* in the FMS, and keeps its own job queue. Jobs are scheduled by locally applying dispatch rules. The selection of a dispatch rule takes into account primary and secondary objectives as well as the state of the work centre, and information received from other agents. This selection is done in a rule based way. The system includes a supervisory agent that monitors the global state of the manufacturing system. It may eventually impose the use of specific dispatch rules to some or all work centre agents if it finds it advisable to do.

AARIA's architecture (Parunak et al. 1997) connects a group of manufacturing capabilities in the form of agents. The system implements the following functionality: finite capacity scheduling, basic planning, order entry, purchasing, bill-of-materials management, inventory management, resource management, personnel management, integrated financials, and reporting. Each agent has pieces of this general functionality and when placed together, they self-configure to provide a full Enterprise Resource Planning and Manufacturing Execution System (ERP/MES). There is no centralised system in AARIA architecture.

Gou and Luh (1997) follow a very modular approach to implement a Holonic Manufacturing System, using eight types of holons: *Product, Part, Machine-Type, Machine, Cell-Coordinator, Cell, Factory Coordinator* and *Factory*. A Product holon represents a manufacturing order, and is composed by Part holons, which represents a stage in the manufacturing process of that product. A Cell holon is a Holarchy composed of a Cell-Coordinator, Machine, Machine-Type and Part holons. A factory consists of multiple cells each with several machines. Products must move through machines (most likely on different cells). Infinite buffer capacity is assumed.

The PROSA reference architecture (Valckenaers et al. 1998) is built around three types of basic holons. The Order holons represent a task in the manufacturing system, the Product holons which hold the product and process knowledge and the Resource holons, which contain a physical part (the production resource) and an information part that controls the resource. The basic holons are assisted by staff holons, which give them advices.

The architectures presented above do not provide means to support real-time constraints. Therefore, they cannot be used to model the novel approach of manufacturing systems underlying real-time constraints. Furthermore, the aimed architecture should support analysis techniques of real-time systems in order to predict and to assure timely correct behaviour.

2.2 The MaSHReC Architecture

There are two basic building blocks in the MaSHReC architecture characterized upon their workload: the off-line and the on-line manufacturing sytem.

- The off-line manufacturing system is made up of all orders preplanned by a Production Planning System (PPS) and a PPS manager holon, responsible for the off-line scheduling of orders and their communication between the off-line and the on-line system.
- The on-line manufacturing system which is made up of manufacturing cells coordinating with each other, in order to execute preplanned and aperiodic orders. Each manufacturing cell consists of a machine with an input and an output buffer. A cell coordinator holon supervising each cell is responsible for the local scheduling of the manufacturing cell as well as the inter-cell coordination scheduling. The MaSHReC architecture is shown in Fig. 2.1.

The manufacturing system consists of:

$$M = \{M_i \mid i \in \mathbb{N}\}$$

a set of workstations or machines,

$$IB = \{IB_i \mid i \in \mathbb{N}\}$$

an input buffer per machine for the parts before execution on the machine,

$$OB = \{OB_i \mid i \in \mathbb{N}\}$$

an output buffer per machine for the parts after execution on the machine,

$$T = \{T_j \mid j \in \mathbb{N}\}$$

a set of tools related to each station,

$$M = \{P_m \mid m \in \mathbb{N}\}$$

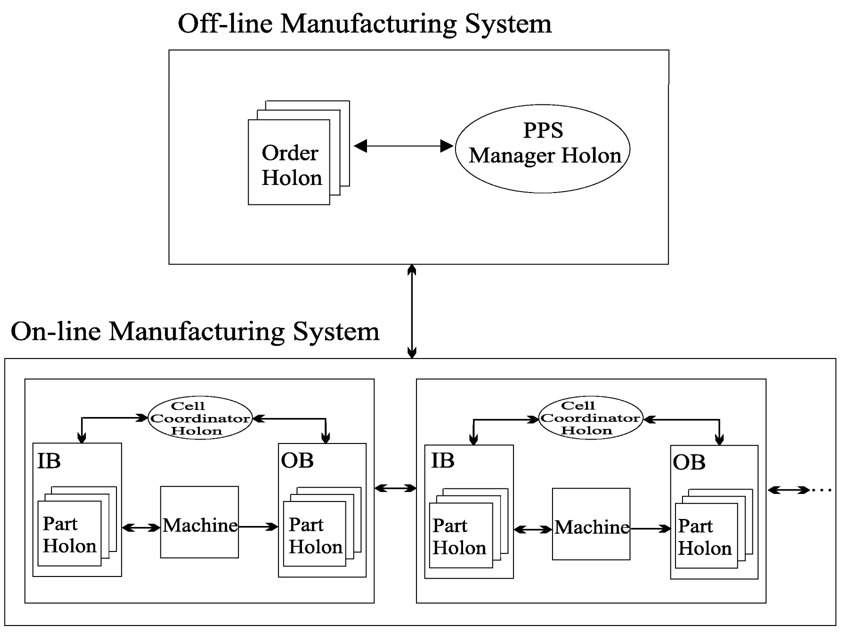


Figure 2.1: The MaSHReC architecture

a set of parts, and

$$C = \{C_n \mid n \in \mathbb{N}\}$$

a set of carriers of tools or parts.

In this thesis, tools and parts are considered to be available as soon as the parts are to be produced on the machine.

2.3 Motivations for the Use of the Holonic Approach

MaSHReC is a holonic architecture since it is characteized by the following propoerties.

- Hierarchical structure: A priori, all manufacturing cells have equal status. However, it turns out to be of advantage if a Cell Coordinator Holon is designed to manage the manufacturing cell activities. At the next hierarchical level, the PPS Manager holon gives guidelines to these regional cell coordinators.

- Heterarchical structure: The internal behaviour of the manufacturing cell could be changed without influencing the overall behaviour of the system.
- Decomposability: The overall task to execute an order before its deadline can be decomposed into part orders (to avoid deadline missing).
- Communication: Manufacturing cells communicate with each other (in general, cells of the same type). The cell manager holon can communicate with all holons in the manufacturing system.
- Social elements: The setting is cooperative within the cells of the manufacturing system.
- Real-time requirement: This scenario underlies hard or firm real-time requirements, where parts have to cope with a changing environment and a fixed deadline.

2.4 Contributions of the Chapter

The scientific contributions of this chapter are summarized as follows:

- *Review of Holonic Manufacturing Architectures*
  Results of a complete review of holonic manufacturing architectures are provided. An architecture supporting real-time considerations was not found in the literature survey.

- *Development of a Global Architecture of a Manufacturing System under Real-Time Constraints*
  A global architecture of a manufacturing System under real-time constraints is developed. The main purpose of this architecture is to facilitate the design and implementation of production control systems.

2.5 Hypotheses Evaluation

The achievements of this chapter, together with Chapter 1 (partly), are basically related to Hypothesis 1, which deals with holonic manufacturing models supporting real-time constraints.

Hypothesis 1 *Due to their autonomous and cooperative characteristics Holonic Manufacturing Systems provide a suitable basis to model the dynamic environment in which manufacturing has to take place. A real-time model supporting holonic features would benefit not only from holonic attributes, but also from predictable, logically and timely correct results.*

Hypothesis 5 *The support of basic properties, including the following ones, is a major challenge in Production Planning and Control System underlying real-time constraints.*

...

4. ***Maintainability****. In order to ensure that possible system modifications are easy to perform, the architecture of a real-time manufacturing system should be designed according to an object-oriented structure.*

The hierarchical approach to production planning is classically used to handle the different descriptions of the production process in complex production systems. The advances in manufacturing systems (see Section 1.2) and the motivation for implementing holonic features (see Section 2.3) reveals the relevance of HMS in dynamic environments. Being the core of a design model for MaSHReC, an architecture designed to support real-time systems and to incorporate holonic attributes is clearly provided. In addition, the aspects which characterize the MaSHReC holonic architecture are evidently defined.

Bibliography

Gou, L. and P. Luh (1997, June). Holonic Manufacturing Scheduling: Architecture, Cooperation, Mechanism and Implementation. In *Proc. of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Tokyo, Japan.

Models a Holonic Manufacturing System with its key elements such as machines, cells, factories, parts, products, operators, teams, etc. having autonomous and cooperative properties.

Kouiss, K., H. Pierreval, and N. Mebarki (1997). Using multi-agent architecture in FMS for dynamic scheduling. *J. of Intelligent Manufacturing 8*, 41–47.

Presents a multi-agent system for dynamic scheduling in Flexible Manufacturing Systems.

Parunak, H., A. Baker, and S. Clark (1997, February). The AARIA Agent Architecture. In *Proc. of the International Conference on Autonomous Agents*, Marina del Rey, CA.

Presents an architecture for a full Enterprise Resource Planning anf Manufacturing Execution System.

Valckenaers, P., J. Wyns, H. V. Brussel, L. Bongaerts, and P. Peeters (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry, Special Issue on Intelligent Manufacturing Systems 37*(3), 255–276.

Presents the PROSA reference architecture for holonic Manufacturing Systems. This architecture consists of three types of holons: product, resource and order with the assistance of staff holons.

Chapter 3

The MaSHReC Design Methodology

The goal of this chapter is to develop a design methodology of a manufacturing system under real-time constraints based on the architecture presented in Chapter 2. The Unified Modeling Language (UML) is used to model the system structure, components, relations, data, functions and interactions.

3.1 Requirements for Modeling Real-Time Systems

The need of timeliness is essential in real-time systems. In "hard" real-time systems the violation of even a single deadline might mean catastrophic results. A crucial requirement therefore is to use models of such systems to derive early and accurate predictions of the actual system. A number of techniques have evolved in the real-time domain for predicting these attributes. These techniques include schedulability analysis procedures, which determine whether a given system will meet all of its deadlines.
For models to be predictive, it is necessary to model not only the structure and behaviour of the system but also the logical and physical resources, or engineering infrastructure on which that system relies (Selic 1999).

3.1.1 Infrastructure modeling requirements

The engineering infrastructure comprises physical devices such as machines and networks as well as logical devices such as concurrent orders or buffers. When attributes of these resources such as capacity and location are assigned numerical values in the model, then the model can be used to make predictions of the actual system. In addition to resources, the engineering infrastructure often include basic services such as scheduling, timing, buffer management, and so forth. These services may also have quantitative attributes such as response time.

3.1.2 Behaviour modeling requirements

The sequential paradigm of specifying behaviour that characterizes procedural programming is often inappropriate for describing the behaviour of real-time systems. This is mainly because real-time events can occur unpredictably and concurrently with other events. Instead two different methods have evolved for specifying the behaviour in real-time systems (Kopetz 1997). The event-driven method is used for modeling discrete behaviours in which concurrent events occur asynchronously. In contrast, the time-driven style is used for continuous or periodic inputs.

3.1.3 Structure modeling requirements

Modeling runtime structures of real-time systems is one of the dominant aspects in the modeling of real-time systems. A key aspect of runtime structures is the relationship between the individual objects. These can be grouped into three categories:

- Peer relationships that occur between directly communicating objects.
- Containment relationships. This includes both composition and aggregation as known in the UML terminology, whereby the container merely encapsulates a component.
- Layering relationships. This is a special case of a client-server relationship in which the client uses a shared server as part of its implementation.

In addition to the preceding three different structural forms, there is a need to model dynamic runtime structures. These are structures in which objects and links are constantly being created and destroyed as the load in the system changes.

3.2 A Design Model for MaSHReC

UML is a general-purpose modeling language. It is a standard for visualizing object-oriented diagrams. UML emerged from the combination of the Booch method (Booch 1991) and the method developed by Rumbaugh et al. (1991). Since the concepts of HMS are closely related to the concepts of object-oriented design, an object-oriented modeling tool is choosen.

Research effort has been conducted to extend UML for the real-time domain by adding new concepts to its base UML-RT (Selic 1998). A work towards achieving this aim has been conducted by the C-Lab in Paderborn in the course of the DESS project[1] (Software Development Process for Embedded Real-Time Systems). It is based on the concept of consistency for sequence diagrams and statecharts, in general and then directed to the domain of real-time modeling. This is of fundamental importance for modeling reliable real-time systems. Further readings can be found in: (Küster and Stroop 2000), (Küster and Stroop 2001), and (Küster 2001).

3.2.1 Modeling the infrastructure of MaSHReC

The MaSHReC infrastructure as defined in section 3.1.1 is modeled with the UML class diagram. The UML class diagram addresses the static design

[1]DESS is sponsored by BMBF and is also a european project as part of the Eureka program ITEA.

view of the system. Figure 3.1 shows a set of classes from the implementation of the MaSHReC infrastructure.

3.2.2 Modeling the real-time behaviour of MaSHReC

MaSHReC is an event-driven (reactive) system, where the production process is launched by the arrival of orders. The production of parts in MaSHReC can be easily interrupted if another more critical concurrent part arrives. It is mostly convenient to describe the underlying model with a transition system formalism. State machines of UML are an example of a transition system that is particularly suitable for highly concurrent real-time systems. Several state machine diagrams of holons in MaSHReC are described in this section.

Figure 3.2 represents the state machine diagram of the PPS Manager Holon. The PPS Manager Holon might be in any of three states: `Idle` (waiting for an aperiodic request or a confirmation order message from the on-line system), `Executing Order Plan` (designating different parts to be produced for a specific order and allocating them to the appropriate cell type) , and `Creating` (creating Part Order Holons corresponding to an aperiodic order).

Figure 3.3 represents the state machine diagram of a Machine Holon. The Machine Holon might be in any of the following states: `Idle` (waiting for a Part Order Holon), `Producing` (producing a part; a part being produced may be preempted due to the arrival of a high priority Part Order Holon), `Blocked` (In case of a machine trouble, the machine passes to a `Blocked` state).

The Cell Coordinator Holon passes from the `Idle` state to the `EDF Scheduling` state at the arrival of a Part Order Holon event and when the cell state is InFunction. Otherwise, it sends the Part Order Holon or Part Order Message to another cell from the same type. On entering the `EDF Scheduling` state, the Cell Coordinator Holon calculates a local EDF schedule. At scheduling end and if the schedule is feasible, the Cell Coordinator Holon returns to the idle state.

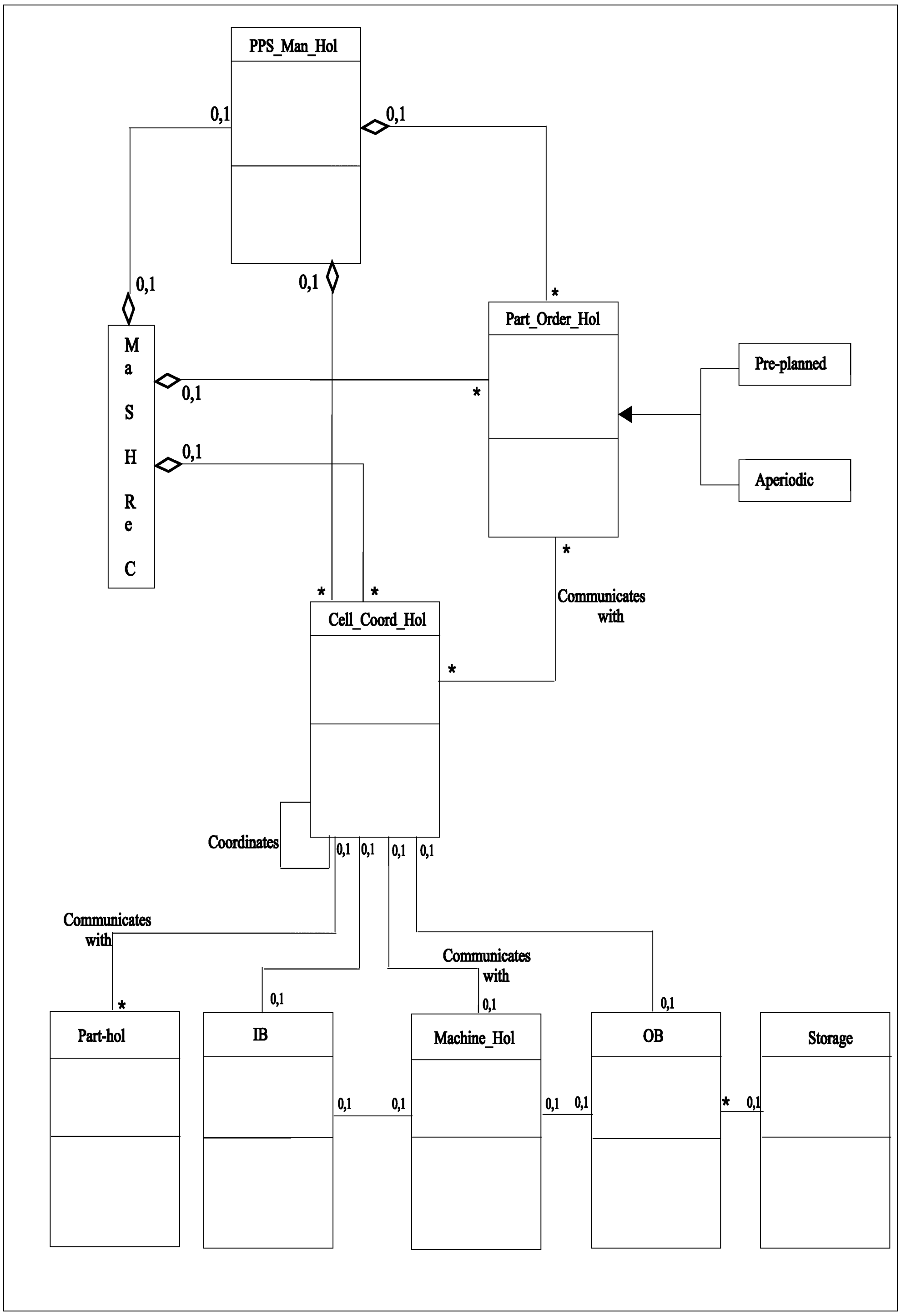


Figure 3.1: The class diagram of the MaSHReC architecture

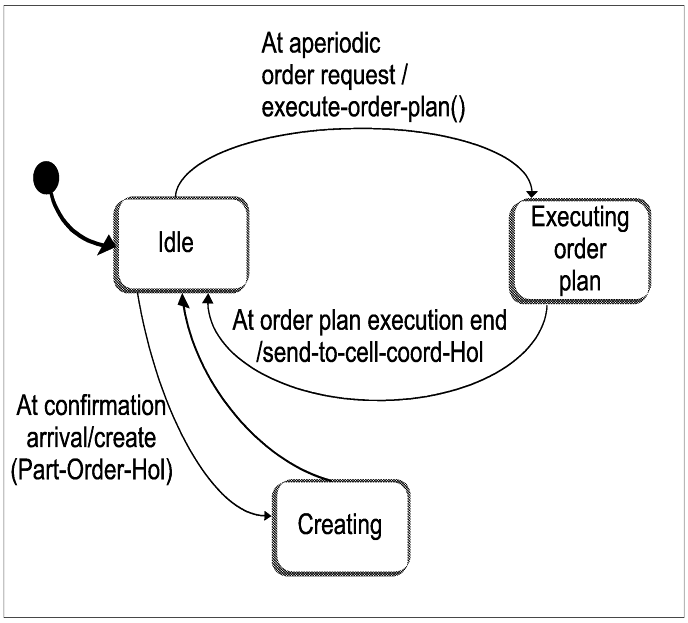

At aperiodic order request / execute-order-plan()
Idle
Executing order plan
At order plan execution end /send-to-cell-coord-Hol
At confirmation arrival/create (Part-Order-Hol)
Creating

Figure 3.2: The state machine diagram of the PPS Manager Holon

3.2.3 Object interaction modeling

While state machine diagrams are useful to describe the reactive behaviour of individual objects, it is also useful to specify and view the behaviour of a set of collaborating objects. From a real-time perspective, sequence diagrams, which emphasize the time ordering of a message, are a useful way to model the interaction among objects. Two features distinguish them from collaboration diagrams (Booch et al. 1999): the object lifeline (represented by vertical dashed lines in the sequence diagram and the focus of control (represented by a tall, thin rectangle in the sequence diagram).

Figure 3.5 shows a sequence diagram that specifies the flow of control involved in initiating aperiodic orders. The sequence begins at the arrival of an aperiodic order by calling the `PPS_Man_Hol`'s `Get_order()` operation. The `PPS_Man_Hol` executes an `Execute_product_plan()` operation (to specify the parts to be produced and their related machine type), which in turn sends the aperiodic order to the `Cell_Coord_Hol` of the specified machine type. The `Cell_Coord_Hol` calls the `EDF_schedule()` op-

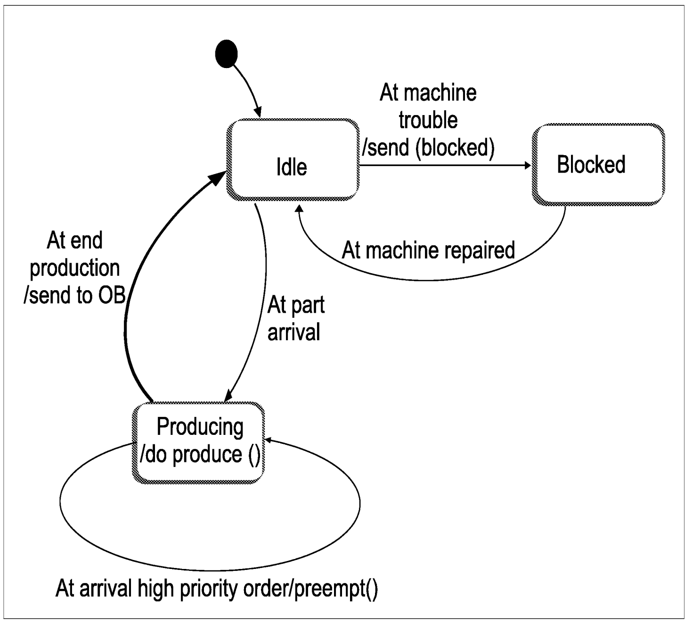

At machine trouble /send (blocked)
Idle
Blocked
At machine repaired
At end production /send to OB
At part arrival
Producing /do produce ()
At arrival high priority order/preempt()

Figure 3.3: The state machine diagram of the Machine Holon

eration, that calculates a local EDF schedule. When the schedule is feasible, the `Cell_Coord_Hol` runs the `send(schedule)` operation to the `PPS_Man_Hol`, which in turn creates the corresponding `Part_Order_Hol`. In the other case, it executes an `Inter_cell_schedule` operation among all the cells from the same type. If the inter-cell schedule is feasible, the `Cell_Coord_Hol` runs a `send(schedule)` operation to the `Cell_Coord_Hol`, which in turn also creates the corresponding `Part_Order_Hol`. If not, a rejection message is sent to `PPS_Man_Hol` and the order is deleted.

The sequence diagram specifying the flow of control involved in initiating and producing Preplanned Parts is represented in Figure 3.6.

The sequence diagram involved at the occurence of a machine trouble is represented in Figure 3.7.

3.2.4 Object interaction modeling using collaboration diagrams

Some recent work did focus on the use of collaboration diagrams for specifying complex real-time architectures. This has the following advantages:

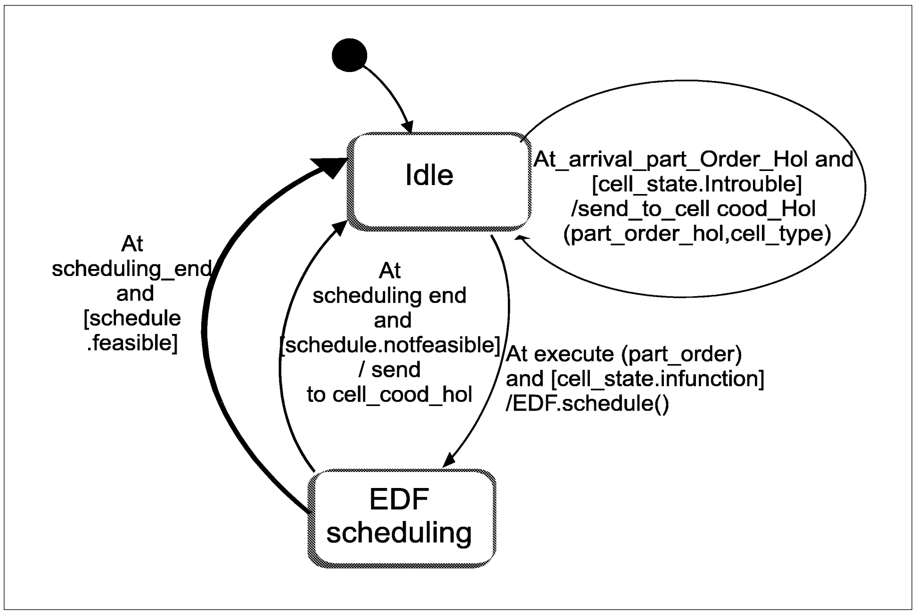

Idle
At_arrival_part_Order_Hol and [cell_state.Introuble] /send_to_cell cood_Hol (part_order_hol,cell_type)
At scheduling_end and [schedule .feasible]
At scheduling end and [schedule.notfeasible] / send to cell_cood_hol
At execute (part_order) and [cell_state.infunction] /EDF.schedule()
EDF scheduling

Figure 3.4: The state machine diagram of the Cell Coordinator Holon

- The architectural models can be formally analyzed for consistency and completeness.
- The models are executable and allow early and precise assessment of the validity of different architectural approaches.
- The implementation is derived directly from architectural specifications using automatic code generation.

Figures 3.8, 3.9, and 3.10 show the same scenario respectively as in figures 3.6, 3.5, 3.7 but in the form of a collaboration diagram. The structure in Figures 3.8, 3.9, 3.10 is clearer, but that sequence is harder to follow.

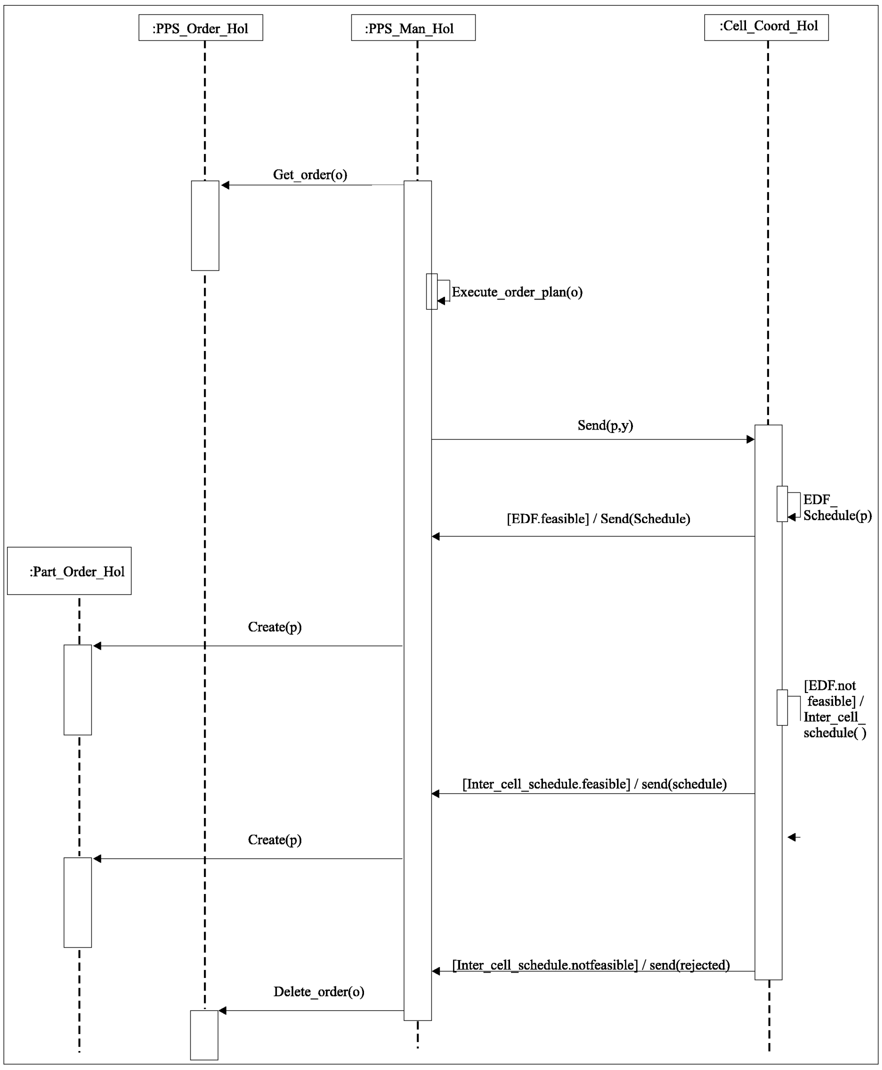

:PPS_Order_Hol
:PPS_Man_Hol
:Cell_Coord_Hol
Get_order(o)
Execute_order_plan(o)
Send(p,y)
EDF_ Schedule(p)
[EDF.feasible] / Send(Schedule)
:Part_Order_Hol
Create(p)
[EDF.not feasible] / Inter_cell_ schedule( )
[Inter_cell_schedule.feasible] / send(schedule)
Create(p)
[Inter_cell_schedule.notfeasible] / send(rejected)
Delete_order(o)

Figure 3.5: The sequence diagram involved in initiating aperiodic orders

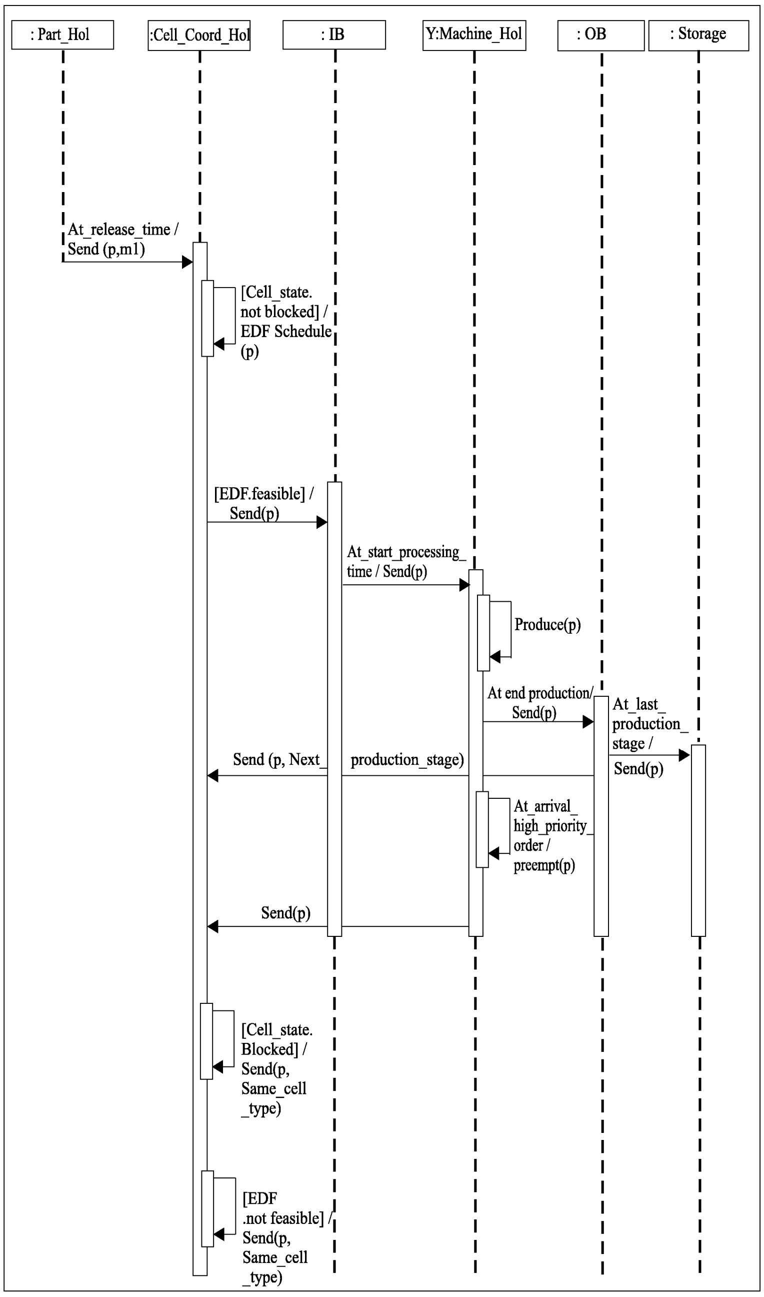

: Part_Hol
:Cell_Coord_Hol
: IB
Y:Machine_Hol
: OB
: Storage
At_release_time / Send (p,m1)
[Cell_state. not blocked] / EDF Schedule (p)
[EDF.feasible] / Send(p)
At_start_processing_ time / Send(p)
Produce(p)
At end production/ Send(p)
At_last_ production_ stage / Send(p)
Send (p, Next_ production_stage)
At_arrival_ high_priority_ order / preempt(p)
Send(p)
[Cell_state. Blocked] / Send(p, Same_cell _type)
[EDF .not feasible] / Send(p, Same_cell _type)

Figure 3.6: The sequence diagram involved in initiating and producing preplanned parts

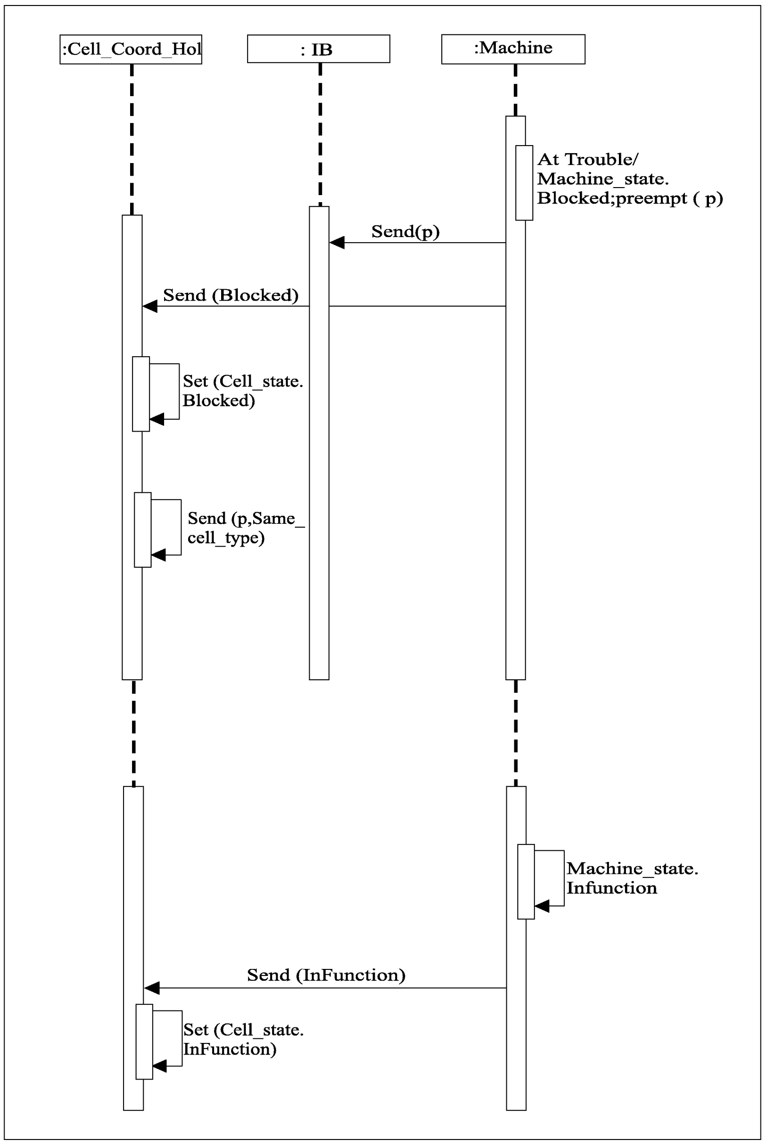


Figure 3.7: The sequence diagram involved at the occurence of a machine trouble

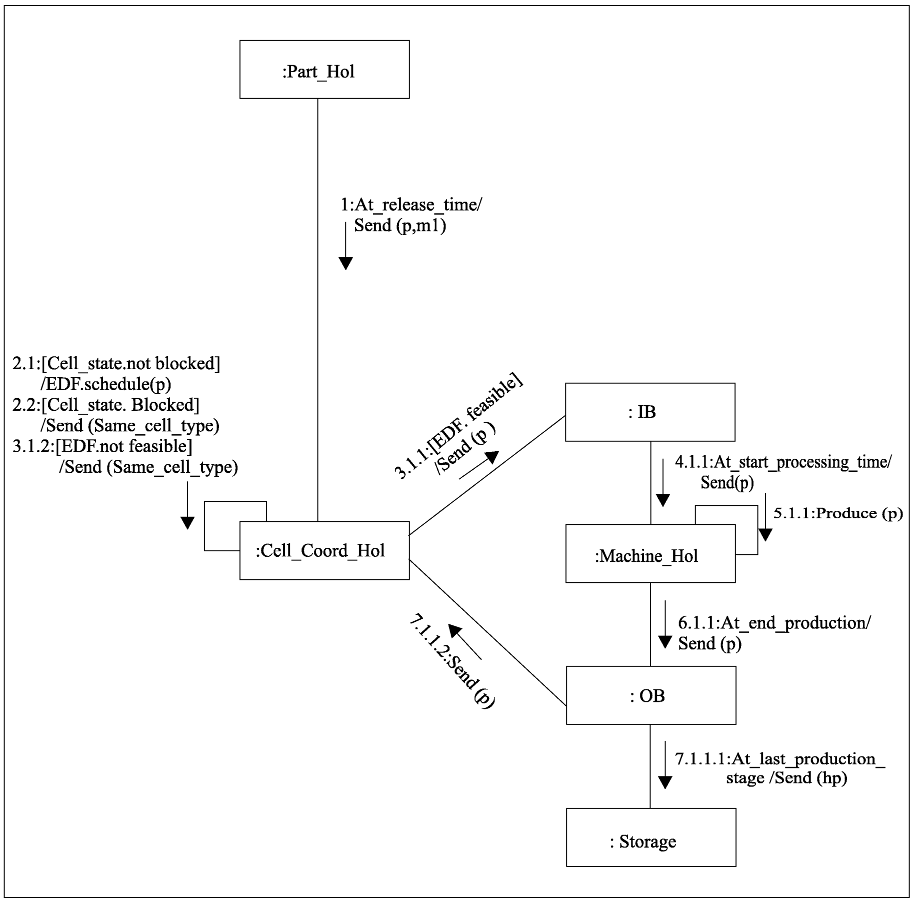

:Part_Hol
1:At_release_time/ Send (p,m1)
2.1:[Cell_state.not blocked] /EDF.schedule(p)
2.2:[Cell_state. Blocked] /Send (Same_cell_type)
3.1.2:[EDF.not feasible] /Send (Same_cell_type)
:Cell_Coord_Hol
3.1.1:[EDF. feasible] /Send (p)
: IB
4.1.1:At_start_processing_time/ Send(p)
5.1.1:Produce (p)
:Machine_Hol
6.1.1:At_end_production/ Send (p)
7.1.1.2:Send (p)
: OB
7.1.1.1:At_last_production_ stage /Send (hp)
: Storage

Figure 3.8: The collaboration diagram involved in initiating and producing pre-planned parts

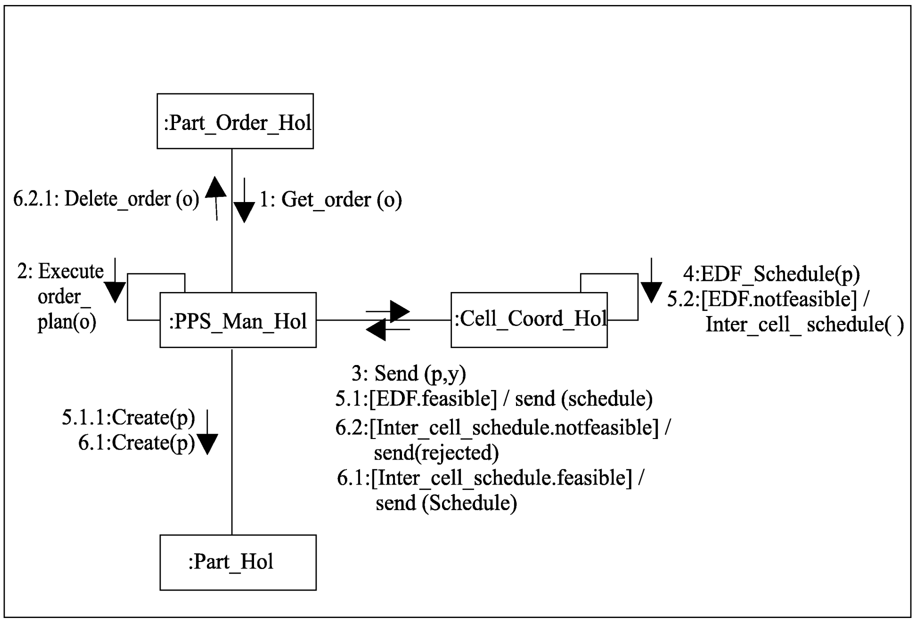

:Part_Order_Hol
6.2.1: Delete_order (o)
1: Get_order (o)
2: Execute order_ plan(o)
:PPS_Man_Hol
:Cell_Coord_Hol
4:EDF_Schedule(p)
5.2:[EDF.notfeasible] / Inter_cell_ schedule( )
3: Send (p,y)
5.1:[EDF.feasible] / send (schedule)
6.2:[Inter_cell_schedule.notfeasible] / send(rejected)
6.1:[Inter_cell_schedule.feasible] / send (Schedule)
5.1.1:Create(p)
6.1:Create(p)
:Part_Hol

Figure 3.9: The collaboration diagram involved in initiating aperiodic orders

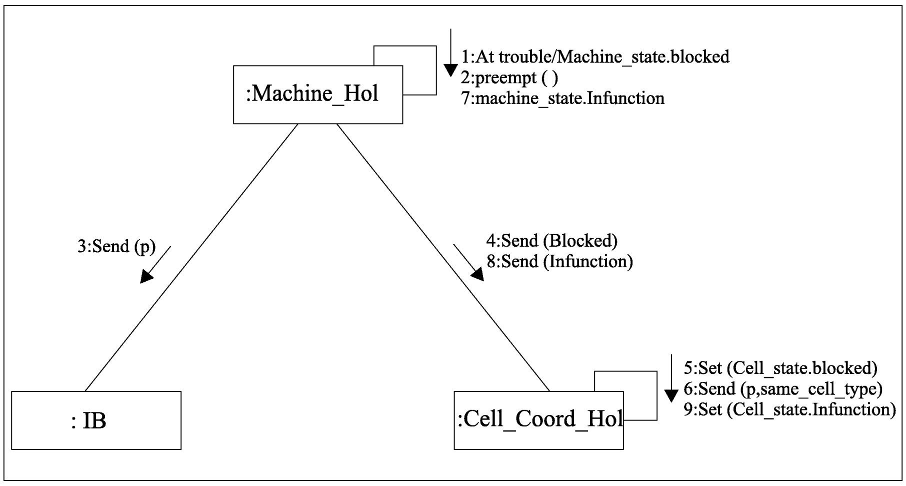

:Machine_Hol
1:At trouble/Machine_state.blocked
2:preempt ( )
7:machine_state.Infunction
3:Send (p)
4:Send (Blocked)
8:Send (Infunction)
: IB
:Cell_Coord_Hol
5:Set (Cell_state.blocked)
6:Send (p,same_cell_type)
9:Set (Cell_state.Infunction)

Figure 3.10: The collaboration diagram involved at the occurence of a machine trouble

3.3 Contributions of the Chapter

The scientific contribution of this chapter can be summarized as follows:

- *A Design Methodology of a Manufacturing System under Real-Time Constraints*
  A design methodology of a manufacturing system under real-time constraints using the holonic approach is provided. UML templates, which allow the design of the system structure, components, relations, data, functions and interactions, are provided. Based on these templates modeling of any real manufacturing system may be carried out.

3.4 Hypotheses Evaluation

This chapter is fully dedicated to the modeling of MaSHReC and is the basis for the validation of the following hypotheses:

Hypothesis 1 *Due to their autonomous and cooperative characteristics Holonic Manufacturing Systems provide a suitable basis to model the dynamic environment in which manufacturing has to take place. A real-time model supporting holonic features would benefit not only from holonic attributes, but also from predictable, logically and timely correct results.*

Hypothesis 5 *The support of basic properties, including the following ones, is a major challenge in Production Planning and Control System underlying real-time constraints.*

...

4. ***Maintainability****. In order to ensure that possible system modifications are easy to perform, the architecture of a real-time manufacturing system should be designed according to an object-oriented structure.*

A structured object oriented model like MaSHReC allows the maintability of the system according to the application demands.

Bibliography

Booch, G. (1991). *Object Oriented Design with Applications.* Redwood City, California: Benjamin/Cummings.

Presents a unified notation that incorporates the Boochś notation (first book) and other methods. Includes several examples of projects implemented in C++.

Booch, G., J. Rumbaugh, and I. Jacobson (1999). *The Unified Modeling Language User Guide.* Addison-Wesley.

Presents the UML conceptual model and applies UML to series of modeling problems.

Kopetz, H. (1997). *Real-Time Systems: Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers.

This book treats issues of hard real-time distributed systems with fault tolerance aspects.

Küster, J. (2001, September). Towards Behavior Consistent Modeling in UML-RT. In *Proc. of the Forum on Design Languages (FDL'01).*

Küster, J. and J. Stroop (2001). Consistent Design of Embedded Real-Time Systems with UML-RT. In *Proc. of the 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, pp. 31–40. IEEE Computer Society.

Küster, J. M. and J. Stroop (2000, October). Towards Consistency of Dynamic Models and Analysis of Timing Constraints. In *Workshop on Formal Design Techniques Using Real-Time UML*, York, UK.

Introduces the notion of consistency for sequence diagrams and state-charts to the domain of real-time modeling using UML/UML-RT.

Rumbaugh, J., M. Blaha, W. Premerlani, S. Eddy, and W. Lorensen (1991). *Object Oriented Modeling and Design.* New York, USA: Prentice Hall, Englewood Cliffs.

Explores the Object Modeling Technique (OMT), a generic method of representing objects and their relationships.

Selic, B. (1998). Using UML for modelling complex real-time systems. *Lecture Notes in Computer Science 1474*, 250–260.

Selic, B. (1999, October). Turning Clockwise: Using UML in the Real-Time Domain. *Communications of the ACM 42*(10), 46–54.
Discusses modeling of real-time systems using UML.

Chapter 4

A Local Scheduling Algorithm for MaSHReC

The purpose of this chapter is to develop a local scheduling algorithm for MaSHReC. Before treating the local scheduling problem, Chapter 4 provides a taxonomy which allows to classify the scheduling algorithms in manufacturing systems under consideration of real-time aspects. Following the taxonomy, a brief overview of the holonic scheduling literature is given. The chapter ends by building the computational model and assigning the local scheduling scheme and its schedulability test.

4.1 Taxonomy of Scheduling

The scheduling problem has many different dimensions. Therefore it is very hard to define a taxonomy that pays tribute to all aspects. However, it is important to attempt such a classification as it allows to define a specific scheduling algorithm and relate it to the other research conducted in scheduling. El-Rewini and Ali (1995) and Dussa-Zieger (1998) propose a taxonomy for the scheduling problem. This section presents a taxonomy that reflects the taxonomy of El-Rewini and Ali (1995) and Dussa-Zieger (1998), but additionally includes new features.

The idea of the taxonomy in Figure 4.1 is derived from the scheduling problem itself. To be more precise, the taxonomy reflects the input to the scheduling algorithm and the nature of the algorithm itself. A scheduling algorithm expects some type of workload, e.g. application programs, and it schedules these programs on some given target machine. Therefore the first classification criteria is the type of target machine which the scheduling algorithm can handle, while the type of workload yields the second criteria. The third aspect that allows to classify scheduling algorithms is the algorithm itself, e.g. whether the produced schedule is optimal or not. The fourth dimension of the classification is the nature of jobs, e.g. whether a job can be preempted or not. Figure 4.1 depicts the taxonomy. It should be noted that the taxonomy is by no means a complete description of all scheduling issues.

4.2 Literature Survey

Scheduling in HMS was investigated by several researchers. Ramos and Sousa (1996) address heterarchical scheduling. They propose a distributed scheduling method based on the contract net protocol and on forward and backward propagation of precedence constraints. Tönshoff and Winkler (1995) propose a completely heterarchical method for distributed resource allocation, where agents perform simulations in virtual worlds. Márkus et al. (1996), Márkus and Váncza (1996), and Váncza and Márkus (1998) propose a market mechanism for task allocation in HMS. Moriwaki et al. (1992) propose an on-line scheduling mechanism in an object-oriented framework. In order to include opportunities for optimization, Gou et al. (1994) define a distributed algorithm based upon Lagrangian relaxation concepts. Also Kádar et al. (1997) discover the need for centralized elements and propose the coupling of a multi-agent control system with a genetic algorithm.

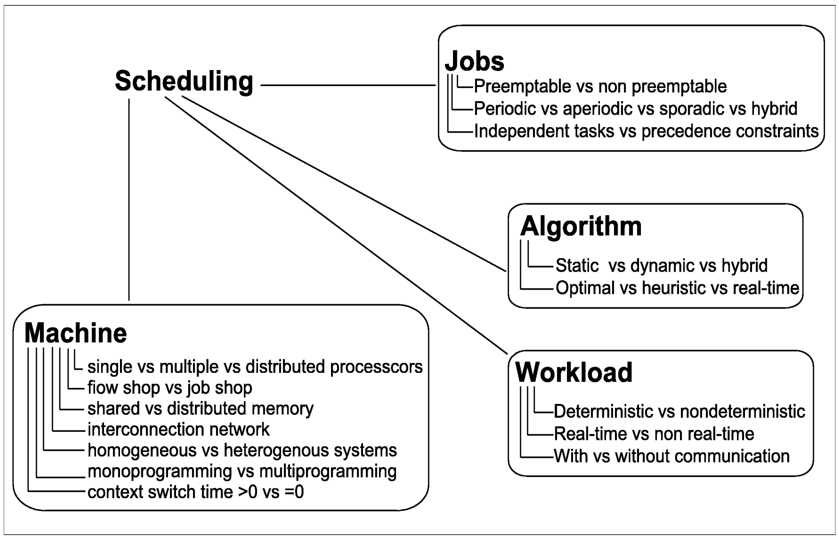

Scheduling
Jobs
Preemptable vs non preemptable
Periodic vs aperiodic vs sporadic vs hybrid
Independent tasks vs precedence constraints
Algorithm
Static vs dynamic vs hybrid
Optimal vs heuristic vs real-time
Machine
single vs multiple vs distributed processcors
flow shop vs job shop
shared vs distributed memory
interconnection network
homogeneous vs heterogenous systems
monoprogramming vs multiprogramming
context switch time >0 vs =0
Workload
Deterministic vs nondeterministic
Real-time vs non real-time
With vs without communication

Figure 4.1: Taxonomy of scheduling

Bongaerts (1998) addresses the issue of integrating hierarchy in distributed systems. He proposes totally distributed algorithms based on a market mechanism and supplemented with a reactive scheduler. While the reactive scheduler is calculating an updated schedule, the autonomous agents execute the existing schedule.

This brief overview on the literature of scheduling in HMS shows that scheduling was restricted to deterministic and stochastic perspectives. The scheduling approaches presented focused on solving and optimizing the deterministic and stochastic scheduling problem. These approaches are illustrated in Fig. 4.2. This thesis adds a new perspective of solving scheduling in HMS in adding time-critical constraints in static and dynamic environments to assure the safety and predictability of the production system.

4.3 From RT Operating Systems to RT Manufacturing

To increase throughput most modern operating systems support multiprogramming. The total time to complete a set of tasks is reduced due to less time being wasted while waiting for external inputs. The sequencing of such a multitasking or multiprogramming system is controlled by an operating

Real-Time constraints	**RT-off-line scheduling**	**RT-on-line scheduling**
Deterministic	Off-line scheduling	Dynamic scheduling
Stochastic	Stochastic considerations in scheduling	Reactive scheduling, pro-active scheduling
	Static	Dynamic

Figure 4.2: The different perspectives of the scheduling problem in HMS and the different approaches to solve it

system service called the scheduler. *Preemptive multitasking* is used in most operating systems designed for efficient multitasking.

Multitasking in manufacturing means production of multiple parts simultaneously on a single machine. It is represented in Fig. 4.3 with the dashed directed arcs and described in Sect. 4.5. It is also important to note that the changeover time for the different parts must be taken into consideration. There is a difference to real-time operating systems where the overhead of context switching is negligibly small compared to the execution time of tasks or is considered constant and included in the worst-case computation time.

Multiprocessing means the assignment of parts to the next available machine. The Fig. 4.3 gives an example of this architecture consisting of five manufacturing cells. Each cell comprises a machine with an input and an output buffer containing a set of parts. Multiprocessing is represented in Fig. 4.3 by the straight directed arcs.

4.4 System Characteristics

The characteristic problem of real-time systems, namely predicting timely correct behaviour of tasks can be solved, if a complete off-line analysis of all application tasks including the operating system and the environment is possible.

An arbitrarily complex manufacturing system cannot be analysed easily to predict its worst-case behaviour. It is necessary to define some restrictions on the structure that a real-time manufacturing system can have.

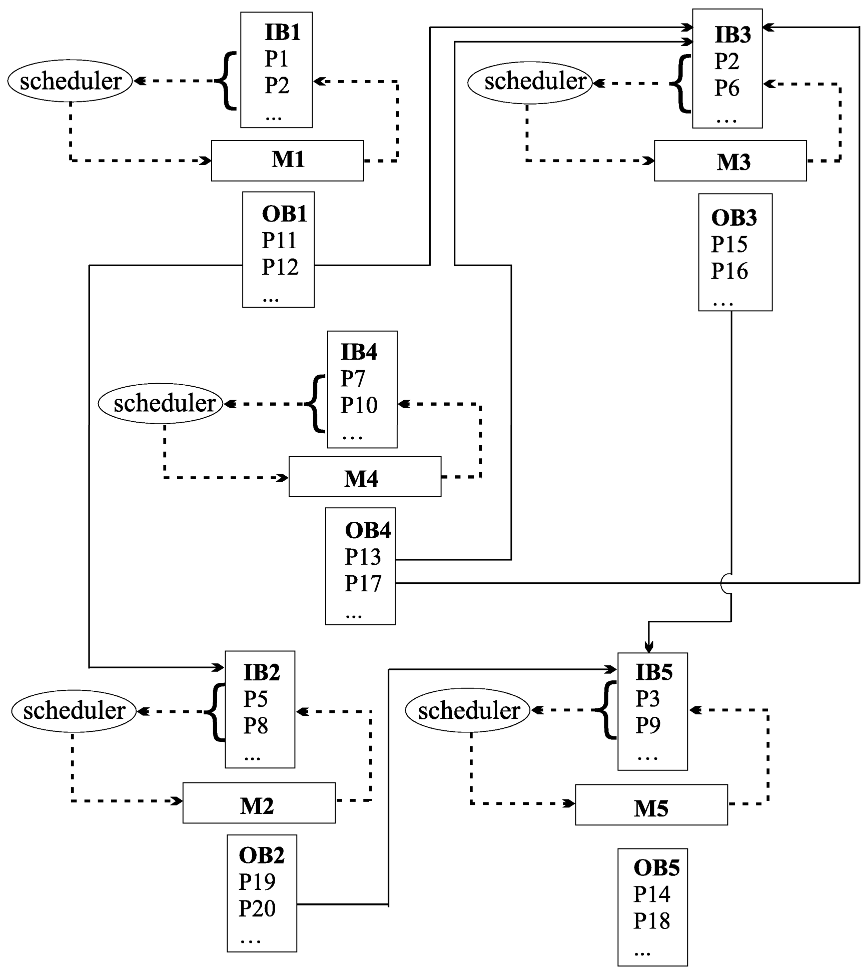


Figure 4.3: Example of an architecture for the on-line manufacturing system

- A uniprocessor and/or multiprocessor platform at each production stage is considered.
- The manufacturing system is assumed to consist of a fixed set of parts.
- Parts are periodic and aperiodic.
- Off-line preplanned requests in a production system under hard real-time constraints correspond to the normal (periodic) functions of the system. Off-line requests are assumed to be preplanned by a Production Planning System (PPS) in the manufacturing environment. An integration of the PPS in the production execution under real-time constraints requires a 100% deadline guarantee of the planned processes.
- Parts are completely independent of each other.
- Parts can be preempted at any time.
- More flexible methods for manufacturing with interleaving of different production tasks takes place. Thus, the production of one product on one machine is substituted by producing different products on the same machine.
- Overheads of context switching time (changeover cost) is accounted for in the algorithm and the schedulability test.
- All parts should be schedulable using worst-case processing times and worst-case arrival rates.

The listed characteristics applies to chapters 4, 5, 6 and 7. Further system characteristics are defined in the following chapters when it is necessary.

4.5 Computational Model for Preemptive Multitasking in Manufacturing Sytems

This simplified, abstract computational model to represent the behaviour of a preemptive multitasking manufacturing system is derived from the hard real-time schedulability theory (Fidge 1998).

The model assumes that the machine programmer wishes to implement *part sets* on a particular machine. Each part P_i arrives infinitely often, each arrival separated from the last one by at least T_i time units. A *periodic* part

C_i	Worst case processing time that may be required by an invocation of part P_i.
T_i	Lower bound between two succesive arrivals of part P_i.
D_i	The deadline for each invocation of part P_i, measured from its arrival time.

Figure 4.4: Part characteritics specified by the programmer of the machine

arrives regularly with a separation of *exactly* T_i time units. Aperiodic parts, which arrive irregularly with no minimum separation, are out of the subject of this chapter. Following chapters take into account aperiodic real-time scheduling.

At each arrival, part P_i issues a notional *invocation request* for up to C_i units of processing time on the machine, its *worst-case processing time.* Each invocation of part P_i must have this request satisfied before its deadline D_i expires. This has to be achieved by executing a worst-case chedulability test.

The scheduler goes through these requests according to a particular *scheduling policy.* Each part making a request is put in a *ready buffer* at which time the part is said to be *released.* The scheduler selects a part to be processed from the ready queue with respect to the scheduling policy it implements. Parts of higher priority can *preempt* the processing part P_i, resulting in a degree of interference I_i to the progess of part P_i. Parts stop being ready by being *suspended* due to parts with higher priority. Suspended parts wait in the input buffer until they become ready again.

Scheduling decisions are based on the *priority* of ready parts. The dynamic-priority scheduling policy *Earliest Deadline First* is considered. The Fig. 4.4 shows the part characteristics.

As already stated in Section 4.4, parts are considered to be independent, i.e. they do not interact or otherwise communicate using shared resources and hence cannot block one another.

4.6 An Extended Schedulability Test for EDF

A common scheduling technique used in real-time systems to dynamically schedule a set of periodic tasks is based upon the Earliest Deadline First algorithm. The dynamic priority assignment and the high schedulable utilization (up to 1) are the main advantages of EDF compared to other dynamic scheduling techniques of periodic tasks such as FIFO (First-in-First-out) and LIFO (Last-in-First-out).

To determine whether the given system of n independent periodic tasks surely meets all the deadlines when scheduled according to the preemptive EDF algorithm on one processor, the following inequality is to be checked

$$\sum_{k=1}^{n} \frac{C_k}{min(D_k, T_k)} \leq 1$$

Applying this schedulability test to the EDF-scheduling of manufacturing systems requires changeover time considerations. Therefore, each execution of a part suffers from at least one changeover time overhead when it starts execution and another changeover time overhead when it completes.

The following example describes a roboter boring process with changeover time considerations: A roboter processes wood panels using different boring tools placed on a circular plate. When a workpiece arrives, (1) the circular plate of the boring tools is rotated to reach the appropriate tool (2) the roboter arm picks up the boring tool, (3) the roboter arm moves to the workpiece and processes it, (4) the roboter arm returns the boring tool to the circular plate, and (5) the roboter arm returns to its initial position. Consequently, changeover time should be accounted for before (steps 1 and 2) and after (steps 4 and 5) processing a part.

It can be accounted for this changeover time overhead in a schedulability test by adding the time spent for the two changeover time overheads at the start and completion of each part to the production time of the part.

More formally, let $O = \{O_{j,m} \mid j, k \in \mathbb{N}\}$ be the switch time or changeover time caused by the arrival of the part from type j at the machine m. If the part can be preempted for a maximum of Q_i times after its production starts, the schedulability test of EDF including changeover time is deduced from the following inequation

$$\sum_{k=1}^{n} \frac{C_k + 2(Q_i + 1)O_{j,m}}{min(D_k, T_k)} \leq 1$$

It should be noted that changeover time overhead is derived by a static analysis of parts belonging to the same type on the machine.

4.7 Contributions of the Chapter

The scientific contributions of this chapter can be summarized as follows:

- *Development of a Taxonomy for the Real-Time Scheduling Problem*
  A taxonomy for the real-time scheduling problem is proposed where scheduling algorithms are organized with respect to the type of workload they can handle, the target machine they schedule on, the characteritics of the algorithm itself and the jobs they intend to schedule.

- *Review of Holonic Manufacturing Scheduling Schemes*
  A complete and brief literature review of holonic manufacturing scheduling schemes is provided. A holonic scheduling scheme supporting real-time considerations was not found in the literature survey.

- *Assignment of the Local Scheduling Policy and the Local Schedulability Test for MaSHReC*
  The scheduling policy used for the local scheduling of MaSHReC is based on the Earliest Deadline First algorithm. An extended schedulability test for EDF, including the novel feature of integrating changeover time overhead, is proposed.

4.8 Hypotheses Evaluation

This chapter presents a local scheduling scheme that assigns different parts to be produced simultaneously on the same machine. This validates the following hypothesis.

Timeliness and predictability of the local scheduling scheme presented in this chapter are a basis for the validation of Hypotheses 5.1 and 5.3.

Hypothesis 5 *The support of basic properties, including the following ones, is a major challenge in Production Planning and Control Systems underlying real-time constraints.*

1. ***Timeliness.*** Results have to be correct not only in their value but also in the time domain.

...

3. ***Predictability.*** One of the most important property that differentiates real-time systems from other conventional systems. The system must be able to predict the consequences of any scheduling decisison. If some task cannot be guaranteed within its timing constraints, the system must notify this fact in advance, so that alternative actions can be planned in time to cope with the event.

Bibliography

Bongaerts, L. (1998). *Integration of Scheduling and Control in Holonic Manufacturing Systems.* Ph. D. thesis, PMA/K.U. Leuven.
Develops shop floor control algorithms based on the notion of holonic manufacturing systems.

Dussa-Zieger, K. (1998). *Model-Based Scheduling and Configuration of Heterogeneous Parallel Systems.* Ph. D. thesis, University Erlangen-Nürnberg.
Studies the problem of scheduling in heterogeneous multiprocessor systems. Uses and implements heuristic algorithms based on genetic algorithms and tabu search.

El-Rewini, H. and H. Ali (1995). Scheduling of Conditional Branches in Parallel Program. *Journal of Parallel and Distributed Computing 24*, 41–54.

Fidge, C. (1998, January). Real-Time Schedulability Tests for Preemptive Multitasking. *J. of Real-Time Systems 14*(1), 61–93.
A tutorial acting as a guide to the major schedulability tests available for preemptive multi-tasking applications.

Gou, L., T. Hasegawa, P. Luh, S. Tamura, and J. Oblak (1994, October). Holonic Planning and Scheduling for a Robotic Assembly Testbed. In *Proc. of the Rensselear's fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, New York. Rensselear Polytechnique Institute.
Applies the holonic concept to a simple robotic assembly testbed. Establishes cooperation mechanisms for planning and scheduling among holons based on an adaptive consistency algorithm and the Lagrangian relaxation technique.

Kádar, B., L. Monostori, and E. Szelke (1997). An object-oriented framework for developing distributed manufacturing architectures. In L. Monostori (Ed.), *Proc. of the Second World Congress on Intelli-*

gent Manufacturing Processes and Systems, Budapest, Hungary, pp. 548–554.

Márkus, A., T. Kis, J. Váncza, and L. Monostori (1996). A market approach to holonic manufacturing. *CIRP Annals 45*(1), 433–436.
Introduces a market mechanism for coordinating the activities of intelligent agents that pursue their own interest by operating under bounded rationality in a changing, hardly predictable environment.

Márkus, A. and J. Váncza (1996, September). Are manufacturing agents different? In S. B. S. Albayrak (Ed.), *Proc. of the European Workshop on Agent-Oriented Systems in Manufacturing*, Berlin, Germany, pp. 86–103. Daimler-Benz AG and T.U. Berlin.
outlines a prototype of an order-processing and scheduling system that has been built on a market mechanism.

Moriwaki, T., N. Sugimura, Y. Martawirya, and S. Wirjomartono (ASME 1992). Production scheduling in autonomous distributed manufacturing system. *Quality Assurance Through Integration of Manufacturing Processes and Systems PED-Vol. 56*, 175–186.

Ramos, C. (1996). A Holonic Approach for Task Scheduling in Manufacturing Systems. In *Proc. of the IEEE International Conference on Robotics and Automation.*

Ramos, C. and P. Sousa (1996, September). Scheduling Orders in Manufacturing Systems using a Holonic Approach. In S. B. S. Albayrak (Ed.), *Proc. of the European Workshop on Agent-Oriented Systems in Manufacturing*, Berlin, Germany, pp. 80–85. Daimler-Benz AG and T.U. Berlin.
Presents an approach to resource allocation in holonic manufacturing based on the contract net protocol.

Sousa, P. and C. Ramos (1997). A Dynamic Scheduling Holon for Manufacturing Orders. In L. Monostori (Ed.), *Proc. of the Second World Congress on Intelligent Manufacturing Processes and Systems*, Budapest, Hungary, pp. 542–547.

Tönshoff, H. and M. Winkler (1995). Shop Control for Holonic Manufacturing Systems. In *Proc. of the 27th CIRP International Seminar on Manufacturing Systems*, Michigan, USA, pp. 329–336. Ann-Arbor.

Váncza, J. and A. Márkus (1998). Holonic manufacturing with economic rationality. In E. W. G. on IMS & EPFL (Ed.), *Proc. of the European*

Workshop on Intelligent Manufacturing Systems (IMS-EUROPE-98), Lausanne, Switzerland, pp. 383–394.

Chapter 5

A Distributed Scheduling Algorithm for MaSHReC

Chapter 3 and 4 introduced the model and local scheduling scheme of MaSHReC respectively. It is the purpose of this chapter to study the scheduling problem for distributed manufacturing systems underlying real-time constraints. The investigation of recent distributed scheduling techniques and of the relevant literature of distributed real-time scheduling schemes is a significant part of this chapter. Following, the scheduling methodology is developed and implemented in a prototyped manufacturing cell. The performance of the proposed scheduling algorithm is analyzed via simulation sutdies.

5.1 Traditional Scheduling Theory vs RT Scheduling Theory

The scheduling theory of real-time systems addresses the problem of meeting the specified timing requirements. Satisfying timing requirements of real-time systems demands the scheduling of system resources according to some well-understood algorithms so that the timing behaviour of the system is understandable, predictable, and maintainable. Scheduling theory is not only restricted to the study of real-time systems, it also arises in the study of manufacturing systems, transportation systems, process control systems, and so on. However, it is important to realize that real-time scheduling problems are different from the scheduling problems usually considered in areas of operations research (Stankovic and Ramamritham 1993). The most operations research distributed scheduling problems consider a fixed system having completely specified and static service characteristics. The goal is to find optimal static schedules that minimize the response time for a given task set. Such algorithms must be based on heuristics, since these scheduling problems are NP-hard. The most real-time systems negligates the goal of traditional operations research problems. The goal is to meet the deadlines for a given task set. The system is dynamic, requiring on-line, adaptive scheduling algorithms. The scheduling problem in its general form is also NP-hard.

Therefore, the emphasis in this chapter is on heuristic algorithms which provide hard real-time constraints for periodic tasks and firm real-time constraints for aperiodic tasks.

5.2 Related Work

An overview of scheduling solutions for HMS was presented in Section 4.2 page 44. Consequently, it was shown that these algorithms are not applicable to the approach presented in this thesis. Many distributed scheduling algorithms have been proposed for traditional distributed systems. Most research on scheduling in traditional distributed systems is restricted to *load balancing* algorithms (Casey 1981), (Eager et al. 1986), (Livny and Melman 1982), (Stankovic 1984), (Stankovic 1985a), (Stankovic 1985b), (Stankovic and Mirchandaney 1986), and (Shirazi et al. 1995). Most of the load balancing algorithms do not consider timing constraints of tasks and cannot be applied to real-time systems (Stankovic and Ramamritham 1988b).

A considerable research effort has been conducted in the area of real-time distributed scheduling over the last three decades. Almost all real-time distributed scheduling algorithms in the literature are based on heuristics. Instead of presenting a detailed survey of the research on real-time distributed systems, a brief overview of all relevant surveyed contributions is depicted and presented in a tabular structured form in Figure 5.1[1] page 62, thus allowing the reader to get an overview of the literature and to establish a comparative study among the different problems addressed in the literature and the problem faced in this chapter. In addition, a brief overview and commentary of the investigated literature is presented in the bibliography at the end of the chapter. However, because the scheduling scheme presented in this chapter is derived from the focused addressing and bidding techniques proposed by Ramamritham and Stankovic (1984) and Stankovic and Ramamritham (1988b), a distinction of these two contributions with the one presented in this chapter is proposed in the following.

1. **Application area.** An important property of a scheduling technique is its application area. While the scheduling scheme presented by Ramamritham and Stankovic (1984) and Stankovic and Ramamritham (1988b) finds its application in real-time operating systems, the heuristic scheduling described in this chapter is applied to real-time production planning and control systems.

2. **Structure of the system.** The scheduling technique developed by Ramamritham and Stankovic (1984) and Stankovic and Ramamritham (1988b) works on loosely coupled distributed systems. The heuristic presented in this chapter inherits its structre from the model developed in Chapter 5.7.

3. **Changeover time cost.** Although the distributivity of the scheduling problem is the same, the costs of changeover time and possible

[1]The references in the first column of Figures 6.1 and 6.2 are composed as follows: when the publication is written by a single author, the reference is composed of the first 3 letters of his family name followed by the year of publication. When the publication is written by more than one author, the reference is the composed of the first letter of the family names of the authors followed by the year of publication.
The abbreviations in the first line of the tables refer to the following terms: sched.: scheduling, alg.: algorithm, PT: Periodic Task, ST: Sporadic Task, AT: Aperiodic Task, Com.: Communication, CST: Context Switch Time, Int.: Integer, par.:parameters, per.: periods, SM: Shared Memory, DM: Distributed Memory, Ind.: Independent, Prec. Cons.: Precedence Constraints, JC: Jitter Control, RM: Resource Management.
The abbreviations in the rest of the table refer to the following terms: D: distributed, U: uniprocessor, M: multiprocessor, H: hard, S: soft, Dy: Dynamic, Fi: fixed

References	Year	Plat-form	Heuristic sched.	Predi-ctable sched.	Serv. alg.	Off-line	On-line	PT	ST	AT	Com. time	Pre-emption	No preem-ption	CST = 0	CST > 0	Int. sched. par.	Int. per.	Prio-rity	SM	DM	Ind. Tasks	Prec. Cons.	JC	RM	Over-load
RSZ89	1989	D	X				X	X		X	X											X			
TM89	1989	D,U		X				H		S	X							Dy							X
SR91	1991	D,M	X			X		X		S															
BS93	1993	D	X				X	X	X			X		X											
SS93	1993	D				X		X				X													
Bet94	1994	D		X			X	X																	
Lar94	1994	D	X			X					X			X								X			
TC94	1994	D		X			X	X	X		X			X				Fi			X				
TK94	1994	D						X	X																
TL95	1995	D	X								X			X											
Foh95	1995	D				X		H		H, S	X	X										X			
AS95	1995	D	X			X		H			X											X		X	
HLF95	1995	D	X			X		X																X	
DS95	1995	D	X			X		X					X									X	X		
LH95	1995	D,M		X								X	X	X							X				
RM95	1995	D,M						H				X						Fi							
SL96	1996	D						X										Fi			X				
PSA97	1997	D	X					X			X			X								X			
SFOC97	1997	D	X					X			X	X													
AH98	1998	D,M	X			X		X																	
WBP98	1998	D		X				X	X		X	X							X						
LL98	1998	D	X											X											
BB99	1999	D,U		X			X	X					X	X								X			
PBWB00	2000	D,M					X	X	X		X	X													
AS00	2000	D	X				X	X			X	X		X								X			
SHAMTSL01	2001	D					X																		
MAF01	2001	D		X			X				X														
CC01	2001	D	X			X					X											X			

Figure 5.1: Real-time distributed algorithms review

preemption of tasks in a Production Planning and Control System are significantly higher than in real-time operating systems.

4. **Transportation time cost.** While the algorithm treated by Ramamritham and Stankovic (1984) and Stankovic and Ramamritham (1988b) considers communication overhead due to the control software, the algorithm presented in this chapter should additionally account for parts transportation time overhead. It should be noted that the dominant factor for scheduling in a production system is the machinery. Consequently, transportation time significantly influences timing considerations.

5. **Local guarantee routine.** The guarantee routine presented by Ramamritham and Stankovic (1984) and Stankovic and Ramamritham (1988b) is extended to include the schedulability test developed in Section 4.6 page 50.

The reader should additionally note that according to the literature investigation presented in Figure 5.1, no real-time distributed scheduling algorithm accounts for changeover time costs.

5.3 The Structure and Behaviour of the Cell Coordinator Holon

Each cell in MaSHReC has a Cell Coordinator Holon. One function of the cell coordinator Holon is the local scheduling of the manufacturing cell (please refer to Section 4.6 page 50). A set (possibly empty) of preplanned parts exits at each cell. Aperiodic parts may arrive at any cell in the manufacturing system. When a new part arrives, the Cell Coordinator Holon checks whether the part can be scheduled at that cell so as to finish before its deadline; if so, the part is *guaranteed.* Otherwise, the Cell Coordinator Holon of the Cell interacts with Cell Coordinator Holons of other cells, using a scheme that combines *bidding* and *focused addressing*, in order to determine whether the part can be sent to be scheduled. Upon arrival at that cell, another attempt is made to schedule the part there. Eventually the part either gets guaranteed and executed or is not guaranteed.

While preplanned parts are guaranteed parts, aperiodic parts, after they arrive, may or may not be guaranteed. However, once guaranteed, they will definitely meet their deadlines.

The structure of the Cell Coordinator Holon is composed of a

- local scheduler,
- bidder, and a
- dispatcher.

Their functions are discussed next.

5.3.1 Bidder and local scheduler

Parts may arrive directly at a cell (aperiodic parts) or at the start of their release time (preplanned parts) or as a result of the interaction between Cell Coordinator Holons of different cells. Parts arriving at a cell are handled by the local scheduler and may or may not cause preemption. Conditions under which preemption is permissible are derived below.

The local scheduler first calls the guarantee test provided in Section 4.6 page 50 in order to guarantee the part locally. If it is guaranteed, then it is stored in the input buffer. Guaranteed parts are dispatched sequentially according to Earliest Deadline First algorithm.

If the part is not guaranteed locally, it is handled by the bidder. The bidder is the component that is involved in the distributed aspect of parts scheduling. The bidder part is the subject of section 5.5.

5.3.2 The dispatcher

It is the dispatcher part that determines which of the guaranteed preplanned parts and aperiodic parts is to be executed next. As mentioned in section 4.6, the EDF is optimal in the sense that it can achieve a schedulable utilization equal to 1. In our scheme, preplanned parts are produced according to the EDF scheme. It should be mentioned that our use of EDF on a single cell does not guarantee an optimal schedule of the manufactruing system as a whole.

The dispatcher's action are simple: whenever a part completes, the dispatcher is invoked, and it selects the next part with the earliest deadline for execution. The input buffer is ordered according to EDF. The run-time of the dispatcher is part of the processing time of every part.

5.4 The Data Structure of the Cell Coordinator Holon

For the purpose of scheduling, information on preplanned parts, such as their deadlines and computation times, is maintained in a data structure called the Preplanned Part Table (PPT). During the operation of the system the guarantee algorithm uses the PPT in conjunction with the concept of a *surplus* to ascertain whether a aperiodic part can be guaranteed. Surplus is derived from information in the System Part Table (SPT). Both SPT and the surplus are described below.

5.4.1 The system part table

Each cell maintains an SPT for all local preplanned, aperiodic, and aperiodic high priority parts guaranteed at any point of time. In the SPT, there is one entry per part which contains the part's arrival time, its latest start time, deadline and computation time. All but the latest start time are inputs. Entries for parts that have already arrived are ordered according to their deadlines. Note that parts may arrive from various sources, and different parts may have the same arrival time.

To compute the latest start time of a part with deadline D and, all preplanned parts with deadlines greater than or equal to D are ordered aconding to the decreasing priority. The latest start time is determined by assuming that parts are scheduled to execute just in time to meet their deadlines. For example, if the first part on the list with priority $R1$ has a computation time $C1$ and a deadline $D1$, it has a latest start time of $D1 - C1$. Suppose the part with priority $R2$, computation time $C2$ and deadline $D2$. If $D2$ is greater than $D1 - C1$, then the part has a latest start time of $D1 - C1 - C2$, otherwise $D2 - C2$. In this manner, latest start times are calculated for every guaranteed parts.

5.4.2 Surplus

Clearly, an aperiodic part can be guaranteed to execute at a cell only if the surplus production time at that cell, between when the part arrives and its deadline, is greater than the processing time requirement for the part. Thus, we are interested in surplus with respect to the part about to be guaranteed or rejected. Surplus, then, is defined as the amount of processing time available on a cell between the time of arrival of the new unguaranteed part and its deadline.

While surplus is not explicitly calculated for guaranteeing local parts, surplus information is implicitely taken into account during the computation of the latest start times for such parts: only if the latest start time is greater than the arrival time of the part, the part is guaranteed. Surplus is computed explicitly while responding to a request for bid (see section 5.5).

5.4.3 The guarantee routine

The guarantee routine local to a cell is invoked to determine if there is enough production time to execute a newly arriving part before its deadline. An aperiodic part can be guaranteed only after ascertaining that guaranteeing a part does not jeopardize previously preplanned parts. If the aperiodic part cannot be guaranteed locally, the part becomes a candidate for bidding and/or focused addressing (see section 5.5). The guarantee routine uses information in the PPT and SPT. Recall that each entry in the SPT contains an arrival time, a latest start time, a deadline, and a processing time. Note that the guarantee routine is coded assuming that before it is called, the SPT has been updated to reflect the current state of the cell.

5.4.4 Considerations of time overheads in scheduling

One of the prime motivation for the above separation of scheduling activity among various scheduling parts is to take into account the time spent on scheduling. Although the control software in manufacturing system is not a dominant factor, especially when compared to the mechanical machinery overheads such as transportation time and changeover time considerations, but it is unavoidable to consider it in manufacturing systems underlying *hard* real-time constraints.

Aperiodic parts are to be examined soon after they arrive. But interrupting a part being produced to guarantee a newly arriving part might result in the running part missing its deadline. This is solved as follows: after the production of the next part to be produced starts, the *check routine* checks if there is sufficient surplus such that running the bidder or the local scheduler, after preempting the newly dispatched part, does not result in guaranteed parts missing their deadlines. If the above is true for the bidder (local scheduler), then the check routine sets the invoke bidder flag to true. If a part arrives from another cell that requires the attention of the bidder, and the invoke bidder bid is set, then the currently running part is preempted and the bidder is executed. If instead, a part arrives locally and the invoke local scheduler flag is set, then the currently part being produced is preempted and the local scheduler part is executed.

5.5 Distributed Part Scheduling

Cell Coordinator Holons are responsible for aperiodic parts that cannot be scheduled on that cell itself. It interacts with the schedulers on other cells in an attempt to find a cell that has sufficient surplus to guarantee the part. This interaction is based on a scheme that combines focused addressing and bidding (Smith 1980).

5.5.1 The focused addressing scheme

Focused addressing utilizes system surplus information to reduce overheads incurred by bidding in determining a good cell to send a part to. It is described in the following.

A cell, before sending request for bids (RFB), uses the surplus information about other cells in the system to determine if a particular cell has a surplus which is significantly greater than the processing time of the new part, which cannot be guaranteed locally. Such a cell has a high probability of guaranteeing this new part, and hence the new part can be sent directly to that cell.

The following procedure is executed to determine the appropriate cell for focused addressing: Estimate the time ART when the part will arrive at the selected cell. If the estimated surplus of that cell, between ART and the deadline D of the part, is FP times the processing time C of the part, then the part is sent to that cell. (The computation of ART and of the surplus between ART and D will be described subsequently, when the bidding scheme is presented. FP is an adaptive parameter used in focused addressing.) If a number of cells satisfy this requirement, one of them is chosen randomly. The chosen cell, refered to as the *focused node*, uses the guarantee routine to check if the arriving part can be guaranteed there. If there is no focused cell, then the bidding scheme is invoked. Also, should the focused cell fail to guarantee the part, and if time considerations permit, the bidding scheme can be invoked.

Rather than invoking bidding only when use of focused addressing fails to guarantee the part, we invoke bidding *while* communication with the focused cell is in progress. This should increase the probability of parts being guaranteed. In this scheme, when a cell sends a part to a focused cell, it also sends RFB messages to other cells with an indication that bids should be returned to the designated focused cell. If the focused cell is unable to guarantee the transferred part, it chooses a cell to send the part to based on bids sent by the bidding cells.

To facilitate this approach to distributed scheduling, every cell has to keep track of the surplus of other cells. Towards this end, the percentage of time during the next window is requested for preplanned parts of the same cell-type. (Window is a system parameter used in the estimation of scheduling delays, surplus, etc. See section 5.6). In addition surplus information is returned on messages that are exchanged in the bidding process. It should be pointed out that given the delays involved in message transmission, such surplus information will be outdated by the time it is received. Hence, the algorithm utilizes this information only to *estimate* the surplus of cells.

The following subsections examine the details of the bidding scheme.

5.5.2 The bidding scheme

Cells making bids do not reserve resources needed to execute the part for which they are bidding. When a part arrives at a cell as a result of its bid being accepted, the part is handled as though it arrived locally. It is possible that the cell is unable to guarantee the part since the cell's surplus has changed since it sent the bid. This can happen due to the arrival of parts as a result of previous bids.

A number of factors that affect bidding require estimations, for example, transportation delays, machine requirements of future parts, etc. In this section, we describe the bidding approach, assuming that the needed estimates are available. In the next section, we show how these estimations can be made.

We now describe the different phases of the bidding process in detail. The main functions of the bidder component on a cell are: sending out request for bids for parts that cannot be guaranteed locally, responding to the request for bids from other cells, evaluating bids, and responding to part awards.

5.5.3 Request for bids

For a part that cannot be guaranteed locally, a decision is made as to whether to transmit an RFB. This decision is based on calculating an earliest possible response time (ER) and a deadline for response, (DR). ER takes into account the fact that an RFB is produced on a remote cell and that a two-way

communication is involved with the bidder.
Hence,

$$\begin{aligned} ER = \; & Current\_time + Processing\_time_{bidder} \\ & +(2 * Worst\_Comm\_delay\_per\_message) \end{aligned} \tag{5.1}$$

where
$Processing\_time_{bidder}$ = Processing time of the bidder, and
$Worst\_Comm\_delay\_per\_message)$ = Worst communication delay for control message such an RFB or a bid.

DR takes into account the fact that after *DR* elapses, there should be sufficient time for

1. the bidder part to evaluate the incoming bids and determine the best bidder,
2. the part to be sent to the best bidder cell,
3. the part to be guaranteed by the local scheduler at the best bidder cell,
4. the machine to be prepared for producing the part, and
5. the part to be produced and completed before its deadline.

$$\begin{aligned} DR = \; & D - Processing\_time_{bidder} \\ & -EST(Transp\_delay\_for\_the\_part) \\ & -2 * (Changeover\_time\_for\_the\_part) \\ & -Processing\_time_{Local\_scheduler} - C \end{aligned} \tag{5.2}$$

where
$EST(Transp\_delay\_for\_the\_part)$ = Average transportation delay for transporting a part,
$Changeover\_time\_for\_the\_part$ = Changeover time for producing the part, and
$Processing\_time_{Local\_scheduler}$ = Processing time of local scheduler

If *ER* is greater or equal to *DR*, then there is no need to transmit an RFB. If *ER* is less than *DR*, then an RFB will be broadcast to all cells of the same type. The RFB message itself contains the following information: D, C, current time, and DR.

One way to reduce the communication overheads due to bidding is to send RFB's only to those cells which have a high probability of responding to the request. Such cells can be identified by using the surplus information about other cells.

Before the RFB is actually sent, the algorithm calculates the time at which to begin bid evaluation, $Time_{bid\_eval}$, where

$$\begin{aligned} Time_{bid\_eval} = \; & Current\_time \\ & +EST(Response\_time\_for\_RFBs) \end{aligned} \tag{5.3}$$

where
$EST(Response\_time\_for\_RFBs)$ = estimated delay between transmission of an RFB and the arrival of a bid.

If $Time_{bid\_eval}$ is less than DR, then the requesting cell waits until $Time_{bid\_eval}$ before evaluating bids. However, if $Time_{bid\_eval}$ is greater than or equal to DR, then we arbitrarily let $Time_{bid\_eval} = (ER + DR)/2$, with the hope that at least one reply arrives in time. Information about the part as well as the $Time_{bid\_eval}$ is placed in the wait_for_bid_queue.

5.5.4 Bidding in response to RFBs

The bidder first estimates if its response will reach the requestor before the deadline for response DR. It proceeds with further actions on the request for bid only if the time of response plus the communication delay for the response is less than the indicated deadline for response.

Once a cell decides to respond, it first computes ART, the estimated arrival time for the part, in case the cell is awarded the part. ART is one of the three components of the bid. Computation of ART is done as follows:

$$\begin{aligned} ART = \; & Current\_time \\ & +Worst\_Comm\_delay\_per\_message \\ & +EST(Bid\_Wait) \\ & +EST(Transp\_delay\_for\_the\_part) \\ & +EST(LS\_Wait) \end{aligned} \tag{5.4}$$

where

$EST(Bid\_Wait)$ = the estimated delay in processing a returned bid, and

$EST(LS\_Wait)$ = the estimated wait time experienced by a transported part at its new cell before it is either guaranteed or rejected.

The second component of the bid is the production time surplus at this cell between the estimated arrival time, ART, and the parts deadline D. We call this SARTD. The surplus information takes into account the following.

1. Future instances of preplanned and guaranteed aperiodic parts: this ensures that guaranteed parts are not jeopardized.

2. Processing time needed for parts that may arrive as a result of previous bids: this ensures that cells requesting bids are aware of other bids by a cell and hence minimizes the probability of a cell being awarded parts with conflicting requirements or being awarded too many parts creating an unstable situation.

3. Surplus resulting from parts that do not execute to their worst case processing time.

While accurate information is available concerning 1., information needed for the second item is estimated based on the past behaviour of the cell.

More precisely, SARTD is computed as follows:

Let $EST(Production\_time\_local\_between\_ART\_and\_D)$
= estimated production time required by local parts that execute on the cell between ART and D.

Let $EST(Production\_time\_bid\_between\_ART\_and\_D)$
= estimated production time required by parts that arrived due to bidding and that execute on the cell between ART and D

Then

$$\begin{aligned} SARTD = {} & \\ & ((D - ART) * (1 - Percent\_pre\_planned\_parts)) \\ & -[(EST(Production\_time\_local\_between\_ART\_and\_D) \\ & +EST(Production\_time\_bid\_between\_ART\_and\_D)) \\ & *EST(Part\_processing\_time\_ratio)] \end{aligned} \tag{5.5}$$

where

Percent_pre_planned_parts = the percentage of production time required by preplanned parts between ART and D, and

EST(Part_processing_time_ratio) = the average value for the ratio (actual production time used by a part / Worst case production time required by that part).

Finally , if SARTD is less than C, then no bid is made since the surplus is not efficient. If SARTD is greater than or equal to C, then a bid is returned with the information, ART, SARTD, and an estimation of how long a new part transported to to this cell will have to wait before it is processed for a possible guarantee.

5.5.5 Bid processing

Bid processing is carried out by the cell that originally sent out the request for bids. A bid part waits for bids returned in response to an RFB until either

1. required minimum number of bids are received (a tunable system parameter)
2. $Time_{bid\_eval}$.

Whether one or both of these factors is used is specified by a tunable system parameter. As soon as conditions 1 or 2 is met, and if any bids have been received, the evaluation of the received bids is started. For each bidding cell, the algorithm computes ETA, the estimated time of arrival of the part at the bidder's cell. For each bidder it estimates SETAD, the surplus between ETA and D using the following formula:

$$SETAD = SARTD * \left(\frac{D - ETA}{D - ART}\right) \tag{5.6}$$

If there is at least one bid whose SETAD is greater than or equal to C, then the bidder part chooses the one with the greatest SETAD as the best bid and the part is sent to the cell that sent the bid. If for all bids, SETAD is less than C, then the bid processor part waits for more bids until DR. For each new bid, if SETAD is greater than C, then the part is immediately sent to the bidder cell. The identity of the second best bidder, if any, is also communicated to the best bidder (see the next subsection).

One final note about the information sent on bids: a cell utilizes this information to keep track of the surplus in other cells. This is used for sending

RFB to cells with high surplus and in focused addressing. In response to a RFB for a part, a bidder send the estimated arrival time and the estimated surplus between the arrival time and the part's deadline. If a message does not respond to an RFB, it is assumed that it does not have sufficient surplus. In reality, a cell may not have sent its bid because it estimated that its bid would not reach the requester before the deadline for response. Information received via bids is bound to be fragmented, and hence, a cell, if needed, utilizes information about available free times sent periodically by other cells.

5.5.6 Response to part award

Once a part is awarded to a cell, the awardee cell treats it as a part that arrived locally at the cell and takes actions to guarantee it. If the part cannot be guaranteed, the cell can request for bids and determine if some other cell has the surplus to guarantee it. However, given that the part was sent to the best bidder and that the part's deadline will be closer than before, the chances of finding another cell with surplus are small. Hence, the decision is made to send not only the part but also the identity of the second best bidder to the best bidder: if the best bidder cannot guarantee the part, then, should time considerations permit, it sends the part to the second best bidder, if any. Otherwise, the part is rejected.

5.6 Estimation Techniques

This section shows how estimates used in the previous section are calculated using exponential smoothing.

Given a set of data $\{y_1, y_2, \ ... \ , y_n\}$, exponential smoothing transforms this set into the smoothed data $\{Y_1, Y_2, \ ... \ , Y_n\}$ by means of a weighted average with exponentially decreasing weights. If Y_m is the smoothed value, then we obtain the next smoothed value Y_{m+1} by means of the relation

$$\begin{aligned} Y_{m+1} &= Y_m + \alpha * (y_{m+1} - Ym) \\ &= \alpha * y_{m+1} + (1 - \alpha) * Y_m \end{aligned}$$

for $m = 0, 1, 2, ...n_1$.

Thus, the current smoothed value is an interpolation between the previous smoothed value and the current observation, where α controls the closeness of the interpolated value to the most recent observation and lies between 0 and 1.

Exponential smoothing is a simple and efficient forecast method because of its weighting process. The weights given to previous values are not equal; instead, they decrease with the age of the data. Since exponential smoothing relies on only two windows of data (the last period's actual value and the forecasted value for the same period), it minimizes the data storage requirements.

For calculating the estimates used in the precedent section using exponential smoothing, time is divided into time slots. A window is formed by five consecutive time slots. As time progresses, the window is moved, i.e., when the current time indicates the end of a time slot, the window is moved to occupy the most recent five time slots. Information such as the delay in processing bids (used to compute $EST(Bid\_wait)$), is gathered for each time slot. The cumulative information in the time slots forming a window is used to make the various estimations described below.

5.6.1 EST(Bid_wait)

To calculate the estimated bid wait, delays invoked in processing bids are gathered. $Window\_bid\_wait$ is computed to be the average *bid_wait* for bids received in the most recent window. The new value of $EST(Bid\_wait)$ is computed using its current value and the value of $Window\_bid\_wait$ via exponential smoothing, i.e.,

$$\begin{aligned} New\_EST(Bid\_wait) = {} & (\alpha * Window\_bid\_wait) \\ & +[(1-\alpha) * EST(Bid\_wait)] \end{aligned} \tag{5.7}$$

5.6.2 EST(LS_wait)

$EST(LS\_wait)$ is the estimation of how long a transported part waits before it is processed by a local scheduler part. This wait time is averaged with previous wait times in the same manner as described above for $EST(Bid\_wait)$, and estimation of the wait time in the future is done by single exponential smoothing.

5.6.3 EST(Response_time_for_RFBs)

Let $T_{receive}$ denote the time when a returned bid is placed in a queue awaiting processing, T_{send} be the time when the RFB for this bid was sent, and

$Turnaround\_time = T_{receive} - T_{send}$. EST(Response_time_for_RFBs) is the average Turnaround_time in window. The average turnaround time is computed over the past five time slots. Estimation is done via exponential smoothing.

5.6.4 EST(Production_time_local_between_ART_and_D) and EST(Production_time_bid_between_ART_and_D)

As seen in the last section, these are used to estimate the surplus between the estimated arrival time of a part, ART, and its deadline D. To do this, each cell maintains the following information:

PGLP = Production time required by Guaranteed Local Parts.

PGBP (Production time of Guaranteed Bidding Parts) = Production time required by parts acquired by bidding and guaranteed

PFBP (Production time of Future Bidding Parts) = Production time required by parts for which bids were sent out and may arrive in the future.

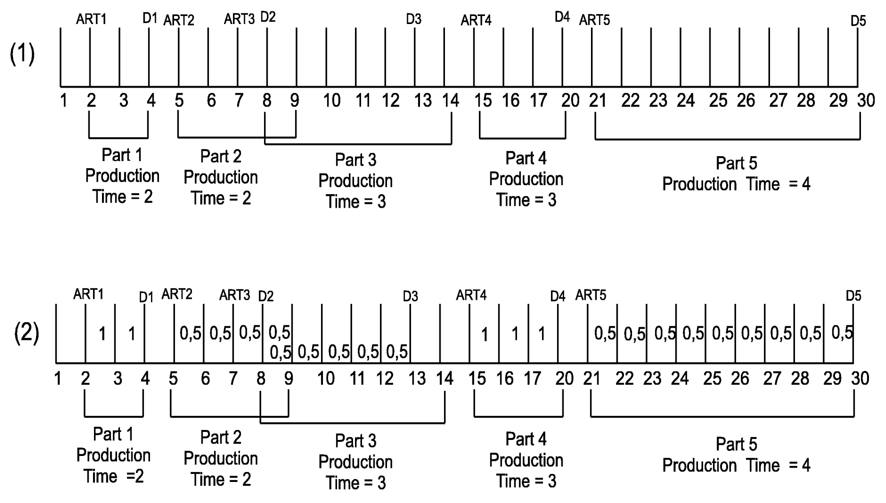
For the sake of clarity and better understanding, an example illustrating the different steps of $EST(Production\_time\_local\_between\_ART\_and\_D)$ computations is presented in Figure 5.2.

PGLP, PGBP, and PFBP information is maintained in an array of time slots. When a part is guaranteed, or a bid is sent, the production time of that part is proportionally divided among all time slots that lie between its ART and D so that it evenly affects all time slots which it overlaps. Refer to steps (1) and (2) of Figure 5.2. Note that in step (2), the production time of part 1, which is 2, is divided in two time slots of 1 production time unit. The production time of part 2, which is 2 is divided in four time slots of 0.5 time units, etc.

A cell also maintains WPGLP, which is the sum of the PGLPs in the previous five time slots. Refer to step (3) of Figure 5.2. Note that WPGLP at time 5 is equal to 2, WPGLP at time 6 is equal to 2.5, etc.. WPGBP, which is the sum of the PGBPs in the previous five time slots; and WPFBP, which is the sum of the PFBPs in the previous five time slots. This information is updated at the end of each time slot. Let

$$Percent\_PGLP = \frac{WPGLP}{window} \tag{5.8}$$

$$Success\_ratio\_of\_bids = \frac{WPGBP}{WPFBP} \tag{5.9}$$


(1)
ART1 D1 ART2 ART3 D2 D3 ART4 D4 ART5 D5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 20 21 22 23 24 25 26 27 28 29 30
Part 1 Production Time = 2
Part 2 Production Time = 2
Part 3 Production Time = 3
Part 4 Production Time = 3
Part 5 Production Time = 4
(2)
ART1 D1 ART2 ART3 D2 D3 ART4 D4 ART5 D5
1 1
0,5 0,5 0,5 0,5
0,5 0,5 0,5 0,5 0,5
1 1 1
0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5 0,5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 20 21 22 23 24 25 26 27 28 29 30
Part 1 Production Time =2
Part 2 Production Time = 2
Part 3 Production Time = 3
Part 4 Production Time = 3
Part 5 Production Time = 4

(3)

Time	5	6	7	8	---
WPGLP	2	2,5	3	2,5	---
Percent-PGLP	$\frac{2}{5}$	$\frac{2,5}{5}$	$\frac{3}{5}$	$\frac{2,5}{5}$	---

Figure 5.2: Illustration example of the estimation technique for local production time between ART and D

Refer to step 3 of Figure 5.2 for the computation of Percent_PGLP. EST(Percent_PGLP) and EST(Success_ratio_of_bids) are computed using single exponential smoothing.

Let PGLP2 be the production time required between ART and D by already guaranteed local parts. Then

$$\begin{aligned} &EST(Production\_time\_local\_between\_ART\_and\_D) = \\ &\quad max[EST(Percent\_PGLP) * (D - ART), PGLP2] \end{aligned} \tag{5.10}$$

Let PGBP2 be the production time required between ART and D of guaranteed parts which arrived via past bidding, and PFBP2 be the production time needed between ART and D for parts for which bids were sent out. Then

$$\begin{aligned} &EST(Production\_time\_bid\_between\_ART\_and\_D) = \\ &\quad max[EST(Sucess\_ratio\_of\_bids) * PFBP2, PGBP2] \end{aligned} \tag{5.11}$$

5.6.5 EST(Part_processing_time_ratio)

This is estimated to be the average of (actual processing time/worst case processing time) of the parts that completed during the past window.

In all above estimations, we assume that the clocks on different cells are synchronized. The only effect of slight asynchrony in the clocks will be that the estimates will be slightly inaccurate. It should be noted that parts are independent and are guaranteed only at the cell where they actually reside, therefore, a cell can guarantee a part based solely on local information. Hence asynchrony in the clock does not affect the guarantee algorithm itself.

5.7 Prototype Simulation

The following graphical simulation study is an attempt to analyze the algorithm's behaviour given the use of the local scheduling scheme developed in Chapter 4 and the distributed scheduling scheme developed in this chapter. Note that the simulation model applies the modeling techniques presented in Chapter 3.

The simulation model consists of four manufacturing cells coordinating with each other in order to execute periodic and aperiodic orders. Each manufacturing cell consists of a machine, an input and output buffer. A cell coordinator holon at each cell is responsible for the local scheduling of a manufacturing cell as well as the inter-cell coordination scheduling. The Production Planning System (PPS) manager holon is responsible for the off-line scheduling of orders and their communication to the cells. The simulation model is represented in Figure 5.3, the following labels refer to the following machinery and holons:

- $Mach\_Hol1$, $Mach\_Hol2$, $Mach\_Hol3$, and $Mach\_Hol4$ are machines.
- IB1, IB2, IB3, and IB4 are input buffers.
- OB1, OB2, OB3, and OB4 are output buffers.
- $Cell\_Coord\_Hol1$, $Cell\_Coord\_Hol2$, $Cell\_Coord\_Hol3$, and $Cell\_Coord\_Hol4$ are cell coordinator holons.
- PPS_Man_Hol is the PPS manager holon.

5.8 Performance Evaluation

In this section, the heuristic algorithm, which has been implemented to solve the aperiodic control problem addressed in this thesis, is evaluated. Since no control algorithm allowing aperiodic part scheduling in a production stage and shift exists[2], a comparison of the presented algorithm with other existing algorithms is not possible. Still, the performance of this algorithm has to be assessed.

The algorithm described in this chapter has been simulated on the 4-cells prototype model proposed in Section 5.7, in order to compare the deadline guarantee of tasks with different periodic and aperiodic loads.

The performance evaluation is based on a set of experimental simulations:

- Since changeover time in production control systems is a dominant factor, the diagram of Figure 5.4 reveals the considered changeover time costs compared to production time. Due to their considerable overhead (up to an overhead approximately equal to production time), changeover time costs involved in the proposed experimental simulations are moderately realistic.

[2]Literature investigations and publications of the author showed that the production control scheduling problem of interest in this thesis is still not solved.

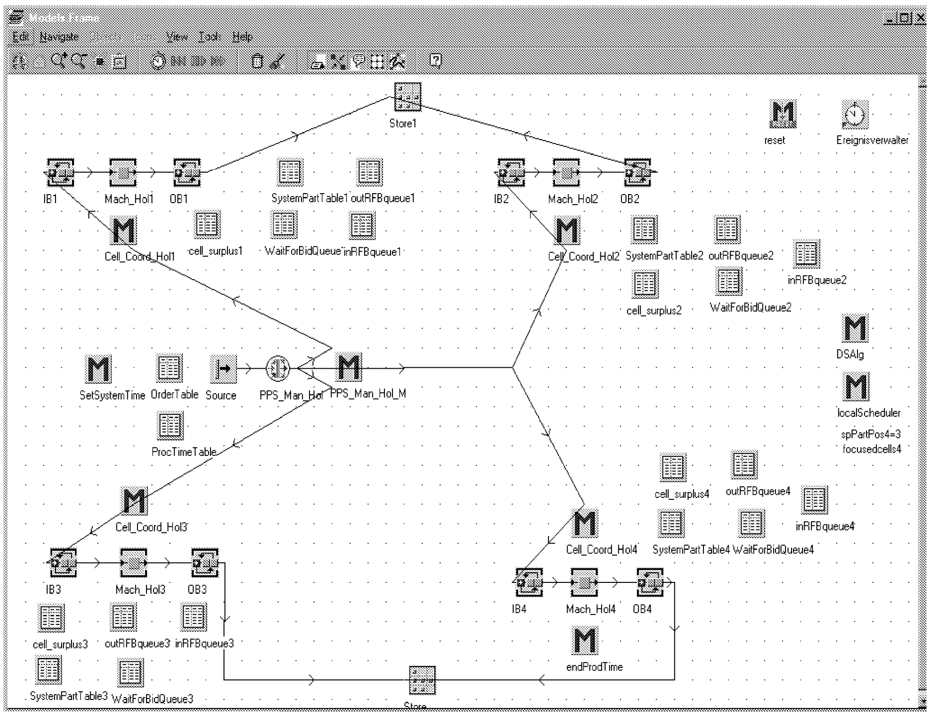


Figure 5.3: Prototype simulation

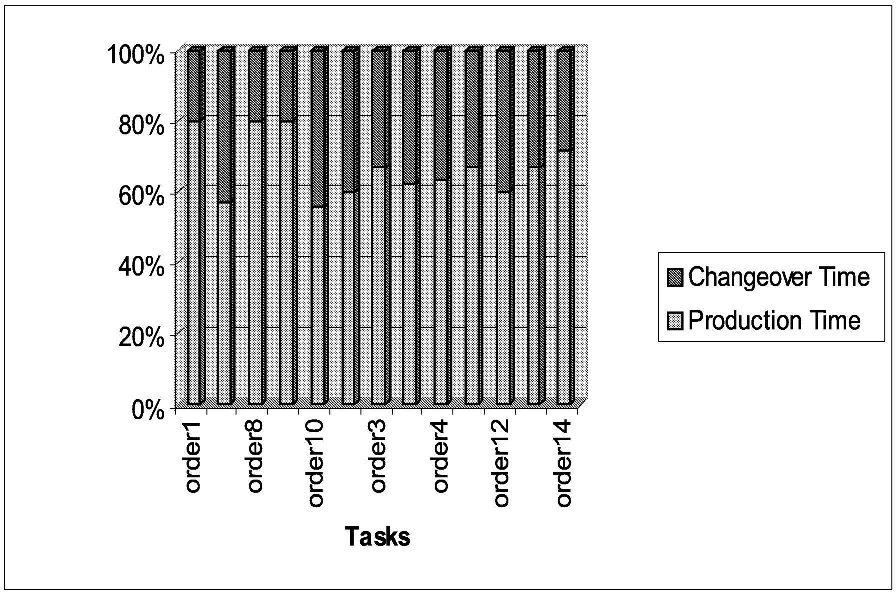


Figure 5.4: Changeover time versus production time

- First, a set of periodic tasks are produced with a periodic machine utilization of 0.66, 0.73, 0.9 and 0.41 for machine 1 of cell 1, machine 2 of cell 2, machine 3 of cell 3, and machine 4 of cell 4 respectively. The experimental simulations of Figure 5.5 have established that all periodic tasks meet their deadlines.

- In the second set of experiments, the effect of aperiodic tasks with a total aperiodic machine utilization of 0.238 is investigated. Analogous to the first set of experiments, the quality of solution is evaluated by comparing the guaranteed deadlines. Figure 5.6 has established that all tasks meet their deadlines.

- Furthermore, the proposed algorithm has been compared with higher aperiodic loads, in order to show peak load conditions. With a total aperiodic machine utilization of 0.412, Figure 5.7 shows that deadline guarantee of all tasks is achieved. However, with a total aperiodic machine utilization of 0.524, two aperiodic tasks miss their deadlines as shown in Figure 5.8.

- Due to high changeover and tranportation time overheads, the algo-

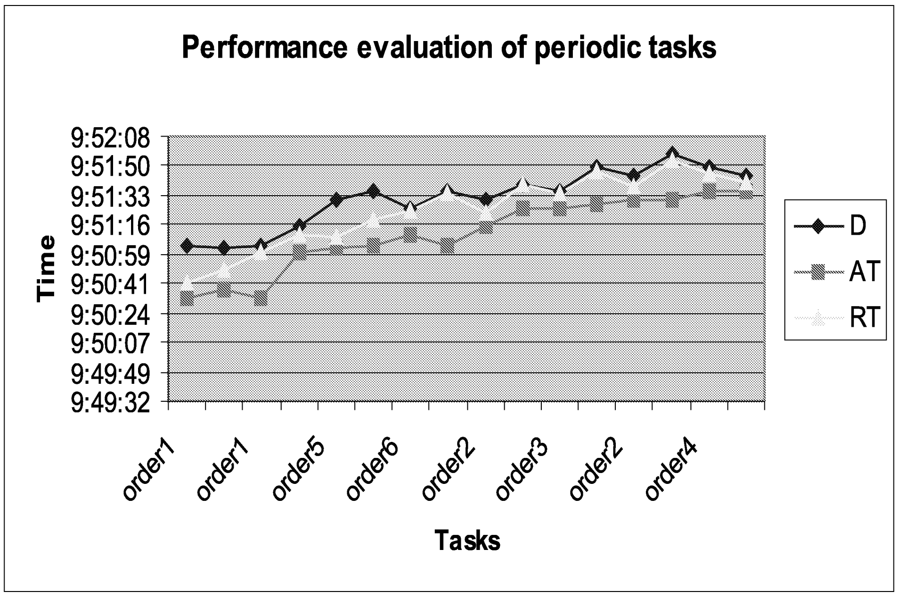


Figure 5.5: Performance evaluation of periodic tasks

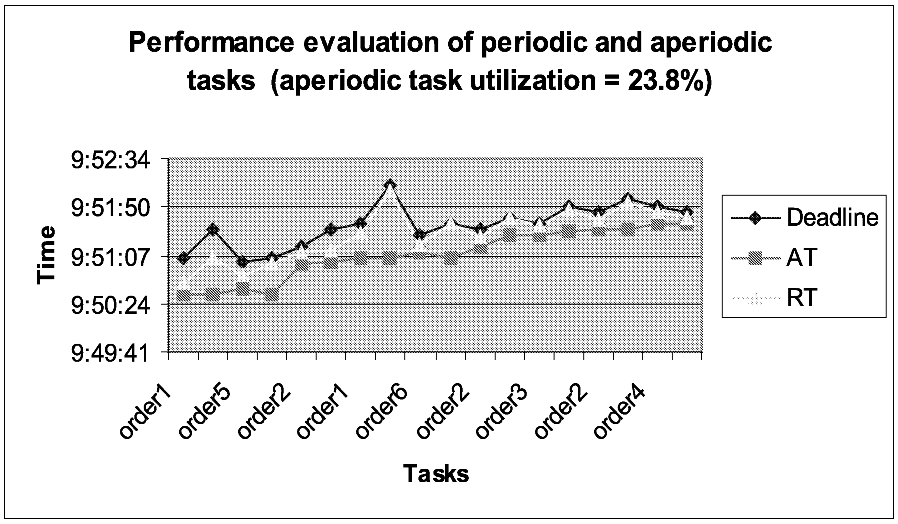


Figure 5.6: Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 23,8%

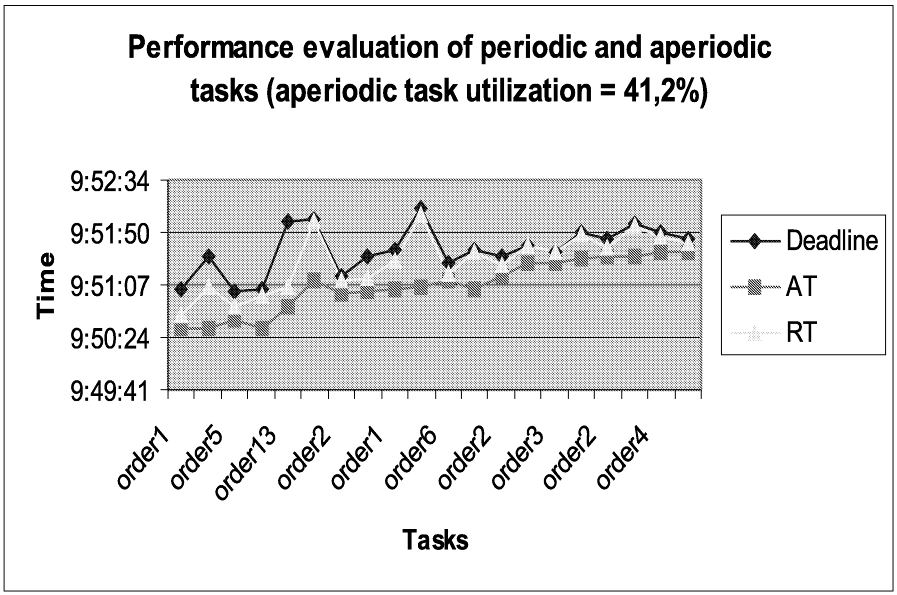


Figure 5.7: Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 41,2,8%

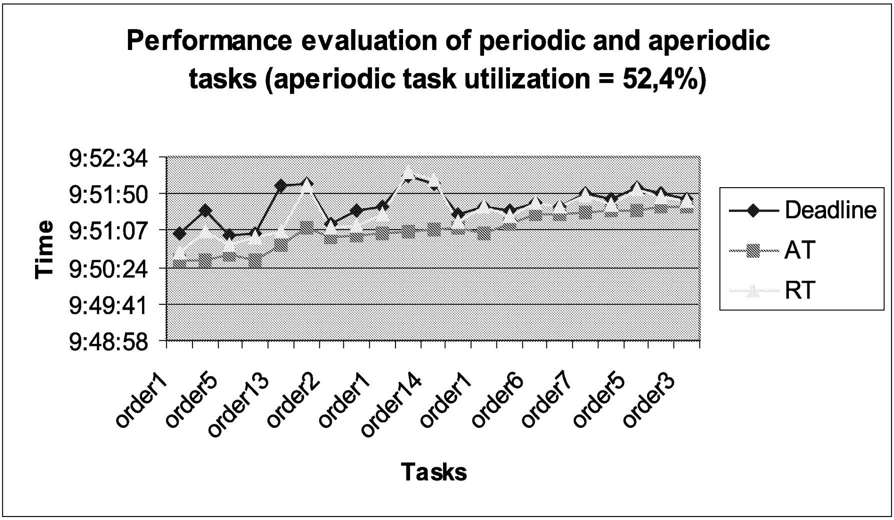


Figure 5.8: Performance evaluation of periodic and aperiodic tasks with an aperiodic task utilization of 52,4%

rithm would not perform better, when the aperiodic load is generated by a large number of tasks with low production time.

5.9 Contributions of the Chapter

The scientific contributions of this chapter can be summarized as follows:

- *Review of Recent Distributed Scheduling Techniques*
  A brief review of distributed scheduling techniques in manufacturing systems is provided. None of the techniques found in the literature supports real-time considerations.

- *A Classification and Review of Real-Time Distributed Scheduling Literature*
  A relevant and structured overview of real-time distributed scheduling theory is presented. The different algorithms are structured according to the problem they solve and to the scheduling parameters which are relevant for the real-time scheduling of manufacturing systems.

- *Development of a Distributed Scheduling Scheme for MaSHReC*
  A heuristic method, which solves the control problem for distributed production systems, is developed. It uses a scheduling component local to every node and a distributed scheduling scheme that is specifically suited to hard periodic and firm aperiodic real-time constraints. Preplanned parts, aperiodic parts, scheduling overheads, communication overheads due to scheduling, transportation overheads, changeover time costs and preemption are all accounted for in the algorithm.

- *A Prototype Implementation and Performance Evaluation of the Proposed Scheduling Scheme*
  A prototyped manufacturing system model is presented. The performance evaluation, via simulation studies, of the prototyped manufacturing system using the proposed scheduling scheme is analyzed.

5.10 Hypotheses Evaluation

Providing a scheduling policy allowing the production of periodic and aperiodic tasks in this chapter clearly validates hypotheses 3.

Hypothesis 3 *On-line requests in a production system under hard real-time constraints are aperiodic tasks to handle external events, which are usually unpredictable. Allowing additional orders in the manufacturing environment may lead to a high-level system utility.*

Bibliography

Abdelzaher, T. and K. Shin (1995, December). Optimal Combined Task and Message Scheduling in Distributed Real-Time Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 162–171.
Presents an optimal algorithm for combined task and message scheduling in distributed hard real-time systems. Uses a branch-and-bound technique to find an optimal solution to the problem.

Altenbernd, P. and H. Hansson (1998). The Slack Method: A New Method for Static Allocation of Hard Real-Time Tasks. *Real-Time Systems 15*, 103–130.
Presents and evaluates the Slack Method, a constructive heuristic for the allocation of periodic hard real-time tasks to multiprocessor or distributed systems.

Anderson, J. and A. Srinivasan (2000, June). Early-Release Fair Scheduling. In *Proc. of the EuroMicro Conference on Real-Time Systems*, Stockholm, Sweden, pp. 35–43. IEEE Computer Society Press.
Presents a variant of P-fair scheduling, the early-release fair (ERfair). ERfair differs from P-fair scheduling in that it has a lower average job response times and run-time costs, particularly in a lightly-loaded systems.

Bate, I. and A. Burns (1999). A Framework for Scheduling in Safety-Critical Embedded Control Systems. In *Proc. of the 6th International Conference on Real-Time Computing Systems and Apllications.*
Presents a computational model that supports the reuse of legacy systems. Develops timing analysis that features low pessimism and low computational complexity.

Bestavros, A. and D. Spartiotis (1993, May). Probabilistic Job Scheduling for Distributed Real-Time Applications. In *Proc. of the First IEEE Workshop on Real-Time Applications*, New York, NY.
Describes a heuristic for the dynamic real-time scheduling in a distributed

environment. When a task is submitted to a node, the scheduling software tries to schedule the task locally so as to meet its deadline. If it fails, it tries to locate another node where this could be done with a high probability of success.

Bettati, R. (1994). *End-to-End Scheduling to Meet Deadlines in Distributed Systems*. Ph. D. thesis, University of Illinois at Urbana-Champaign, Zrich.

Presents two algorithms for scheduling flow shops where tasks can be serviced more than once by some processors. Describes a technique to schedule flow shops that consist of periodic tasks and to analyze their schedulability.

Casey, L. (1981). Decentralized Scheduling. *The Australian Computer Journal 13*(2).

Chiu, J.-F. and G.-M. Chiu (2001, December). Placing Forced Checkpoints in Distributed Real-Time Embedded Systems. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Presents a scheme for placing forced checkpoints in a distributed real-time embedded systems so as to make all checkpoints useful for rollback recovery.

DiNatale, M. and J. Stankovic (1995, December). Applicability of Simulated Annealing Methods to Real-Time Scheduling and Jitter Control. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 190–199.

Introduces a scheduling approach, which minimizes jitter for periodic tasks in distributed static systems. Presents a general framework consisting of an abstract architecture model and a general programming model are .

Eager, D., E. Lazowska, and J. Zahorajan (1986, May). Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Trans. on Software Engineering SE-12*(5), 662–675.

Uses the *system decomposition* technique to evaluate three types of load sharing algorithms. This technique enables the entire system to be modeled in terms of a single node, replying upon the conjecture that the decomposition method is asymptotically exact as the number of nodes, N, becomes larger. Concludes that any redistribution strategy was better than none, and that simple policies were almost as effective as more complex ones.

Fohler, G. (1995, December). Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 152–

161.
Presents an algorithm for the joint scheduling of periodic and aperiodic tasks in statically scheduled distributed real-time systems.

Hsueh, C.-W., K.-J. Lin, and N. Fan (1995, December). Distributed Pinwheel Scheduling with End-toEnd Timing Constraints. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 172–181.
Presents an end-to-end scheduling model for real-time distributed system based on the pinwheel scheduling algorithm.

Larsson, E. (1994). *The Scheduling Tool.* Technical report ProVia-DoCs-94204, Department of Computer Sytems, Uppsala University.
Presents an off-line scheduling tool that maps a set of process graphs onto a particular system configuration. The tool accept as input process graphs of the RED processes to be scheduled, together with a target system description, and produces as output one schedule for each node executing RED processes.

Liu, J. W. S. and R. Ha (1995). *Advances in Real-Time Systems* (Sang H. Song ed.)., Chapter 9, Efficient Methods of Validating Timing Constraints, pp. 196–220. Prentice Hall.
Presents worst-case bounds and efficient algorithms for determining how late the completion times of independent jobs with arbitrary release times can be in a dynamic multiprocessor or distributed system when their release times and execution times may vary from one instance to another.

Livny, M. and M. Melman (1982, April). Load balancing in homogeneous broadcast distributed systems. In *ACM Computer Network Performance Symposium*, pp. 47–55.
Studies the probability that in a homogeneous distributed computing system, a customer waits for service at one node while at least one node is idle. Shows that for a wide range of system traffic intensity, this probability is close to one.

Martins, E., L. Almeida, and J. Fonseca (2001, December). Integrating Traffic Scheduling and Schedulability Analysis in an FPGA-based Coprocessor. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Presents a dedicated coprocessor to support the traffic management in the communication system. The coprocessor integrates both scheduling and schedulability analysis functions.

Peng, D.-T., K. Shin, and T. Abdelzaher (1997, December). Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems. *IEEE Transactions on Software Engineering 32*(12), 745–758.

Presents an optimal solution to the problem of allocating communicating periodic tasks in a distributed real-time systems. The task system is modeled with a task graph and are assigned to processing nodes by using a branch & bound search algorithm.

Poledna, S., A. Burns, A. Wellings, and P. Barrett (2000, February). Replica Determinism and Flexible Scheduling in Hard Real-Time Dependable Systems. *IEEE Transactions on Computers 49*(2), 100–111.

A method, called timed messages, to avoid the inconsistent order and timing of replicated tasks in real-time distributed systems is presented. The major advantage of timed messages is its efficiency and flexibility while guaranteeing deterministic operation of replicated tasks.

Ramamritham, K. and J. Stankovic (1984). Dynamic Task Scheduling in Hard Real-Time Distributed systems. *IEEE Software 1*(3), 65–75.

Deals with multiprocessor scheduling in hard real-time distributed systems. Uses a uniprocessor scheme for local scheduling, and perform distributed scheduling for tasks which are potentially subject to timing failures at run-time.

Ramamritham, K., J. Stankovic, and W. Zhao (1989, August). Distributed Scheduling of Tasks with Deadlines and Resource Requirements. *IEEE Transactions on Computers 38*(8), 1110–1123.

Evaluates four algorithms for cooperation in a distributed real-time system. They differ in the way a node treats a task that cannot be guaranteed locally.

Rhee, I. and G. Martin (1995, December). A Scalable Real-Time Synchronization Protocol for Distributed Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 21–27.

Proposes a distributed protocol for the synchronization of real-time tasks that have variable resource requirements. The protocol is intended for large-scale distributed or parallel systems in which processes communicate by message passing.

Santos, J., E. Ferro, J. Orozco, and R. Cayssials (1997). A Heuristic Approach to the Multitask-Multiprocessor Assignment Problem using the Empty Slots Method and Rate Monotonic Scheduling. *Real-Time Systems 13*, 167–199.

Presents a heuristic approach to the problem of assigning a set of preemptable resource-sharing and blockable real-time tasks to be executed in a set of heterogeneous processors communicated through an interprocessor network.

Sha, L. and S. Sathaye (1993). Distributed real-time system design: Theoretical concepts and applications. Technical Report CMU/SEI-93-TR-2 ESC-TR-93-179, Software Engineering Institute, Canergie Mellon University.

Describes the use of generalized rate monotonic scheduling theory for the design and analysis of a distributed real-time system.

Shirazi, B., A. Hurson, and K. Kavi (1995). *Scheduling and Load Balancing in Parallel and Distributed Systems.* New York: IEEE Computer Society Press.

Smith, R. (1980, December). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers C-29*(12), 1104–1113.

Pioneers the research in communication among cooperating distributed agents with the contract net protocol.

Stankovic, J. (1984, June). Simulation of Three Adaptive Decentralized Controlled Job Scheduling Algorithms. *Computer Network 8*(3), 199–217.

Stankovic, J. (1985a, February). An application of bayesian decision theory to decentralized control of job scheduling. *IEEE Transactions on Computers C-34*(2), 117–130.

The delay in transferring state information and tasks makes it impossible for a node scheduler to obtain the necessary data to take an optimal decision. The Bayesian decision based algorithm tries to reduce uncertainty through estimates based on information provided by the exchange of messages.

Stankovic, J. (1985b). Stability and Distributed Scheduling Algorithms. *IEEE Transactions on Software Engineering SE-11*(10), 1141–1152.

Lists two scheduling methods. The first is adaptive with dynamic reassignment, and is based on broadcast messages and stochastic learning automata. The second method uses bidding and one time-assignment in a real-time environment.

Stankovic, J., T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu (2001, December). Feedback Control Scheduling in Distributed Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 59–70.

Presents a framework (Distributed Feedback Control Real-Time Scheduling) for developing software control algorithms based on a theory of feedback control to a distributed open system.

Stankovic, J. and R. Mirchandaney (1986, December). Using stochastic learning automata for job scheduling in distributed processing systems. *J. of Parallel and Distributed Computing 3*, 527–552.

Stankovic, J. and K. Ramamritham (1988a). *Tutorial Hard Real-Time Systems* (J. A. Stankovic and K. Ramamritham ed.)., Paper: Scheduling Algorithms for Hard Real-Time Sytems - A Brief Survey, pp. 150–173. IEEE Computer Society Press.

Stankovic, J. and K. Ramamritham (1988b). *Tutorial Hard Real-Time Systems*, Paper: Evaluation of a Flexible Task Scheduling for Distributed Hard Real-Time Systems, pp. 273–286. IEEE Computer Society Press.

Stankovic, J. and K. Ramamritham (1991, May). The Spring Kernel: A New Paradigm for Real-Time Systems. *IEEE Software 8*(3), 62–72.

Implements and evaluates multiprocessor schedulers running on single, dedicated nodes of small scale parallel embedded systems using a static heuristic function, which may integrate timing, resource and preceding constraints.

Stankovic, J. and K. Ramamritham (1993). *Advances in Real-Time Systems*, Article: Misconceptions About Real-Time Computing, pp. 17–25. IEEE Computer Society Press.

Sun, J. and J. Liu (1996, May). Synchronization Protocols in Distributed Real-Time Systems. In *Proc. of the 16th International Conference on Distributed Computing Systems*, pp. 38–45.

Focuses on distributed real-time systems that contain independent, periodic tasks scheduled by fixed priority scheduling algorithms. Describes three synchronization protocols together with algorithms to analyze the schedulability of the system when these protocols are used.

Tia, T.-S. and J.-S. Liu (1995). Assigning Real-Time Tasks and Resources to Distributed Systems. *Special Issue of the International Journal of Mini and Microcomputers 17*(1), 18–25.

Presents a method for allocating periodic tasks where different tasks may have different deadlines. Graph based heuristics, which attempt to minimize interprocess communication and based on clustering and graph-bisection, are used for task assignment.

Tindell, K. and J. Clark (1994). Holistic Schedulability Analysis for Distributed Hard Real-Time Systems. *Microprocessing & Microprogramming 40*, 117–134.

Performs an early work on an event-driven model for scheduling of distributed systems. Analyses schedulability for distributed systems where tasks with arbitrary deadlines communicate by message passing and shared data areas. Uses periodic tasks for the first task in the transaction. Subsequent tasks are triggered as sporadic tasks when the preceding task has been completed.

Tokuda, H. and C. Mercer (1989, July). Arts: A Distributed Real-Time Kernel. In *Operating Systems Review*, Volume 23 of *3*, pp. 29–53. ACM Press.

Introduces a real-time object model and the integrated time-driven scheduling model to develop real-time computing systems in a distributed environment. Describes the Advanced Real-Time Technology (ARTS) kernel and the real-time toolset consisting of schedulability analyzer, *Scheduler 1-2-3*, and the real-time monitor/debugger.

Wellings, A., L. Beus-Dukic, and D. Powell (1998, December). Real-Time Scheduling in a Generic Fault-Tolerant Architecture. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain.

Presents a real-time scheduling for Generic Upgradable Architecture for Real-Time Dependable Systems (GUARDS). Uses an extended response-time analysis to predict the timing properties of replicated real-time transactions.

Chapter 6

Predictable Monoprocessor Scheduling

The goal of this chapter is to improve the results presented in the precedent chapter, especially those dealing with the unpredictability of heuristic mechanisms. To address the problem of *predictabe* aperiodic scheduling in the presence of hard periodic tasks is an important challenge for MaSHReC. This is achieved upon the monoprocessor platform of a production stage in MaSHReC by applying real-time server algorithms. A survey of real-time server mechanisms is addressed. An evaluation of dynamic server algorithms assists to find the appropriate scheduling technique to be applied to MaSHReC. Furthermore, an extension of a server-based algorithm to include changeover time costs is provided. The predictability of the system is proved through schedulability analysis techniques.

6.1 Predicatble Aperiodic Scheduling

Real-time manufacturing systems must be able to handle not only periodic tasks, but also aperiodic tasks. Periodic tasks are used to implement off-line pre-planned requests. While periodic tasks in real-time manufacturing systems have hard deadlines, aperiodic tasks may have soft, hard or no deadlines at all according to the priority of aperiodic requests.

When aperiodic tasks have hard deadlines, the goal of the system is to allow the production of aperiodic tasks without jeopardizing the schedulability of hard periodic tasks. A number of algorithms that solve this problem in fixed priority systems can be found in the literature. Figures 6.1 and 6.2[1] state, among other real-time uniprocessor algorithms, the relevant algorithms treating aperiodic tasks in fixed priority systems (Refer to the algorithms with "X", "S", or "F" in the column "AT" and with "F" in the column "Priority" of Figures 6.1 and 6.2). The same problem has been solved in the context of dynamic priority systems (Refer to the algorithms with the letter "D" in the column "Priority" of Figures 6.1 and 6.2 in order to point to dynamic priority algorithms). A common approach to solve the problem of aperiodic scheduling in real-time scheduling theory is the use of *servers*. A server is a special purpose process in the system, whose *capacity* is used to serve aperiodic requests. A server is usually scheduled by a specific algorithm designed in such a way that periodic tasks do not miss their deadline and at the same time, the machine is allocated to the server as soon as possible, in order to maintain the deadline of aperiodic tasks.

[1]The references in the first column of Figures 6.1 and 6.2 are composed as follows: when the publication is written by a single author, the reference is composed of the first 3 letters of his family name followed by the year of publication. When the publication is written by more than one author, the reference is the composed of the first letter of the family names of the authors followed by the year of publication.
The abbreviations in the first line of the table refer to the following terms: sched.: scheduling, alg.: algorithm, PT: Periodic Task, ST: Sporadic Task, AT: Aperiodic Task, Com.: Communication, CST: Context Switch Time, Int.: Integer, par.:parameters, per.: periods, SM: Shared Memory, DM: Distributed Memory, Ind.: Independent, Prec. Cons.: Precedence Constraints, JC: Jitter Control, RM: Resource Management.
The abbreviations in the rest of the table refer to the following terms: D: distributed, U: uniprocessor, M: multiprocessor, PI: parallel identical, H: hard, S: soft, F:firm, Dy: Dynamic, Fi: fixed, DS: deferrable server, SS: sporadic server, PE: priority exchange, TBS: total bandwidth server, ETBS: extended total bandwidth server.

6.2 State-of-the-Art

A number of essential features characterizes the scheduling technique presented in this chapter. These are the uniform platform upon which the server algorithm is executed, predictabilty of the control technique, task preemption, on-line execution, handling of periodic and aperiodic tasks, dynamic priority of tasks, and context switch time considerations. Figures 6.1 and 6.2 provide a broad overview of relevant uniprocessor real-time scheduling techniques. As shown in Figures 6.1 and 6.2, selected features attempted for in this chapter are considered in the literature. However, a combination of all features relevant for this chapter is not treated.

Furthermore, an important remaining problem, that distinguishes real-time operating systems from production control systems, is to analyze the effects of *context switch* operations on the real-time schedule and schedulability analysis. Almost all the literature investigated considered no context switch time overhead. However, a technique presented by Burns et al. (1995) is capable of incorporating a non-negligible context switch cost into the schedulability analysis. Unfortunately, these schedulability analyses are based on the assumption that the cost for a context switch is constant and independent of the underlying thread-invocation pattern (Jonsson 1998).

6.3 Server Algorithms Evaluation

This chapter addresses the problem of aperiodic scheduling of manufacturing systems under hard real-time constraints including the novel feature of integrating switch time overheads. As already mentioned, the use of servers is a common approach in real-time systems theory to solve the problem of aperiodic task scheduling. The main idea is to use a *server*, which is a periodic task whose purpose is to service aperiodic requests as soon as possible. A short description and evaluation of different server scheduling techniques considered in the literature is given.

A Polling server (Shin and Chang 1995) is a periodic task with a fixed priority level (usually the highest) and an execution capacity. The capacity of the server is calculated off-line and is normally set to the maximum possible, such that the task set, including server, is schedulable. At run-time, the Polling server is released periodically and its capacity is used to service aperiodic real-time tasks. Once this capacity has been exhausted, execution is suspended until it can be replenished at the server's next release. The Polling server will usually significantly improve the response times of soft

References	Year	Plat-form	Heuristic sched.	Predi-ctable sched.	Serv. alg.	Off-line	On-line	PT	ST	AT	Com. time	Pre-emption	No preem-ption	CST = 0	CST > 0	Int. sched. par.	Int. per.	Prio-rity	SM	DM	Ind. Tasks	Prec. Cons.	JC	RM	Over-load
Ser72	1972	U						X																	
LL73	1973	U		X		X		X				X		X				Fi							
LM80	1980	U		X				X						X											
LM81	1981	PI, PU		X				X				X		X			X				X				
LLL82	1982	U		X				X						X											
LW82	1982	U		X								X		X				Fi			X				
LLN87	1987	U		X				X						X											
BSR88	1988	U	X							X															X
Str88	1988	U		X	DS		X	X						X											
SLCG89	1989	U, MI	X					X				X									X				
BMR90	1990	U		X					X			X													
Spr90, SSL89	1989	U		X	SS		X	X	X	S		X		X				Dy			X				
TM89	1989	D,U		X				H		S	X							Dy							X
SSL89	1989	U		X	PE		X	X		S		X		X				Dy			X				
BHR90	1990	U, PI						X				X		X											
ABRW91	1991	U						X	X																
YS91	1991	U				X		X				X						Fi							
SK91	1991	U					X	X										Dy			X				
KS91	1991	U	X				X	X		F		X		X											
LR92	1992	U	X			X		H		S								Fi							
KS92	1992	U	X				X					X		X											X
DTB93	1993	U, M				X	X	X	X			X		X				Fi							
SBS95	1995	U			TBS		X	H		F		X		X				Dy			X				X
Str95	1995	U		X			X																		
BSS95	1995	U					X											Dy							X
DW95	1995	U						H	H																
KS95	1995	U						S										Fi							
GMM95	1995	U	X			X	X														X	X			
Bar95	1995	U				X		X										Dy							
OF96	1996	U	X					X	X									Fi				X			
FL97	1997	U	X									X													X

Figure 6.1: Real-time uniprocessor algorithms evaluation

References	Year	Plat-form	Heuristic sched.	Predi-ctable sched.	Serv. alg.	Off-line	On-line	PT	ST	AT	Com. time	Pre-emption	No preem-ption	CST =0	CST >0	Int. sched. par.	Int. per.	Prio-rity	SM	DM	Ind. Tasks	Prec. Cons.	JC	RM	Over-load
RCG97	1997	U		X			X	X		X		X						X			X				
AS97	1997	U		X			X	H																	
MC97	1997	U		X				X				X						Fi							
SGL97	1997	U		X				X				X						X			X	X			
IF98	1998	U				X	X		X																
HPS98	1998	U	X				X	X	X			X		X											
AB98	1998	U		X		X		X				X						Fi							
LMKLL99	1999	U	X			X						X						Fi							
CLB99	1999	U		X	ETBS		X	X		S		X		X											
SLST99	1999	U					X																		
LSTS99	1999	U					X																		
BB99	1999	U, D		X			X	X					X	X								X			
AMMM99	1999	U	X					F													X				
SC00	2000	U	X				X				X														
BB01	2001	U					X	X		F															
BC01	2001	U		X			X	F																	
MFFR01	2001	U		X			X	X															X		
AMMM01	2001	U	X			X	X	H																	
LLA01	2001	U		X																	X			X	
CS01	2001	U			X		X	H		S		X										X			
El-Ke01b	2001	U		X	X		X	X		F		X			X						X				
OSSF01	2001	U					X	X				X						X			X				
PA01	2001	U		X			X	X										X			X				

Figure 6.2: Real-time uniprocessor algorithms evaluation (continued)

tasks over background processing. However, if the ready tasks exceed the capacity of the server, then some of them will have to wait until its next release, leading to potentially long response times. Conversely, no aperiodic tasks may be ready when the server is released , wasting its high priority capacity.

This latter drawback is avoided by the Priority Exchange server (Sprunt et al. 1989), Deferrable server (Strosnider 1988), Sporadic server (Sprunt 1990) and Total Bandwith server (Spuri and Buttazzo 1996) algorithms. These are all based on similar principles to the Polling server. However, they are able to preserve capacity if no tasks are pending when they are released. A *consumption rule* gives the conditions under which the execution budget is preserved and consumed and a *replenishment rule* indicates when and by how much the budget is replenished. Due to these properties, they are named "bandwidth preserving algorithms". The three algorithms differ in the ways in which the capacity of the server is preserved and replenished and in the schedulability analysis needed to determine their maximum capacity. Like a Poller server, the execution budget of a Deferrable server is replenished periodically. However, unlike a Poller server, when a Deferrable server finds no aperiodic task ready for execution, it preserves its budget. The Dynamic Priority Exchange server trades its runtime with runtime of lower priority periodic tasks in case there are no aperiodic requests pending. In this way, the server time is only exchanged by periodic tasks but never wasted (unless there are idle times). The Sporadic sever differs from Deferrable server and Priority Exhange server in the way it replenishes its capacity. Whereas Deferrable server and Priority Exhange server replenish their capacities to its full value at the beginning of each server period, Sporadic server replenishes its capacity only after it has been consumed by aperiodic task execution.

In general, all three servers offer improved responsiveness over the polling approach. However, there are still disadvantages with these more complex server algorithms. They are unable to make use of slack time which may be present due to the often favourable phasing of periodic tasks (i.e. not worst case). Furthermore, they tend to degrade to providing essentially the same performance as the Polling server at high loads. The Deferrable and Sporadic servers are also unable to reclaim spare capacity gained, when for example, hard tasks require less than their worst case execution time. This spare capacity can however be reclaimed by the Extended Priority Exchange Algorithm (Sprunt et al. 1988).

A more simple bandwidth preserving algorithm to schedule aperiodic tasks is given by Total Bandwidth servers. The main idea of a Total Band-

🙂 Excellent 😐 Good ☹ Poor

	Performance	Computational complexity	Memory requirement	Implementation complexity
BKG	☹	🙂	🙂	🙂
DPE	😐	😐	😐	😐
DDS	😐	😐	😐	😐
TB	😐	🙂	🙂	🙂
EDL	🙂	☹	☹	☹
IPE	🙂	😐	☹	☹
TB*	🙂	☹	🙂	😐

Figure 6.3: Evaluation summary of dynamic priority server

width sever is to assign a possible earlier deadline to each aperiodic request such that the overall processor utilization of the aperiodic load is less or equal than a server utilization value.

A qualitative evaluation among the servers is presented in Figure 6.3 (Buttazzo 1997). This comparison shows that the Total Bandwidth Server algorithm and the nearly optimal Improved Priority Exchange algorithm have a good performance. However, compared to the Improved Priority Exchange algorithm, the Total Bandwidth server scheme does not require large memory capacity and it benefits from a low implementation and computational complexity. Less switching time overheads and a better maintainability of the system are achieved through a low computational complexity and a low implementation complexity respectively. Since manufacturing systems are concerned with a significant changeover time cost and should provide a high degree of maintainability (refer to hypothesis 5 page 7), the Total Bandwidth Server mechanism is adopted (called in Section 6.5).

6.4 Assumptions and Terminology

The algorithm to be defined in the following sections is based on the following assumptions and terminology.

- $\tau = \{\tau_{i,j} \mid i,j \in \mathbb{N}\}$ A set of periodic tasks with <u>hard</u> deadlines. i denotes the number of the part, and j the group to which it is affiliated according to its changeovertime.
- $J = \{J_{i,j} \mid i,j \in \mathbb{N}\}$ A set of active aperiodic tasks ordered by increasing deadline, $J_{1,j}$ being the task with the shortest absolute deadline.
- $O = \{O_{j,k} \mid j,k \in \mathbb{N}\}$ The context switch time or changeover time caused by the arrival of a part from type j at the machine k. Changeover time is derived by a static analysis on the machine.
- $\overline{O} = \{\overline{O}_{j,k} \mid j,k \in \mathbb{N}\}$ denotes the actual switch time or changeover time caused by the arrival of the part from type j at the machine k.
- The arrival time of each aperiodic task is <u>unknown</u>.
- The worst case production time of each aperiodic task is considered to be known at its arrival time.
- A periodic task $\tau_{i,j}$ has a constant period T_i and a constant worst case production time C_i. The worst case production time of a task is derived by a static analysis on the machine.
- All aperiodic tasks have firm deadlines and can be rejected.
- r_i denotes the arrival time of task $J_{i,j}$.
- $\overline{r}_i$ denotes the corrected release time of task $J_{i,j}$.
- C_i denotes the maximum production time of task $J_{i,j}$, i.e. the worst case production time needed for the machine to execute task $J_{i,j}$ without interruption.
- $\overline{C}_i$ denotes the actual production time of task $J_{i,j}$.
- d_i denotes the absolute deadline of task $J_{i,j}$, i.e. the time before which the task should complete its production in order to be useful to the system.

- m_i denotes the deadline tolerance of task $J_{i,j}$, i.e. the maximum time that task $J_{i,j}$ may execute after its deadline, and still produce a valid result.
- v_i denotes the task value, i.e. the relative importance of task $J_{i,j}$ with respect to the other tasks in the set.
- f_i denotes the finishing time of task $J_{i,j}$, i.e. the time at which task $J_{i,j}$ completes its execution and leaves the system.
- E_i denotes the exceeding time, i.e. the possible lateness of task $J_{i,j}$ in case of overload.

6.5 The Total Bandwidth Server Algorithm

In this section, the Total Bandwidth Server algorithm with its extensions are briefly recalled.

6.5.1 The fisrt version of the Total Bandwidth Server

The main idea of the Total Bandwidth Server is to assign, whenever possible, the total bandwidth (in term of machine production time) of the server, each time an aperiodic task enters the system. This is done by assigning a deadline

$$d_k = max(r_k, d_{k-1}) + \frac{C_k}{U_s}$$

(C_k is the maximum execution time of the request and U_s is the server utilization factor or bandwidth)
to the request and to schedule it according to the Earliest Deadline First algorithm together with the periodic tasks in the system. Spuri and Buttazzo (1994) investigated the problem of the joint hard periodic and soft aperiodic scheduling under dynamic priority systems. The Total Bandwidth server algorithm showed the best performance/cost ratio among the several servers described.

6.5.2 Extension of the Total Bandwidth Server: the Robust Total Bandwidth algorithm

Spuri et al. (1995) extended the original formulation of the Total Bandwidth Server algorithm: the robust Total Bandwidth algorithm. The extended algorithm focused on the problem of the joint hard periodic and *firm* aperiodic scheduling under dynamic priority systems. It extends the Total Bandwidth Server to include preemption and a technique, including a rejection and a reclaiming strategies, for the addition of robustness in case of transient overloads. This mechanism has been proved effective by extensive simulations (Spuri et al. 1995).

6.6 Adapting the TBS algorithm to the Monoprocessor Scheduling of a Production Stage

While most of real-time systems scheduling theory assumes that context switch time overheads are negligible, production control systems, which have a higher changeover time cost, are not allowed to deal with this assumption. The problem is more complex in *real-time* production control systems where disregarding changeover time overheads may lead to undesired results, such as the *domino effect* (Spuri, Buttazzo, and Sensini 1995), in which a missed deadline causes a series of subsequent deadlines to be also missed, and in hard real-time manufacturing systems to disruptive results. This section adapts the Total Bandwidth Server algorithm to the planning and control of manufacturing systems in retaining the advantages of the previous Total Bandwith Server algorithms recalled in the previous section while overcoming their limitations. Note that the following section treats the problem of resource reclaiming simultaneously.

6.7 Adding Changeover Time Costs to TBS with Resource Reclaiming

Since deadlines of aperiodic tasks are assigned based on their *estimated* maximum execution time, this may be a drawback for the TB server when the value is overestimated. Spuri et al. (1995) proposed a reclaiming technique, to correct the assigned deadlines. Whenever a request completes earlier, its actual execution time is used to compute the deadline that could have been assigned to it if its execution time had been known in advance. This value is then used to compute the deadline for the next request.

Integrating changeover time costs affects also the TB server, since deadlines are also based on this value. There are two methods to solve the problem of integrating changeover time costs:

- One is to assign a constant maximum changeover time and to include it in the estimated execution time. This would result in a lot of wasted machine time, specifically when the arriving aperiodic part has the same type as the precedent one, and subsequently the machine does not execute any switching time.

- The second alternative is to consider changeover time costs in the computation of the deadlines of aperiodic tasks. More formally the i-th task to be executed receives a deadline equal to:

  $$d'_i = \overline{r}_i + \frac{C_{i,j} + 2 * O_{i,j}}{U_s}$$

  where C_i is the maximum execution time of the task and U_s is the server utilization factor. The reader should refer to the example in section 4.6 page 50 for an explanation of $2 * O_{i,j}$. $\overline{r}_i$ is the "corrected" release time of the task and is computed as:

  $$\overline{r}_i = max(r_i, \overline{d}_{i-1}, f_{i-1})$$

  At task completion the corrected deadline is computed as:

  $$\overline{d}_i = \overline{r}_i + \frac{\overline{C}_{i,j} + 2 * O_{i,j}}{U_s}$$

Furthermore, the corrected deadline $\overline{d}$ may be optimized by considering the "corrected" changeover time cost, when the executed changeover is shorter than the assumed maximum changeover time. At task completion the corrected deadline is computed as

$$\overline{d}_i = \overline{r}_i + \frac{\overline{C}_{i,j} + 2 * \overline{O}_{i,j}}{U_s}$$

where $\overline{O}$ is the actual changeover time. Being $\overline{C}_{i,j} < C_{i,j}$ and $2\overline{O}_{i,j} < O_{i,j}$, we have $\overline{d}_i < d'_i$, that is we try to reclaim the unused computation time and the unused changeover time by assigning a shorter deadline to the next request.

6.8 Schedulability Analysis

To prove the schedulability of the TB server with this new formulation, a slight extension of the proof of Buttazzo (1997) to include changeover time is proceeded. Buttazzo (1997) first showed that the actual aperiodic machine utilization cannot exceed U_s, and then he shows that the overall utilization can be up to 100%.

Lemma 1 *In each interval of time* $[t_1,t_2]$, *if* $\overline{C}_{ape}$ *is the total execution time actually demanded by aperiodic requests arrived at* t_1 *or later and served with deadlines less than or equal to* t_2*, and* $\overline{O}_{ape}$ *the total changeover time due to the arrival of these aperiodic requests in the interval of time* $[t_1,t_2]$ *then*

$$\overline{C}_{ape} + \overline{O}_{ape} \leq (t_2 - t_1)U_s \tag{6.1}$$

Proof. By definition

$$\overline{C}_{ape} = \sum_{t_1 \leq r_k, d'_k \leq t_2} \overline{C}_k \tag{6.2}$$

$$\overline{O}_{ape} = \sum_{t_1 \leq r_k, d'_k \leq t_2} \overline{O}_k \tag{6.3}$$

The index k indicates the order of execution. Thus, there must be two indexes k_1 and $k2$ such that

$$\sum_{t_1 \leq r_k, d'k \leq t_2} \overline{C}_k \leq \sum_{k=k_1}^{k_2} \overline{C}_k \tag{6.4}$$

and

$$\sum_{t_1 \leq r_k, d'k \leq t_2} \overline{O}_k \leq \sum_{k=k_1}^{k_2} \overline{O}_k \tag{6.5}$$

By adding the inequations (6.4) and (6.5) we obtain

$$\sum_{t_1 \leq r_k, d'k \leq t_2} \overline{C}_k + \sum_{t_1 \leq r_k, d'k \leq t_2} \overline{O}_k \leq \sum_{k=k_1}^{k_2} \overline{C}_k + \sum_{k=k_1}^{k_2} \overline{O}_k \tag{6.6}$$

From the inequations (6.2) and (6.3) it follows that

$$\overline{C}_{ape} + \overline{O}_{ape} \leq \sum_{k=k_1}^{k_2} \overline{C}_k + \sum_{k=k_1}^{k_2} \overline{O}_k \tag{6.7}$$

$$\overline{C}_{ape} + \overline{O}_{ape} \leq \sum_{k=k_1}^{k_2} (\overline{C}_k + \overline{O}_k) \tag{6.8}$$

$$\overline{C}_{ape} + \overline{O}_{ape} \leq \sum_{k=k_1}^{k_2} (\overline{d}_k - \overline{r}_k)U_s \tag{6.9}$$

$$\overline{C}_{ape} + \overline{O}_{ape} \leq U_s \sum_{k=k_1}^{k_2} (\overline{d}_k - \overline{r}_k) \tag{6.10}$$

Since $\overline{d}_{k-1} \leq \overline{r}_k$, the inequation (6.10) becomes

$$\overline{C}_{ape} + \overline{O}_{ape} \leq U_s(\overline{d}_{k_2} - \overline{r}_{k_1}) \tag{6.11}$$

Finally, being $\overline{d}_{k_2} \leq d'_{k_2} \leq t_2$ and $\overline{r}_{k_1} \geq r_{k_1} \geq t_1$, we have

$$\overline{C}_{ape} + \overline{O}_{ape} \leq U_s(t_2 - t_1) \tag{6.12}$$

We can now prove the following result.

Theorem 1 *Given a set of n periodic tasks with processor utilization U_p and a TB server with processor utilization U_s, the whole set is feasibly scheduled if and only if*

$$U_p + U_s \leq 1 \tag{6.13}$$

Proof. "If". Suppose there is an overflow at time t. The overflow must be preceeded by a period of continuous utilization of the processor. Furthermore, from a certain point t' on, only instances of tasks (periodic or aperiodic) ready at t' or later and having deadlines less than or equal to t are run. Let $\overline{C}$ be the total execution time and $\overline{O}$ the total changeover time demanded by these instances. Since there is an overflow at time t, we must have

$$t - t' \leq \overline{C} + \overline{O} \tag{6.14}$$

We also know that

$$\overline{C} + \overline{O} \leq \sum_{i=1}^{n} \left\lfloor \frac{t-t'}{T_i} \right\rfloor C_i + \overline{C}_{ape} + \overline{O}_{ape} \tag{6.15}$$

$$\leq \sum_{i=1}^{n} \frac{t-t'}{T_i} C_i + (t-t')U_s \tag{6.16}$$

$$\leq (t-t')(U_p + U_s) \tag{6.17}$$

The following contradiction is derived

$$U_p + U_s \geq 1 \tag{6.18}$$

"Only if". If an aperiodic request enters the system periodically, say each $T_s \geq 0$ units of time, and the execution time $C_s + O_s = \overline{C}_s + \overline{C}_s = T_s U_s$, the server behaves exactly as a periodic task with period T_s and execution time C_s. Having a processor utilization $U = U_p + U_s$ and using the theorem of Liu and Layland (1973) , we can conclude that $U_p + U_s \leq 1$.

6.9 Contributions of the Chapter

The scientific contributions of this chapter can be summarized as follows:

- *A Classification and Review of Real-Time Uniprocessor Scheduling Literature*
  A relevant and structured overview of real-time Uniprocessor scheduling theory is presented. The different algorithms are structured according to the problem they solve and to the scheduling parameters which are relevant for the real-time scheduling of manufacturing systems.

- *Predictable Scheduling upon a Monoprocessor Production Stage in MaSHReC*
  A predictable scheduling, based on the TBS algorithm, is adapted upon a monoprocessor production shift in MaSHReC. It is extended to include the novel feature of accounting for changeover time overheads.

6.10 Hypotheses Evaluation

The scheduling problem treated in this chapter clearly validates the hypotheses related with the production of aperiodic tasks, system timeliness and system predictability. The corresponding hypotheses are 3, 5.1 and 5.2 respectively. The reader should refer to Section 1.5 page 7 of Chapter 1 for a detailed definition of the cited hypotheses.

Bibliography

Abdelzaher, T. and K. Shin (1997, September). Comment on "A Pre-Run-Time Scheduling Algorithm for Hard Real-Time Systems". *IEEE Transactions on Software Engineering 23*(9), 599–600.
Shows that the branch-and-bound implicit enumeration algorithm does not always succeed in finding a feasible solution, and describes the reason why this algorithm might fail.

Atlas, A. and A. Bestravos (1998, December). Statistical Rate Monotonic Scheduling. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain.
Statistical Rate Monotonic Scheduling is a generalization of the classical Rate Monotonic Scheduling results of Liu and Layland. This algorithm allows the scheduling of periodic tasks with variable resource requirement to achieve a requested statistical Quality of Service guarantee. It Yield controllable and predictable Quality of Service unrelated to the period of a given task.

Audsley, N., A. Burns, M. Richardson, and A. Wellings (1991). Hard Real-Time Scheduling: The Deadline Monotonic Approach. In *Eighth IEEE Workshop on Real-Time Operating Systems and Software*, pp. 133–137.
Investigates schedulability tests for sets of periodic processes whose deadlines are permitted to be less than their period. Such a relaxation enables sporadic processes to be directly incorporated without alteration to the process models.

Aydin, H., R. Melhem, D. Mossé, and P. Mejia-Alvarez (2001, December). Dynamic and Agressive Scheduling Techniques for Power-Aware Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 95–105.
Addresses power-aware scheduling of periodic hard real-time tasks using dynamic voltage scaling.

Aydin, H., R. Melhem, S. Mossé, and P. Mejia-Alvarez (1999, December). Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona. IEEE Computer Society Press.

Addresses the periodic reward-based scheduling problem in the context of uniprocessor systems. Focuses on linear and concave reward functions, which adequately represent realistic applications such image and speech processing, time-dependent planning and multimedia presentations.

Baruah, S. (1995, December). Fairness in Periodic Real-Time Scheduling. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 200–209.

Describes a quantitative measure of temporal fairness "pfairness" in periodic real-time scheduling. Presents the Weight-Monotonic scheduling algorithm, a static priority scheduling algorithm for generating "pfair" schedules.

Baruah, S., R. Howell, and L. Rosier (1990). Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor. *Real-Time Systems 2*, 301–324.

investigates preemptive scheduling algorithms of periodic real-time task systems on one processor.

Baruah, S., A. Mok, and L. Rosier (1990, December). Preemptively Scheduling Hard Real-Time Sporadics Tasks on One Processor. In *Proc. of the Real-Time Systems Symposium*, pp. 182–190. IEEE Computer Society Press.

Gives a necessary and sufficient conditions for a sporadic task system to be preemptively schedulable on one processor.

Bate, I. and A. Burns (1999). A Framework for Scheduling in Safety-Critical Embedded Control Systems. In *Proc. of the 6th International Conference on Real-Time Computing Systems and Apllications.*

Presents a computational model that supports the reuse of legacy systems. Develops timing analysis that features low pessimism and low computational complexity.

Bernat, G. and C. Cayssials (2001, December). Guaranteed On-Line Weakly-Hard Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 25–35.

Presents an on-line scheduling framework called Bi-Modal Scheduler for weakly hard real-time systems. In a normal mode, tasks can be scheduled with a generic scheduler. Weakly hard constraints are guaranteed to be satisfied by switching to a panic mode for which schedulability tests exist.

Biyabani, S., J. Stankovic, and K. Ramamritham (1988, December). The Integration of Deadline and Criticalness in Hard Real-Time Scheduling. In *Proc. of the 9th Real-Time Systems Symposium*, Los Alamitos, California, pp. 152–169. CS Press.

Presents two heuristric approaches for distributed hard real-time computer systems. The algorithms explicitely account for both deadlines and criticalness of tasks when making scheduling decisions.

Burns, A. and G. Bernard (2001). Jorvik: A framework for effective scheduling. Technical Report YCS-334, Department of Computer Science, University of York.

Presents a collection of mechanisms that together form a framework for the support of flexible scheduling with mix hard and soft tasks, periodic and aperiodic load and intertask relationships.

Burns, A., K. Tindell, and A. Wellings (1995, May). Effective Analysis for Engineering Real-Time Fixed Priority Schedulers. In *IEEE Transactions on Software Engineering*, Volume 21 of *5*, pp. 475–480.

Presents an analysis that enables the cost of the scheduler (clock overheads, queue manipulations and release delays) to be factored into the standard equations of calculating worst-case response times in hard real-time systems.

Buttazzo, G. (1997). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer academic Publishers.

Introduces basic concepts for real-time computing, with emphasis on predictable scheduling algorithms. Handles periodic and aperiodic task scheduling, tasks with precedence constraints, access protocols to shared resources, schedulability analysis, and handling overload conditions.

Buttazzo, G., M. Spuri, and F. sensini (1995, December). Value vs Deadline Scheduling in Overload Conditions. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 90–99.

Presents a comparative study among scheduling algorithms which use different priority Assignments and different guarantee mechanism during overload conditions.

Caccamo, M., G. Lipari, and G. Buttazzo (1999, December). Sharing Resources among Periodic and Aperiodic Tasks with Dynamic Deadlines. In *Proc. of the IEEE Real-Time Systems Symposium, Phoenix, Arizona*, pp. 284–293.

Addresses the problem of scheduling hybrid real-time task sets consisting of hard periodic and soft aperiodic tasks that may share resources in exclusive mode in a dynamic environment. Considers that resources are accessed through the Stack Resource Policy and aperiodic tasks are serviced by the tunable Total Bandwidth Server.

Caccamo, M. and L. Sha (2001, December). Aperiodic Servers with Resource Constraints. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 161–170.
Addresses the problem of integrating resource constraints in the execution of hybrid task sets including hard period and soft aperiodic task sets using a capacity based server.

Davis, R., K. Tindell, and A. Burns (1993). Scheduling Slack Time in Fixed Priority Pre-emptive Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, pp. 222–231.
Addresses the problem of jointly scheduling tasks with both hard and soft time constraints. Determines the maximum processing time which may be stolen from hard deadline periodic or sporadic tasks, without jeopardising their timing constraints.

Davis, R. and A. Wellings (1995, December). Dual Priority Scheduling. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 100–109.
Presents a strategy for scheduling tasks with soft deadlines in real-time systems containing periodic, sporadic and adaptive tasks with hard deadlines.

El-Kebbe, D. A. (2001, March). A UML Model for the MaSHReC Architecture. In *Proc. of the International Congress on Information Science Innovations in Intelligent Automated Manufacturing*, Dubai, United Arab Emirates.
Investigates current holonic manufacturing architectures. The MaSHReC model is designed using the Unified Modeling Language (UML). UML templates are provided to allow the design of the system structure, components, relations, data, functions and interactions.

Feng, W. and J. W.-S. Liu (1997, February). Algorithms for Scheduling Real-Time Tasks with Input Error and End-to-End Deadlines. *Transactions on Software Engineering 23*(2), 93–106.
Describes algorithms for scheduling preemptive, imprecise, composite tasks in a real-time system. Extends the imprecise computation technique to account for input error and end-to-end timing constraints. Develops five algorithms to minimize the output error of each composite task.

Ghosh, S., R. Melhem, and D. Mousse (1995, December). Enhancing Real-Time Schedules to Tolerate Transient Faults. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 120–129.

Presents a scheme that can be used to guarantee that the execution of real-time tasks can tolerate transient und intermittent faults. The scheme is based on reserving sufficient slack in a schedule such that a task can be re-executed before its deadline without compromising the guaranteed given to other tasks.

Hong, I., M. Potkonjak, and M. Srivastava (1998, November). On-Line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor. In *Proc. of the International Conference on Computer-Aided Design.*
Considers the problem of scheduling hard periodic and firm sporadic tasks on variable voltage processor to optimize power consumption.

Isovic, D. and G. Fohler (1998). Handling Sporadic Tasks in Off-line Scheduled Distributed Real-Time Systems. In *Proc. of the 11th Euromicro Conference on Real-Time Systems.* IEEE.
Presents an algorithm to handle event-triggered sporadic tasks in the context of time-triggered, off-line scheduled systems. Provides off-line schedulability test for sporadic tasks.

Jonsson, J. (1998). Compile-Time Scheduling of Real-Time Threads on Multi-Level-Context Architectures. In *Proc. of the Seventh Swedish Workshop on Computer System Architecture.*
Addresses the problem of how to schedule periodic, real-time threads on a class of architectures referred to as multi-level-context (MLC) architectures such as real-time operating system architectures.

Koren, G. and D. Sasha (1992, December). D-over: An Optimal On-Line Scheduling Algorithm for Overloaded Real-Time Systems. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona, pp. 290–299. IEEE.
Presents an optimal on-line algorithm for overloaded systems. Optimal means that the algorithm gives the best competitive factor possible relative to an off-line scheduler.

Koren, G. and D. Shasha (1991). An optimal scheduling algorithm with a competitive factor for real-time systems. Technical Report TR572, Department of Computer Science, New York University.
Presents an algorithm for a possibly overloaded system which behaves like the EDF algorithm when the system is underloaded, and obtains at least a quarter of the maximum value that an optimal clairvoyant algorithm could obtain even when the system is overloaded.

Koren, G. and D. Shasha (1995, December). Skip-Over: Algorithms and Complexity for Overloaded Systems That Allow Skips. In *Proc. of the*

Real-Time Systems Symposium, Pisa, Italy, pp. 110–117.
Investigates the problem of uniprocessor scheduling of occasionally skippable periodic tasks (tasks with acceptable deadline missing).

Lamastra, G., G. Lipari, and L. Abeni (2001, December). A Bandwidth Inheritance Algorithm for Real-Time Task Synchronization in Open Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 151–160.
Presents the BandWidth Inheritance (BWI) scheduling strategy that extends the bandwidth reservation approach to open systems where tasks can interact through shared resources. Off-line sched anal. not possible. Arrival Time unknown.

Lawler, E. and C. Martel (1981, Februar). A Note on Preemptive Scheduling of Periodic Real-Time Tasks. *Information Processing Letters 12*(1), 9–12.
Considers a problem in which periodic tasks are to to be preemptively scheduled on a system of parallel processors. Shows that there exists a feasible schedule if and only if there exists a feasible schedule which is cyclic with a period equal in length to the least common multiple of the periods of the individual tasks.

Lee, S., S. Min, C. Kim, C.-G. Lee, and M. Lee (1999). Cache-Conscious Limited Preemptive Scheduling. *Real-Time Systems 17*, 257–282.
Proposes a scheduling scheme, called Limited Preemptive Scheduling, to address the problem of inter-task cache interference due to preemptions in multi-tasking real-time systems.

Lehoczky, J. and S. Ramos-Thuel (1992, December). An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, Phoenix, Arizona, pp. 110–123.
Presents an approach (Slack Stealing) for servicing aperiodic requests within the context of a hard real-time system by making any spare processing time available as soon as possible. A means of determining the maximum amount of slack which may be stolen, without jeopardising the hard timing constraints, is thus the key to the operation of the algorithm.

Leung, J. and M. Merril (1980). A Note on Preemptive Scheduling of Periodic Real-Time Tasks. *Information Processing Letters 11*(3), 115–118.

Leung, J. and J. Whitehead (1982, December). On the complexity of fixed priority scheduling of periodic, real-time tasks. In *Performance Evaluation*, Volume 2 of *4*, Netherlands, pp. 237–250.

Formulates an alternative priority assignment policy, where task deadlines can be less than the period of a task. Provides simple analysis to determine the schedulability of such tasks.

Liu, C. and J. Layland (1973, January). Scheduling algorithms for multiprogramming in a hard real-time environment. *J. of the ACM 20*(1), 46–61.

Describes the different aspects of scheduling of periodic systems. Presents the first proof of the optimality of rate-monotonic and Earliest Deadline First scheduling for periodic systems.

Liu, C., J. Liu, and A. Liestman (1982). Scheduling with Slack-Time. In *Acta Informatica*, Volume 17, pp. 31–41.

Liu, J., K.-J. Lin, and S. Natarajan (1987, December). Scheduling Real-Time Periodic Jobs Using Imprecise Results. In *Proc. of the 8th Real-Time Systems Symposium*, San Jose, California, pp. 252–260. IEEE.

Outlines an approach to design general purpose real-time computer systems that can skip non-critical portions of scheduled jobs in order to avoid missed deadline during system overloads.

Lu, C., J. Stankovic, G. Tao, and S. Son (1999, June). Design and Evaluation of a Feedback Control EDF Scheduling Algorithm. In *Proc. of the Real-Time Systems Symposium.*

Presents and evaluates the feedback control real-time scheduling.

Marti, P., J. Fuertes, F. Fohler, and K. Ramamritham (2001, December). Jitter Compensation for Real-Time Control Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 39–48.

Proposes an approach for real-time scheduling of control systems by compensating for sampling jitter and sampling-actuation delay through the adjustment of controller parameters.

Mok, A. and D. Chen (1997, October). A Multiframe Model for Real-Time Tasks. *IEEE Transactions on Software Engineering 23*(10), 635–645.

Presents a multiframe real-time task model which allows the execution time of a task to vary from one instance to another by specifying the execution time of a task in terms of a sequence of numbers.

Mok, A. and W. Wang (2001, December). Window-Constrained Real-Time Periodic Task Scheduling. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 15–24.

Shows that the Dynamic Window-Constrained Scheduling fails for arbitrarily low aggregate utilization rates of the packet streams. Defines the notion of Pfairness in relation to the Window-Contrained Scheduling model. Defines an EDF for the Window-Contrained scheduling problem.

Oliveira, R. and J. Fraga (1996). Scheduling Imprecise Computation Tasks with Intra-Task / Inter-Task Dependence. In *Proc. of the 21st IFAC/IFIP Workshop on Real-Time Programming*, pp. 51–56.

Presents two heuristics to be used as admission policy when the system is made of imprecise tasks. The objective of the admission policy is to maximize system utility through the selection of optional parts for execution. These heuristics are supposed to be used combined with off-line schedulability tests and on-line acceptance tests already described in the literature.

Orozco, J., R. Santos, J. Santos, and E. Ferro (2001, December). Hybrid Rate-Monotonic/Reward-Based Scheduling of Real-Time Embedded Sytems. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Studies the on-line scheduling of periodic, independent, preemptable real-time sets of tasks consisting of a mandatory and an optional part. Presents a scheduling method allowing a rearrangement of slack time based on the detection of singular instants during the processing.

Padreiras, P. and L. Almeida (2001, December). A Practical Approach to EDF Scheduling on Controller Area Network. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Compares Earliest Deadline First algorithm with the rate monotonic approach on a FTT CAN protocol. Concludes that using EDF, a higher bus utilization and a reduced jitter and delay time for messages with long periods are achieved.

Ripoll, I., A. Crespo, and A. Garcia-Fornes (1997, June). An Optimal Algorithm for Scheduling Soft Aperiodic Tasks in Dynamic-Priority Preemptive Systems. *IEEE Transactions on Software Engineering 23*(6), 388–400.

Presents the theoretical foundations for the EDF Exact Slack Stealer algorithm which provides a solution to the problem of jointly scheduling hard periodic tasks and aperiodic tasks. Is optimal for periodic tasks since it is based on EDF and optimal for aperiodic tasks since it gives the shortest response time to aperiodic tasks.

Serlin, O. (1972, May). Scheduling of Time Critical Processes. In *Proc. of*

the Spring Joint Computer Conference, Atlantic city, New Jersey, pp. 925–932. Montvale, NJ: American Federation of Information Processing Societies.
(First algorithm to handle periodic tasks.).

Shih, W.-K., J. Liu, J.-Y. Chung, and D. Gillies (1989, July). Scheduling Tasks with Ready Times and Deadlines to Minimize Average Error. *Operating Systems Review 23*(3), 14–28.
Presents a preemptive optimal algorithm for scheduling n dependent tasks with rational ready times, deadlines, and processing times on uniprocessor systems. The tasks are logically decomposed into mandatory and optional subtasks.

Shin, K. G. and Y.-C. Chang (1995, December). A reservation-based algorithm for scheduling both periodic and aperiodic real-time tasks. *IEEE Transactions on Computers 44*, 1405–1419.

Sprunt, B. (1990, August). *Aperiodic Task Scheduling for Real-Time Systems.* Ph. D. thesis, Department of Electrical and Computer Engineering, Canergie Mellon University.
Develops the Sporadic Server Algorithm for scheduling aperiodic tasks in real-time systems. Demonstrates that the Sporadic Server algorithm is able to guarantee deadlines for hard-deadline aperiodic tasks and provides good responsiveness for soft-deadline tasks.

Sprunt, B., J. Lehoczky, and L. Sha (1988, December). Exploiting Unused Periodic Time For Aperiodic Service Using the Extended Priority Exchange Algorithm. In *Proc. of IEEE Real-Time Systems Symposium*, pp. 251–258.
Presents an extended version of the priority exchange server algorithm for exploiting unused periodic time.

Sprunt, B., L. Sha, and J. Lehoczky (1989). Aperiodic Task Scheduling for Hard Real-Time Systems. In *J. of Real-Time Systems*, pp. 27–60.
Develops the Sporadic Server algorithm for scheduling aperiodic tasks in real-time systems. This algorithm extends the rate monotonic algorithm which was designed to schedule periodic tasks. It guarantees deadlines for hard periodic tasks and provide good responsiveness for soft aperiodic tasks.

Spuri, M. and G. Buttazzo (1994, December). Efficient Aperiodic Service under Earliest Deadline Scheduling. In *Proc. of the 15th IEEE Real-Time Systems Symposium*, Portorico, pp. 2–11.
Proposes and studies server algorithms under earliest deadline first schedul-

ing. Studies the performance of the Improved Priority Exchange algorithm.

Spuri, M. and G. Buttazzo (1996). Scheduling aperiodic tasks in dynamic priority systems. *Real-Time Systems Journal 10*, 179–210.

Spuri, M., G. Buttazzo, and F. Sensini (1995, December). Robust Aperiodic Scheduling under Dynamic Priority Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, Pisa , Italy, pp. 210–219.
Extends the Total Bandwidth Server algorithm to handle firm aperiodic tasks and then integrates a guarantee mechanism that allows to achieve degradation in case of transient overload.

Stankovic, J., C. Lu, S. Son, and G. Tao (1999, June). The Case for Feedback Control Real-Time Scheduling. In *Proc. of the Euromicro Conference on Real-Time Systems.*
Studies the use of feedback control concepts in soft rea-time scheduling systems, with the goal of the development of a theory of feedback control scheduling.

Stewart, D. and P. Khosla (1991, May). Real-Time Scheduling of Sensor-Based Control Systems. In *Proc. of the Eighth IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, pp. 144–150.
Proposes the maximum urgency first algorithm, which is a mixed priority real-time scheduling algorithm (combination of fixed and dynamic priority scheduling). The motivation behind this algorithm is to provide guaranteed soft real-time scheduling.

Streich, H. (1995). TaskPair-Scheduling: An Approach for Dynamic Real-Time Systems. *Int. Journal of Mini & Microcomputers 17*(2), 77–83.
Presents an on-line scheduling approach which merges the concepts of guaranteeing (an activity) and exception handling (due to time outs). It gives a guarantee, when the task is accepted, that the TaskPair will hold its time constraints.

Strosnider, J. (1988, August). *Highly Responsive Real-Time Token Rings.* Ph. D. thesis, Department of Electrical and Computer Engineering, Canergie Mellon University.
Develops the deferrable server algorithm. The execution of the deferrable server is replenished periodically. Unlike a poller, when a deferrable server finds no aperiodic jobs for execution, it preserves its budget.

Sun, J. and M. G. J. Liu (1997, October). Bounding Completion Times of Jobs with Arbitrary Release Times, Variable Execution Times, and Resource Sharing. *IEEE Transactions on Software Engineer-*

ing 23(10), 603–615.
Presents three algorithms for computing upper bounds on the completion times of jobs that have arbitrary release times and priorities.

Swaminathan, V. and K. Chakrabarty (2000, November). Real-Time Task Scheduling for Energy-Aware Embedded Systems. In *Proc. of the IEEE Real-Time Systems Symposium.*
Presents an approach for scheduling periodic tasks in real-time systems. The presented approach minimizes the total energy consumed by the task set and guarantees that the deadline for every periodic task is met.

Tokuda, H. and C. Mercer (1989, July). Arts: A Distributed Real-Time Kernel. In *Operating Systems Review*, Volume 23 of *3*, pp. 29–53. ACM Press.
Introduces a real-time object model and the integrated time-driven scheduling model to develop real-time computing systems in a distributed environment. Describes the Advanced Real-Time Technology (ARTS) kernel and the real-time toolset consisting of schedulability analyzer, *Scheduler 1-2-3*, and the real-time monitor/debugger.

Young, M. and L.-C. Shu (1991). Hybrid online/offline scheduling for hard real-time systems. Technical Report SERC-TR-100-P, Software Engineering Research Center, Department of Computer Sciences, Purdue University.
Constructs an off-line scheduler that optimally allocates idle time to improve rate-monotonic schedulability.

Chapter 7

Predictable Multiprocessor Scheduling

One step further than dealing with predictable real-time scheduling upon monoprocessor production stages in manufacturing systems (refer to Chapter 6) is to predictably schedule aperiodic tasks upon multiprocessor production stages. This chapter deals with this problem. First, a comparative study of state-of-the-art real-time multiprocessor schemes is provided. The preemptive scheduling of systems of hybrid (periodic and aperiodic) tasks on a platform comprised of several uniform multiprocessors is considered. A scheduling algorithm, the Total Bandwidth Server on uniform multiprocessors, is developed. The scheduling algorithm is analyzed by considering its performance when it is allowed to run on faster machines. The predictability of the system is proved through schedulability analysis techniques and illustrated by an example. The performance of the multiprocessor algorithm proposed in this chapter is analyzed through simulation experiments.

7.1 Taxonomy of Multiprocessor Platforms

Scheduling theorists distinguish between at least three different kinds of multiprocessor machines depending on their production time:

- **Parallel identical machines:** All parallel machines have equal task production time.
- **Parallel uniform machines:** Machines differ in their production time, but the production time of each machine is constant and does not depend from the type of the task.
- **Parallel unrelated machines:** The production time of the machine depends on the particular task processed.

The uniform parallel machines model is a very relevant one for modelling many actual application systems including mamufacturing systems. There are several reasons for this:

- The existence of this model gives production system designers the freedom to use machines with different production speeds, rather than constrainting them to always use identical processors. In fact, production lines with multiprocessor platforms in a production stage are widespread.
- Even when all the machines available are identical, they may not all be exclusively available for the execution of the real-time periodic tasks, but may be required to devote a certain time of their production capacity to some other (non real-time or aperiodic tasks). Each such machine can be modelled by another one of lower production capacity.
- As new and faster machines become available, one may choose to improve the performance of a system by upgrading some of its machines. If the only model we have available is the identical multiprocessor model, we must necessarily replace all the processors simultaneously. With the uniform parallel machines model, we can however choose to replace just a few of the processors, or indeed simply add some faster processors while retaining all previous processors.

7.2 State-of-the-Art

Scheduling of real-time systems has been well studied, particularly upon uniprocessor platforms. In multiprocessor platforms, there are several processors available upon which jobs may execute. Recently steps have been taken towards obtaining a better understanding of real-time scheduling on identical multiprocessors (Philips et al. 1997), (Moir and Ramamurthy 1999), (Abdelzaher and Shin 2000), and (Aydin et al. 1999). However, not much is known about real-time scheduling on uniform or unrelated processors.

Furthermore, task scheduling in hard real-time systems can be *static* or *dynamic*. A static approach calculates schedules for tasks off-line and it requires complete a priori knowledge of tasks' characteristics. Although static approaches have low run-time cost, they are inflexible and cannot adapt to a changing environment or to an environment whose behavior is not completely predictable. Several uniprocessor on-line algorithms, such as the Earliest Deadline First algorithm (Liu and Layland 1973) and the Least Laxity algorithm (Mok 1983) are known to be optimal in the sense that if a set of jobs that can be scheduled such that all jobs complete by their deadlines, then these algorithms will also schedule these sets of jobs to meet all deadlines. However, no on-line scheduling algorithm in multiprocessor systems can be optimal: this was shown for the simplest (identical) multiprocessor model by Hong and Leung (1988) and the result from Hong and Leung (1988) can be directly extended to the more general (uniform or unrelated) machine models.

Philips et al. (1997) explored the use of *resource augmentation* techniques[1] for the on-line scheduling of real-time tasks. They considered two problems in dynamic scheduling: scheduling to meet deadlines in a preemptive identical multiprocessor setting, and scheduling to provide good response time in a number of scheduling environments. Using the resource augmentation approach, they established that several well-known on-line algorithms, that prove poor performance from an absolute worst-case perspective, are optimal for the problems in question when allowed moderately more resources. Funk et al. (2001) extended this method to be applied upon uniform parallel machines. However, results derived from their work are applied only to periodic task systems.

The idea of this chapter is based on competitive analysis theory, introduced by Kalyanasundaram and Pruhs (1995) and extended by Funk et al. (2001). A primary contribution of this chapter is the consideration of *both* hard periodic and hard aperiodic tasks in uniform multiprocessor scheduling.

[1]A method of analysis introduced by Kalyanasundaram and Pruhs (1995) for uniprocessor scheduling, comparing the performance of an on-line algorithm to the performance of an optimal off-line algorithm when the on-line algorithm is given extra resources.

Additionally, the feature of including changeover time costs is accounted for.

Similar to the literature review provided in Chapter 5 and 6, a brief overview of relevant contributions to multiprocessor scheduling is depicted and presented in a tabular structured form in Figure 7.1[2] thus allowing the reader to get a broad overview of the literature and to establish a comparative study among the different problems addressed in the literature and the problem faced in this chapter. In addition, a brief overview and commentary of the investigated literature is presented in the bibliography at the end of the chapter.

The following essential features characterize the scheduling technique presented in this chapter: a uniform multiprocessor platform upon which the server algorithm is executed, predictability of the control scheduling technique, on-line execution, handling of periodic and aperiodic tasks considerations, preemption, and context switch time considerations. Figure 7.1 shows that no real-time scheduling approach takes into account the cited features relevant for the problem treated in this chapter, especially the ones related with changeover time considerations and predictable multiprocessor scheduling of periodic **and** aperiodic tasks.

7.3 The On-line Parallel Model

Based on the following assumptions and terminology, the scheduling of hard real-time systems upon a uniform multiprocessor platform comprised of $m \in \mathbb{N}^*$ machines (there is at least one machine) is considered in this chapter.

- $\pi = \{s_1, s_2, ..., s_m\} \mid m \in \mathbb{N}^*$ represents a m-machine uniform multiprocessor platform in which machines have *speeds* or *production capacities* $s_1, s_2, ..., s_m$ respectively; without loss of generality, it is assumed

[2]The references in the first column of Figures 6.1 and 6.2 are composed as follows: when the publication is written by a single author, the reference is composed of the first 3 letters of his family name followed by the year of publication. When the publication is written by more than one author, the reference is composed of the first letter of the family names of the authors followed by the year of publication.
The abbreviations in the first line of the table refer to the following terms: sched.: scheduling, alg.: algorithm, PT: Periodic Task, ST: Sporadic Task, AT: Aperiodic Task, Com.: Communication, CST: Context Switch Time, Int.: Integer, par.:parameters, per.: periods, SM: Shared Memory, DM: Distributed Memory, Ind.: Independent, Prec. Cons.: Precedence Constraints, JC: Jitter Control, RM: Resource Management.
The abbreviations in the rest of the table refer to the following terms: D: distributed, U: uniprocessor, M: multiprocessor, MI: multiprocessor identical, MU: multiprocessor uniform, PI: parallel identical, PU: parallel uniform, H: hard, S: soft, F:firm, Fi: fixed.

References	Year	Plat- form	Heuristic sched.	Predi- ctable sched.	Serv. alg.	Off- line	On- line	PT	ST	AT	Com. time	Pre- emption	No preem- ption	CST = 0	CST > 0	Int. sched. par.	Int. per.	Prio- rity	SM	DM	Ind. Tasks	Prec. Cons.	JC	RM	Over- load
DL78	1978	M		X										X											
LM81	1981	PI, PU		X				X				X		X			X				X				
RS84	1984	M		X										X											
DD86	1986	M		X										X											
ZRS87, ZRS87a	1987	M	X											X											
TWW87	1987	M	X											X											
SLCG89	1989	U, MI	X					X				X									X				
RSS90	1990	M	X										X								X				
BHR90	1990	U, PI						X				X		X											
SR91	1991	M, D	X			X		X		S															
DTB93	1993	U, M				X	X	X	X			X		X				Fi							
KS93	1993	PI		X			X			F		X		X					X	X	X				
FS94	1994	M		X			X	X	X	HS									X						
OS94	1994	M	X			X		H																	
TL95	1995	M	X					X																	
LH95	1995	M, D		X								X	X	X							X				
RM95	1995	M, D						H				X						Fi							
BCPV96	1996	MI						X									X								
PSTW97	1997	MI		X			X					X	X	X							X				
AH98	1998	M, D	X			X		X																	
MR99	1999	MI										X						X							
PBWB00	2000	M, D					X	X	X		X	X													
GLN01	2001	M										X													
ZMC01	2001	M																			X	X			
HA01	2001	M						X																	
RS01	2001	M		X				S																	
NLC01	2001	M	X							X															
ABJ01	2001	MI						X				X						Fi							
AS00a	2001	MI						X																	
FGB01	2001	MU		X			X	X																	
MW01	2001	PU		X			X	F						S											

Figure 7.1: Real-time multiprocessor algorithms evaluation

that these speeds have positive values and they are indexed in a decreasing manner: $s_q \geq s_{q+1}$ for all $q, 1 \leq q < m$.

- $\tau = \{\tau_{i,j} \mid i, j \in \mathbb{N}\}$ A set of periodic tasks with <u>hard</u> deadlines. i denotes the number of the task and j the group to which its is affiliated according to its changeover time.
- $J = \{J_{i,j} \mid i, j \in \mathbb{N}\}$ A set of active aperiodic tasks ordered by increasing deadline, $J_{1,j}$ being the task with the shortest absolute deadline.
- Each Job $\tau_{i,j}$ or $J_{i,j}$ is characterized by an arrival time r_i, a production time c_i and a deadline d_i, respectively r_j, c_j, d_j. Each Job $\tau_{i,j}$ is additionally characterized by a period p_i. Whereas arrival times, production times and deadlines of the periodic jobs $\tau_{i,j}$ are known in advance, it is assumed that, for an aperiodic job set $J_{i,j}$, these relevant information about jobs are known when a job arrives.
- The preemptive multiprocessor scheduling model is considered. In the preemptive scheduling model presented in this thesis, a job may be interrupted and subsequently resumed *with* a penalty.
  $O = \{O_{j,m} \mid j, m \in \mathbb{N}^*\}$ represents the switch time or changeover time caused by the arrival of the part from type j at the machine m. Changeover time is derived by a static analysis on the machine.
- $u_i = c_i/p_i$ The utilization u_i of a task is the ratio of its execution requirement to its period. Without loss of generality, the tasks in τ and J are indexed according to a decreasing utilization: $u_i \geq u_i + 1$ for all i, $1 \leq i < n$.
- In the context of uniform multiprocessor scheduling, a work-conserving scheduling algorithm is defined to be one that satisfies the following conditions (Funk et al. 2001): i) no machine is idled while there are active jobs awaiting execution and, ii) if at some instant there are fewer than m (the number of processors in the uniform multiprocessor platform) active jobs awaiting execution then the active jobs are executed upon the fastest machines. More formally, at any instant t and for all $k > j$, if the j'th-slowest processor is idled by the work-conserving scheduling algorithm, then the k'th-slowest processor is also idled at instant t.
- **Job preemption is permitted.** That is, a job executing on a machine may be preempted, prior to completing execution, and its ex-

ecution may be resumed later. Unfortunately, state-of-the-art real-time multiprocessor scheduling techniques assume that there is no penalty associated with such preemption. It is obvious that disregarding this assumption is inappropriate for manufacturing systems where changeover time overhead may have a considerable time value.

- **Job migration is permitted.** That is, a job that has been preempted on a particular machine may resume execution on the same or different processor. The penalty associated with such migration is unfortunately not accounted for in the literature. Manufacturing system applications necessitate that transport costs of a part or product from one machine to another are regarded. Accounting for transport costs in uniform multiprocessor platforms is beyond the scope of this thesis.

- **Job parallelism is forbidden.** That is each job may execute on at most one processor at any given instant in time.

7.4 Total Bandwidth Server on Uniform Multiprocessors

Recall that the TBS server technique is used to schedule jointly hard periodic and hard aperiodic tasks under dynamic priority systems upon uniprocessor platforms. One main benefit of this technique is that it guarantees both periodic and aperiodic task sets. An extension of the TBS technique to include changeover time costs is developed in Section 6.6 page 102. Each aperiodic request receives a deadline

$$\overline{d}_j = \overline{r}_j + \frac{\overline{C}_j + 2 * \overline{O}_{j,m}}{U_s}$$

where

$$\overline{r}_j = max\ (r_j, \overline{d}_{j-1}, f_{j-1})$$

In this chapter, we define a TBS algorithm to be implemented upon uniform multiprocessor systems according to the following rules:

- No machine is idled while there is an active job awaiting execution.

- When fewer than m jobs are active, they are acquired to execute upon the fastest machines while the slowest are idled.

- Higher priority jobs are executed on faster processors. More formally, if the j'th-slowest processor is executing job J_g at time t under the TBS implementation, it must be the case that the deadline of J_g is not greater than the deadlines of jobs (if any) executing on the $(j+1)$'th-, $(j+2)$'th-, $(j+3)$'th-, ..., m'th-slowest machines.

- Whenever the j-th aperiodic task arrives at time $t = r_j$, it receives a deadline

  $$\overline{d}_j = \overline{r}_j + \frac{\overline{C}_j + 2 * \overline{O}_{j,m}}{U_s}$$

  where

  $$\overline{r}_j = max\ (r_j, \overline{d}_{j-1}, f_{j-1})$$

The utilization of the server U_s will be defined later.
Unavoidably, some additional notations are given in the following.

Definition 1 (W(A,π,I,t)). *Let I denote any set of jobs, and π any uniform multiprocessor platform. For any algorithm A and time instant $t \geq 0$, let W(A,π,I,t) denote the amount of work done by algorithm A on jobs of I over the interval [0,t), while executing on π.*

Definition 2 (S_j). *Let π denote a m-processor uniform multiprocessor platform with processor capacities $s_1, s_2, ..., s_m, s_j \geq s_{j+1}$ for all j, $1 \leq j < m$. S_j is defined as follows:*

$$S_j = \sum_{l=1}^{j} s_l \ \ for\ all\ j,\ 1 \leq\ j \leq m$$

Definition 3 (λ_π). *(Funk et al. 2001) Let π denote a m-processor uniform multiprocessor platform with processor capacities $s_1, s_2, ..., s_m, s_j \geq s_{j+1}$ for all j, $1 \leq j < m$. λ_π is defined as follows:*

$$\lambda_\pi = \max_{j=1}^{m} \left\{ \frac{\sum_{k=j+1}^{m} s_k}{s_j} \right\}$$

The parameter λ_π measures the "degree" by which π differs from an identical multiprocessor platform. Consequently, λ_π becomes progressively smaller as the speeds of the processors differ from each other by greater amounts.

Lemma 1. (Funk et al. 2001) *Let π denote a m-processor uniform multiprocessor platform with processor capacities $s_1, s_2, ..., s_m, s_j \geq s_{j+1}$, for all j, $1 \leq j < m$. Let π' denote a m-processor uniform multiprocessor platform with processor capacities $s'_1, s'_2, ..., s'_m, s'_j \geq s'_{j+1}$, for all j, $1 \leq j < m$. Let A denote any m-processor uniform multiprocessor algorithm, and A' any work-conserving m-processor uniform multiprocessor algorithm. If the following condition is satisfied by π and π':*

$$S'_m \geq \lambda_{\pi'}.s_1 + S_m \tag{7.1}$$

then for any set of jobs I and at any time-instant $t \geq 0$

$$W(A', \pi', I, t) \geq W(A, \pi, I, t) \tag{7.2}$$

Lemma 1 specifies a condition under which any work-conserving algorithm A' (such as TBS) executing on π' is guaranteed to complete at least as much work as any other algorithm A (including an optimal algorithm) executing on π, when both algorithms are executing any set of jobs I. This condition expresses the additional production capacity needed by π' in terms of the $\lambda_{\pi'}$ parameter, and the speed of the fastest processor in π. The smaller the value of $\lambda_{\pi'}$ (the more π' deviates from being an identical multiprocessor), the smaller the amount of this excess processing capacity is needed.

The processing of aperiodic tasks can be integrated into a periodic environment by introducing one or more periodic tasks to execute the aperiodic tasks. Therefore, we may deal with aperiodic tasks in a similar way to periodic tasks. As a result, the following theorem uses Lemma 1 to deduce whether a work-conserving algorithm can feasibly schedule a task set: it states that any collection of jobs I that is feasible on a uniform multiprocessor platform π will be scheduled to meet all deadlines by algorithm TBS on any platform π' satisifying the condition of lemma 1.

Theorem 1 *Let I denote an instance of jobs that is feasible on m-processor uniform multiprocessor platform π. Let π' denote another m-processor uniform multiprocessor platform. Let the parameter $\lambda_{\pi'}$ of π' be as defined in Definition 3:*

$$\lambda_{\pi'} = \max_{j=1}^{m} \left\{ \frac{\sum_{k=j+1}^{m} s'_k}{s'_j} \right\}$$

If the condition of Lemma 1 is satisfied by platforms π and π':

$$S'_m \geq \lambda_{\pi'}.s_1 + S_m$$

then I will meet all deadlines when scheduled using TBS algorithm executing on π'.

Proof. The proof of Theorem 1 is, like the TBS algorithm itself, simple. By definition, TBS assigns a deadline to an aperiodic request. The request is then inserted into an input buffer and scheduled by EDF. Therefore, it remains to prove that Theorem 1 is valid under EDF. This is done in Funk et al. (2001). Theorem 1 follows.

Before proceeding further, the following definition is given.

Definition 4 (s-speed approximation algorithm). *(Philips et al. 1997) Given an input I to a scheduling problem with m-machines and an objective function value V of scheduling all jobs by their deadlines, a* ***s-speed approximation algorithm*** *finds a solution of value* ***V*** *using* ***m*** *speed-**s** machines.*

Philips et al. (1997) showed that if a set of jobs is feasible on m identical machines, then the same set of jobs will be scheduled to meet all deadlines by EDF on identical machines in which the individual machines are $(2 - \frac{1}{m})$ time as fast as in the original system. Since identical parallel machines are a special case of uniform parallel machines, in which the production capacities are equal, results of Philips et al. (1997) concerning EDF-scheduling on identical multiprocessor platforms are obtained as an immediate result of Theorem 1:

Corollary 1. *TBS is a preemptive, $(2 - \frac{1}{m})$-speed algorithm for hard real-time scheduling on parallel machines.*

Theorem 2 *Given a set of n periodic tasks with machine utilization U_p and a Total Bandwidth server with machine utilization U_s, the whole set is feasibly scheduled upon a multiprocessor platform if and only if*

$$U_p + U_s \leq S_m$$

where

$$U_p = U_{p_1} + U_{p_2} + ... + U_{pm}$$

U_{p_1}, U_{p_2}, ..., U_{pm} are the periodic utilization of the 1st, 2nd, ..., m-th machine respectively , and

$$U_s = U_{s_1} + U_{s_2} + ... + U_{sm}$$

U_{s_1}, U_{s_2}, ..., U_{sm} are the total bandwidth utilization of the 1st, 2nd, ..., m-th machine respectively.

Proof. Funk et al. (2001) proved that a set of periodic tasks can be scheduled to meet all deadlines on a uniform multiprocessor platform if and only if the following constraints holds:

$$\begin{aligned}&U_p \leq S_m\\&U_k \leq S_k \ for\ all\ k = 1, ..., m\end{aligned}$$

Since TBS schedules aperiodic tasks as periodic tasks using EDF, all tasks scheduled using TBS are considered as a periodic load and thus machine utilization of periodic and aperiodic tasks is $U_p + U_s$. The theorem follows.

7.5 Schedulability Analysis of Hybrid Task Systems on Uniform Multiprocessors

In this section, the theory developed in Section 7.4 is applied to study the deadline-based scheduling of hybrid (hard periodic and hard aperiodic tasks) task systems on uniform multiprocessor platforms. The method of analysis developed in this section proceeds as follows:

1. an exact test for determining whether a given hybrid task system is feasible on a particular uniform multiprocessor platform is developed and

2. this exact feasibility test along with the results obtained in section 7.4 are used, to design a schedulability analysis for determining whether a given hybrid task system will be successfully scheduled by TBS on a specified uniform multiprocessor platform.

Funk et al. (2001) identified a uniform multiprocessor platform upon which a given periodic task system τ is schedulable. They determine a sufficient condition for τ to be successfully scheduled by EDF on any given multiprocessor platform π' (Theorem 3).

Theorem 3 (Funk et al. 2001) *Let $\pi' = [s'_1, s'_2, ..., s'_m]$ denote any m-processor multiprocessor platform, and let $\lambda_{\pi'}$ be as defined in Definition 3:*

$$\lambda_\pi = \max_{j=1}^{m} \left\{ \frac{\sum_{k=j+1}^{m} s_k}{s_j} \right\}$$

Periodic task system τ will meet all deadlines when scheduled on π' using EDF if the following condition holds:

$$S'_m \geq \lambda_{\pi'} * max \left\{ u_1, \frac{U_p}{m} \right\} + U_p$$

Below, it is shown how the problem of scheduling periodic tasks on uniform multiprocessors can be transformed to the scheduling of periodic and aperiodic tasks. Whereas EDF is a scheduling policy trying to schedule up to the whole capacity of the multiprocessor platform, the TBS algorithm upon multiprocessor platforms aims at using the whole capacity of the system, while assuring that a fraction of this capacity is dedicated to aperiodic requests. This necessitates the computation of the server utilization of the multiprocessor platform given by Theorem 4.

Theorem 4 *Let $\pi' = [s'_1, s'_2, ..., s'_m]$ denote any m-processor multiprocessor platform, and let $\lambda_{\pi'}$ be as defined in Definition 3:*

$$\lambda'_\pi = \max_{j=1}^{m} \left\{ \frac{\sum_{k=j+1}^{m} s_k}{s_j} \right\}$$

The aperiodic task system J has a utilization

$$S'_m = \lambda_{\pi,} * u_1 + U_p + U_s$$
$$U_s = S'_m - \lambda_{\pi,} * u_1 - U_p$$

Proof. According to Funk et al. (2001), a set of periodic tasks indexed according to a decreasing utilization is feasible on a m-processor uniform multiprocessor platform $\pi = (s_1, s_2, ..., s_m)$ with $s_1 = u_1$ and $S_m = U_p$. By Theorem 2, a set of periodic and aperiodic tasks achieves a maximum utilization equal to $U_p + U_s = S_m$.

Hence by Theorem 1, a set of periodic and aperiodic tasks will meet all deadlines when it is scheduled using TBS on π', if

$$S'_m \geq \lambda_{\pi'}.s_1 + S_m$$

Since $s_1 = u_1$ and $S_m = U_p + U_s$, the following inequation is obtained

$$S'_m \geq \lambda_{\pi'}.u_1 + U_p + U_s$$

As the TBS scheme upon a uniform multiprocessor platform tries to use the whole capacity of the multiprocessor platform, the precedent inequation becomes

$$S'_m = \lambda_{\pi'}.u_1 + U_p + U_s$$

This equation states the "degree" by which the production capacity of the multiprocessor platform should be reduced (in terms of $\lambda_{\pi'}$ and the speed of the fastest processor) in order to achieve a 100% guarantee of periodic and aperiodic tasks. From this equation, U_S is deduced as follows:

$$U_s = S'_m - \lambda_{\pi,} * u_1 - U_p$$

The theorem follows.

As a result, deadlines of aperiodic jobs may be computed as defined in the following corollary.

Corollary 3. *The aperiodic jobs $J(r_j, c_j)$ of a m-processor uniform multiprocessor platform are scheduled using TBS and a total bandwidth as defined in Theorem 4 with a deadline*

$$d_j = max(r_j, d_{j-1}) + \frac{c_j}{U_s}$$

Furthermore, the Corollary 4 follows directly from the results of the precedent chapter, involving changeover time costs in the TBS algorithm (Section 6.6 page 102) and allowing resource reclaiming (Section 6.7 page 102).

Corollary 4 *Aperiodic jobs $J(r_j, c_j)$ of a m-processor uniform multiprocessor platform are scheduled using TBS and a total bandwidth as defined in Theorem 4 with a deadline*

$$\overline{d}_j = \overline{r}_j + \frac{\overline{C}_j + 2 * \overline{O}_{j,m}}{U_s}$$

where

$$\overline{r}_j = max\ (r_j, \overline{d}_{j-1}, f_{j-1})$$

Theorem 4 is now illustrated by an example.

Example. *Consider a task system τ comprised of five periodic tasks (c_i, p_i)*

$$\tau = \{(15,10),(4,5),(14,20),(6,15),(2,10)\}$$

and an aperiodic task (r_i, c_i)

$$J = \{(5,3)\}$$

for this system, $u_1 = 1.5$, $u_2 = 0.8$, $u_3 = 0.7$, $u_4 = 0.4$, $u_5 = 0.2$. Suppose that τ and J are to be TBS-scheduled upon the uniform multiprocessor platform $\pi' = [3, 1, 0.5]$ - will all deadlines be met?

By Definition 3, the value of $\lambda_{\pi'}$ for the uniform multiprocessor platform π' is

$$\lambda_{\pi'} = max\left\{\frac{1+0.5}{3}, \frac{0.5}{1}\right\} = \frac{1}{2}$$

and the total production capacity is

$$3 + 1 + 0.5 = 4.5$$

By Funk et al. (2001) τ is feasible on some 3-processor uniform multiprocessor platform having a total production capacity of

$$1.5 + 0.8 + 0.7 + 0.4 + 0.2 = 3.6$$

and with the fastest processor having a production capacity

$$s_1 = u_1 = 1.5$$

By theorem 4, the aperiodic requests J are feasible on the 3-processor uniform multiprocessor platform with a total bandwidth of

$$\begin{aligned} 4.5 = 0.5 * 1.5 + 3.6 + U_s \\ 4.5 = 4.35 + U_s \\ U_s = 0.15 \end{aligned}$$

The aperiodic job $J = \{(5, 3)\}$ is to be scheduled on the 3-processor multiprocessor platform with a deadline equal to:

$$d_j = max\{5, 0\} + \frac{3}{0.15} = 25$$

and τ and J can consequently be scheduled by TBS to meet all deadlines on π' with $U_s = 0.15$ and $d_j = 25$.

7.6 Practical Conclusions

The results presented in this chapter have various practical implications, which results from the application of the resource augmentation paradigm. First, the results provide system designers with analytic guidelines for coping with lack of knowledge of the future. The results that describe the performance of an algorithm when given faster machines not only tell the system designer how much faster the processors need to be to insure a desired performance level, they also have implications for the performance of the original system in a setting where job-arrival rate is reduced.

7.7 Performance Evaluation

The purpose of this section is to evaluate the scheduling scheme proposed in this chapter. Implementation issues for TBS scheduling upon uniform multiprocessor platforms allow to evaluate the effect of aperiodic task generation over various total bandwidth utilizations and the effect of aperiodic processor utilization over various uniform multiprocessor platforms.

In the first experiment, an evaluation of tasks scheduled with TBS upon a uniform multiprocessor platforms with different periodic utilizations is achieved. The model consider a 3-multiprocessor platform with aperiodic task utilization varying from 0.15 to 2.

Figure 7.2 depicts changeover time considered in the experiment. Changeover time may achieve an overhead approximately equal to production time. This permits a moderately realistic representation of changeover time considerations in production systems.

Figure 7.3 demonstrates the significance of aperiodic server utilization when scheduling a set of aperiodic tasks upon a uniform multiprocessor platform. As seen in this figure, tasks produced with a low server utilization suffer from a broad deadline assignment. This effect may be inacceptable in applications, where aperiodic tasks are expected frequently.

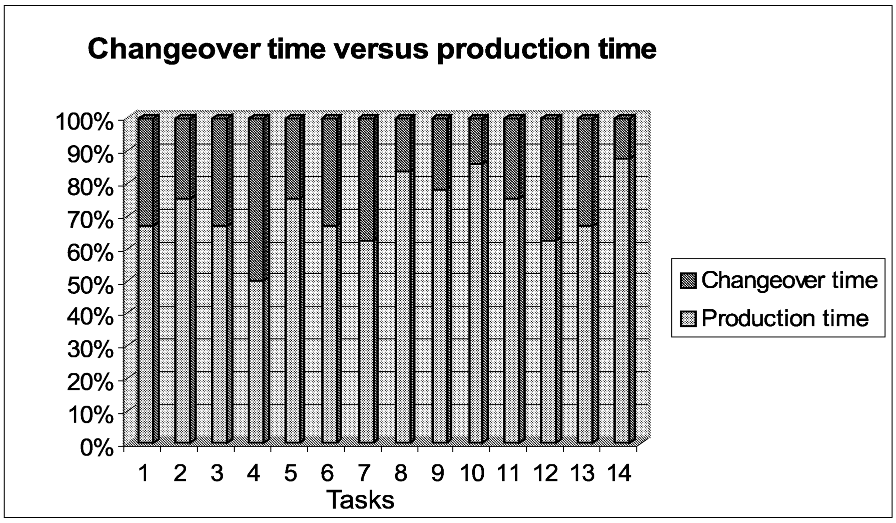


Figure 7.2: Changeover time versus production time

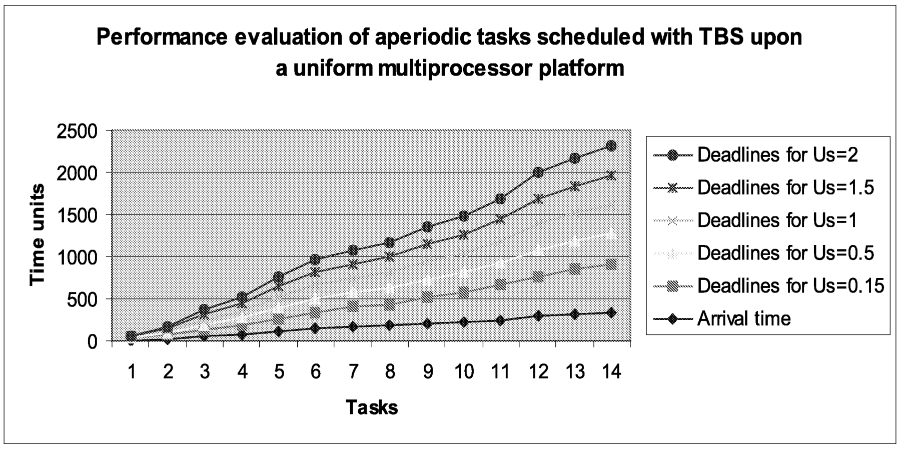


Figure 7.3: Performance evaluation of aperiodic tasks scheduled with TBS upon a uniform multiprocessor platform

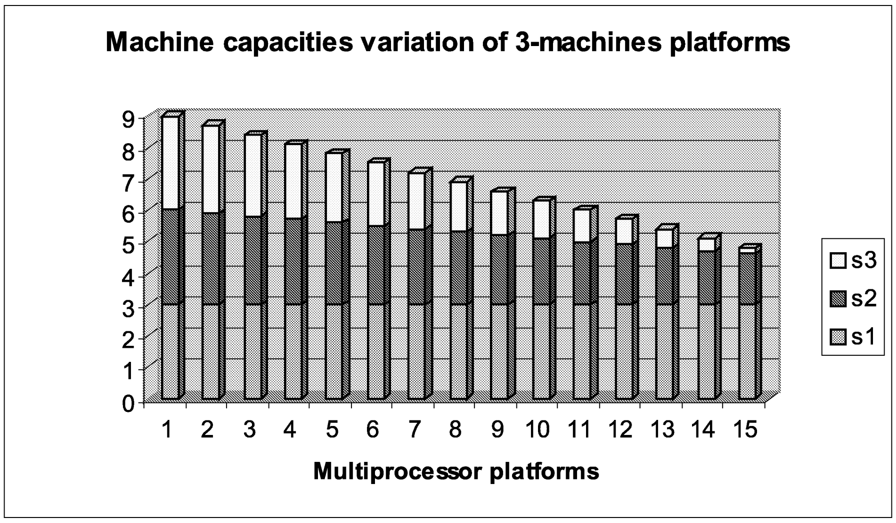


Figure 7.4: Machine capacities variation of 3-machines platforms

The second experiment aims at studying the effect of periodic and aperiodic task utilizations, when a set of periodic and aperiodic tasks are scheduled upon uniform 3-multiprocessors platforms with different production capacities. Aperiodic server utilizations are computed using Theorem 4 considering a periodic utilization of 3,6.

Figure 7.4 shows the variation of production capacities of different platforms considered in this experiment. The first multiprocessor platform is an identical platform. The production capacity of the second machine is invariable for all platforms. Production capacities of the first machines are augmented gradually by 0,2 time units. Production capacities of the third machines are reduced gradually by 0.2. The purpose of this representation of production capacities is to achieve progressive representation of results and therefore, to provide clarity for the reader. By coupling Figure 7.4 and Figure 7.5, the second experiment has established that as a multiprocessor platform differs from an identical platform, server utilizations become progressively smaller. This is due to the fact that when a multiprocessor platform differs from an identical platform, λ_π becomes smaller and consequently aperiodic server utilization (computed by Theorem 4) becomes greater. Furthermore, it should be stated that as multiprocessor platforms differ from an identical platform, the wasted utilization of machines becomes

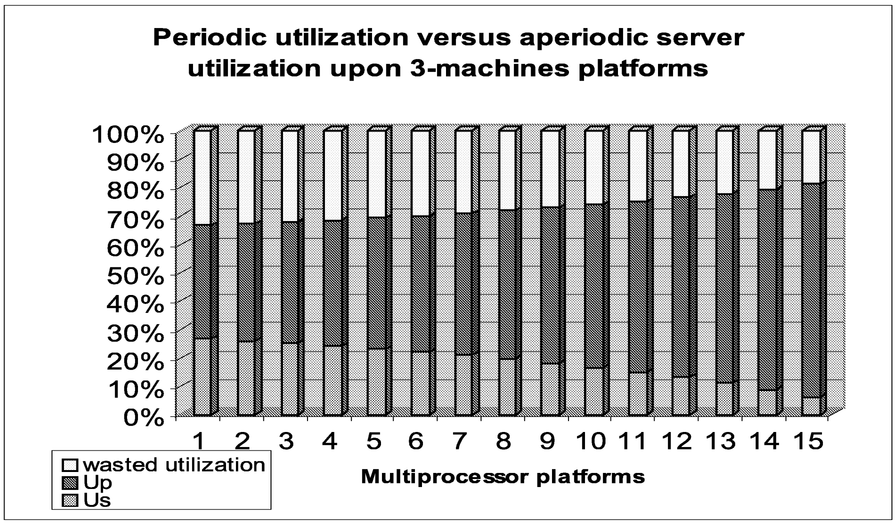


Figure 7.5: Periodic utilization versus aperiodic server utilization upon 3-machines platforms

smaller. This is due to higher aperiodic utilizations.

7.8 Contributions of the Chapter

The scientific contributions of this chapter can be summarized as follows:

- *Review of Multiprocessor Scheduling Algorithms*
  A relevant and structured overview of real-time multiprocessor scheduling theory is presented. The different algorithms are structured according to the problem they solve and to the scheduling parameters which are relevant for the real-time scheduling problem of this chapter.

- *Predictable Scheduling upon a Uniform Multiprocessor Production Stage*
  A predictable on-line scheduling scheme, the Total Bandwidth Server on uniform multiprocessor platforms, is developed. It allows the novel features of predictably scheduling hybrid (hard periodic and hard aperiodic) tasks on uniform multiprocessor platforms and accounting for changeover time costs.

- *Development of a Schedulability Analysis for the On-Line Uniform Multiprocessor Scheduling of hybrid tasks in a Hard Real-Time Environment*

- *Performance Evaluation of the Proposed Scheduling Scheme*
  Simulation studies of a prototyped multiprocessor platform of a manufacturing system underlying hard real-time constraints are generated in order to study the performance of the system.

7.9 Hypotheses Evaluation

Since this chapter provides computation techniques for predictable real-time scheduling of hard periodic and hard aperiodic tasks with a preemptive setting, it evidently validates almost all hypotheses strived for in this thesis. These are hypothesis 2, 3, 4, 5.1 and 5.2. The reader should refer to Section 1.5 page 7 of Chapter 1 for a detailed definition of the cited hypotheses.

Bibliography

Abdelzaher, T. and K. Shin (2000, January). Period-Based Load Partitioning and Assignment for Large Real-Time Applications. *IEEE Transactions on Computers 49*(1), 81–87.
Proposes an algorithm (heuristic) to the problem of workload partitioning and assignment of large heterogeneous real-time applications, which groups tasks by period. Evaluates the presented approach using simulations.

Altenbernd, P. and H. Hansson (1998). The Slack Method: A New Method for Static Allocation of Hard Real-Time Tasks. *Real-Time Systems 15*, 103–130.
Presents and evaluates the Slack Method, a constructive heuristic for the allocation of periodic hard real-time tasks to multiprocessor or distributed systems.

Anderson, J. and A. Srinivasan (2000, June). Early-Release Fair Scheduling. In *Proc. of the EuroMicro Conference on Real-Time Systems*, Stockholm, Sweden, pp. 35–43. IEEE Computer Society Press.
Presents a variant of P-fair scheduling, the early-release fair (ERfair). ERfair differs from P-fair scheduling in that it has a lower average job response times and run-time costs, particularly in a lightly-loaded systems.

Andersson, B., S. Baruah, and J. Jonsson (2001, December). Static-priority scheduling on multiprocessors. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 193–202.
Considers the problem of preempive scheduling of systems of periodic tasks in multiprocessor identical systems. Proposes a scheduling algorithm for static priority scheduling of such systems based on the extension of the uniprocessor rate monotonic scheduling algorithm.

Aydin, H., R. Melhem, S. Mossé, and P. Mejia-Alvarez (1999, December). Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona. IEEE

Computer Society Press.

Addresses the periodic reward-based scheduling problem in the context of uniprocessor systems. Focuses on linear and concave reward functions, which adequately represent realistic applications such image and speech processing, time-dependent planning and multimedia presentations.

Baruah, S., N. Cohen, C. Plaxton, and D. Varvel (1996, June). Proportionate Progress: A notion of Fairness in Resource Allocation. *Algorithmica 15*(6), 600–625.

Defines a notion of fairness, called P-fairness to be used in a variety of resource allocation problems. Shows that P-fair schedules exists for the resource sharing problem, which is a slight generalization of the periodic scheduling problem.

Baruah, S., R. Howell, and L. Rosier (1990). Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor. *Real-Time Systems 2*, 301–324.

investigates preemptive scheduling algorithms of periodic real-time task systems on one processor.

Davari, S. and S. Dhall (1986, December). An On line Algorithm For Real-Time Tasks Allocation. In *Proc. of the 7th Real-Time Systems Symposium*, New Orleans, Louisiana, pp. 194–200. IEEE.

Davis, R., K. Tindell, and A. Burns (1993). Scheduling Slack Time in Fixed Priority Pre-emptive Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, pp. 222–231.

Addresses the problem of jointly scheduling tasks with both hard and soft time constraints. Determines the maximum processing time which may be stolen from hard deadline periodic or sporadic tasks, without jeopardising their timing constraints.

Dhall, S. and C. Liu (1978, February). On a Real-Time Scheduling Problem. *Operations Research 26*(1), 127–140.

Forbes, H. and K. Schwan (1994). Rapid - a multiprocessor scheduler for dynamic real-time applications. Technical Report GIT-C-94-23, College of Computing, Georgia Institute of Technology.

Investigates and evaluates operating system support for on-line scheduling of real-time tasks on shared memory multiprocessors.

Funk, S., J. Goossens, and S. Baruah (2001, December). On-line Scheduling on Uniform Multiprocessors. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 183–192.

Considers the on-line scheduling of hard real-time systems on multi-processor machines. Results are applied to the scheduling of periodic task systems.

Gai, P., G. Lipari, and M. D. Natale (2001, December). Minimizing Memory Utilization of Real-Time Task Sets in Single and Multi-Processor Systems-on-a-chip. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 73–83.

Presents an algorithm for sharing resources in multi-processor systems by preemptive tasks. This allows to guarantee the schedulability of hard real-time task sets while minimizing RAM usage.

Holman, P. and J. Anderson (2001, December). Guaranteeing Pfair Supertasks by Reweighting. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 203–212.

Considers the "supertask" approach, in which a set of Pfair tasks is scheduled as a single task in a multiprocessor system. Presents reweighting rules for both EPDF- and EDF-scheduled component tasks.

Hong, K. and J. Leung (1988, December). On-line Scheduling of real-time tasks. In *Proc. of the Real-Time Systems Symposium*, Huntsville, Alabama, pp. 244–250. IEEE Computer Society Press.

.

Kalyanasundaram, B. and K. Pruhs (1995, Oktober). Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science*, Los Alamitos, pp. 214–223. IEEE Computer Society Press.

Introduces the notion of competitive analysis for uniprocessor scheduling, in which the on-line algorithm is allowed more resources than the optimal off-line algorithm.

Koren, G. and D. Sasha (1993, November). MOCA: A Multiprocessor On-Line Competitive Algorithm for Real-Time System Scheduling. In *Proc. of the 14th Real-Time Systems Symposium*, pp. 172–181.

Studies on-line scheduling with worst-case guarantees in multi-processor real-time environments. Considers two memory models: a distributed system having a centralized scheduler and a shared memory multiprocessor.

Lawler, E. and C. Martel (1981, Februar). A Note on Preemptive Scheduling of Periodic Real-Time Tasks. *Information Processing Letters 12*(1), 9–12.

Considers a problem in which periodic tasks are to to be preemptively scheduled on a system of parallel processors. Shows that there exists a feasible schedule if and only if there exists a feasible schedule which is cyclic with a period equal in length to the least common multiple of the periods of the individual tasks.

Liu, C. and J. Layland (1973, January). Scheduling algorithms for multiprogramming in a hard real-time environment. *J. of the ACM 20*(1), 46–61.

Describes the different aspects of scheduling of periodic systems. Presents the first proof of the optimality of rate-monotonic and Earliest Deadline First scheduling for periodic systems.

Liu, F., P. Luh, and B. Moser (1998). Scheduling and Coordination of Distributed Design Projects. *CIRPS Annals 47*, 111–113.

Studies the scheduling and coordination of distributed design projects with uncertainties while managing design risks. Presents a mathematical optimization model that balances modeling accuracy and computational complexity. Develops a solution methodology that combines Lagrangian relaxation and stochastic dynamic programming.

Liu, J. W. S. and R. Ha (1995). *Advances in Real-Time Systems* (Sang H. Song ed.)., Chapter 9, Efficient Methods of Validating Timing Constraints, pp. 196–220. Prentice Hall.

Presents worst-case bounds and efficient algorithms for determining how late the completion times of independent jobs with arbitrary release times can be in a dynamic multiprocessor or distributed system when their release times and execution times may vary from one instance to another.

Moir, M. and S. Ramamurthy (1999, December). Pfair Scheduling of Fixed and Migrating Periodic Tasks on Multiple Resources. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona. IEEE Computer Society Press.

Presents a scheduling scheme of periodic preemptable tasks on multiple resources. Considers a task model that allows arbitrary mixes of fixed and migratable tasks, and prove the existence of an optimal Pfair scheduler in this model.

Mok, A. (1983). *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment.* Ph. D. thesis, Massachusetts Institute of Technology.

(Presents the on-line algorithm Least Laxity Algorithm).

Mok, A. and M. Dertouzos (1978, November). Multiprocessor Scheduling in a Hard Real-Time Environment. In *Proc. of the 7th Texas Confer-*

ence on Computer Systems, Houston, Texas. IEEE/ACM.

Mok, A. and W. Wang (2001, December). Window-Constrained Real-Time Periodic Task Scheduling. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 15–24.
Shows that the Dynamic Window-Constrained Scheduling fails for arbitrarily low aggregate utilization rates of the packet streams. Defines the notion of Pfairness in relation to the Window-Contrained Scheduling model. Defines an EDF for the Window-Contrained scheduling problem.

Nissanke, N., A. Leulseged, and S. Chillara (2001, December). A Framework for Probabilistic Analysis of Multiprocessor Scheduling Environments. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Sets up a framework for probabilistic analysis of task scheduling in a multiprocessor environment. Assumes that the tasks are not known precisely and that they inherently carry an element of uncertainty or unpredictability.

Oh, Y. and S. Son (1994). Scheduling hard real-time tasks with tolerance of multiple processor failures. *Microprocessing and Microprogramming 40*, 193–206.
Studies the problem of scheduling a set of hard real-time tasks with duplication. Proves that the problem of scheduling a set of non-preemptive tasks on m $\geq$ 3 processors to tolerate one arbitrary processor failure is *NP-complete* even when the tasks share a common deadline. A heuritic algorithm is proposed to solve the problem.

Philips, C. A., C. Stein, E. Torng, and J. Wein (1997, May). Optimal Time-Critical Scheduling via Resource Augmentation. In *Proc. of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, Texas, pp. 140–149.
Considers the problem real-time dynamic scheduling in a preemptive multiprocessor setting, and scheduling to provide good response time in a number of scheduling environments. Uses a relaxed notion of competitive analysis, in which the on-line algorithm is allowed more resources than the optimal off-line algorithm to which it is compared.

Poledna, S., A. Burns, A. Wellings, and P. Barrett (2000, February). Replica Determinism and Flexible Scheduling in Hard Real-Time Dependable Systems. *IEEE Transactions on Computers 49*(2), 100–111.
A method, called timed messages, to avoid the inconsistent order and timing of replicated tasks in real-time distributed systems is presented. The major advantage of timed messages is its efficiency and flexibility while guaranteeing

deterministic operation of replicated tasks.

Ramamritham, K. and J. Stankovic (1984). Dynamic Task Scheduling in Hard Real-Time Distributed systems. *IEEE Software 1*(3), 65–75.

Deals with multiprocessor scheduling in hard real-time distributed systems. Uses a uniprocessor scheme for local scheduling, and perform distributed scheduling for tasks which are potentially subject to timing failures at run-time.

Ramamritham, K. and J. Stankovic (1994). Scheduling Algorithms and Operating Systems Support for Real-Time Systems. In *Proc. of the IEEE*, Volume 82 of *1*, pp. 55–67.

Presents the state of the real-time field in the areas of scheduling and operating system kernels.

Ramamritham, K., J. Stankovic, and P.-F. Shiah (1990, April). Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems. *IEEE Transactions on Parallel and Distributed Computing 1*(2), 184–194.

Presents a scheduling algorithm, based on heuristics, to schedule a set of tasks in multiprocessor systems. The approach constructs explicitly task execution plans. It assumes that non-blocking tasks with known worst case execution times and resource requirements are the entities being scheduled.

Regehr, J. and J. Stankovic (2001, December). A Framework for Composing Soft Real-Time Schedulers. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 3–14.

Presents a hierarchical CPU scheduling to support applications in open systems. .

Rhee, I. and G. Martin (1995, December). A Scalable Real-Time Synchronization Protocol for Distributed Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 21–27.

Proposes a distributed protocol for the synchronization of real-time tasks that have variable resource requirements. The protocol is intended for large-scale distributed or parallel systems in which processes communicate by message passing.

Shih, W.-K., J. Liu, J.-Y. Chung, and D. Gillies (1989, July). Scheduling Tasks with Ready Times and Deadlines to Minimize Average Error. *Operating Systems Review 23*(3), 14–28.

Presents a preemptive optimal algorithm for scheduling n dependent tasks with rational ready times, deadlines, and processing times on uniprocessor systems. The tasks are logically decomposed into mandatory and optional subtasks.

Sprunt, B. (1990, August). *Aperiodic Task Scheduling for Real-Time Systems.* Ph. D. thesis, Department of Electrical and Computer Engineering, Canergie Mellon University.

Develops the Sporadic Server Algorithm for scheduling aperiodic tasks in real-time systems. Demonstrates that the Sporadic Server algorithm is able to guarantee deadlines for hard-deadline aperiodic tasks and provides good responsiveness for soft-deadline tasks.

Stankovic, J. (1988, October). A Serious Problem for Next-Generation Systems. *Computer 21*(10), 10–19.

Stankovic, J. and K. Ramamritham (1991, May). The Spring Kernel: A New Paradigm for Real-Time Systems. *IEEE Software 8*(3), 62–72.

Implements and evaluates multiprocessor schedulers running on single, dedicated nodes of small scale parallel embedded systems using a static heuristic function, which may integrate timing, resource and preceding constraints.

Tia, T.-S. and J.-S. Liu (1995). Assigning Real-Time Tasks and Resources to Distributed Systems. *Special Issue of the International Journal of Mini and Microcomputers 17*(1), 18–25.

Presents a method for allocating periodic tasks where different tasks may have different deadlines. Graph based heuristics, which attempt to minimize interprocess communication and based on clustering and graph-bisection, are used for task assignment.

Tokuda, H., J. Wendorf, and H. Wang (1987, December). Implementation of a Time-Driven Scheduler for Real-Time Operating Systems. In *Proc. of the 8th Real-Time Systems Symposium*, San Jose, California, pp. 271–280. IEEE.

Zhao, W., K. Ramamritham, and J. Stankovic (1987a, August). Preemptive Scheduling Under Time and Resource Constraints. *IEEE Transactions on Computers C-36*(8), 949–960.

Discusses the use of a heuristic approach for scheduling atsks with timing and resource constraints. Validates the results through simulation studies.

Zhao, W., K. Ramamritham, and J. Stankovic (1987b, May). Scheduling Tasks with Resource Requirements in Hard Real-Time Systems. *IEEE Transactions on Software Engineering SE-13*(5), 564–577.

Zhu, D., R. Melhem, and B. Childers (2001, December). Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems. In *Proceedings of the Real-Time*

Systems Symposium, London, pp. 84–94.
Focuses on power-aware scheduling for multi-processor real-time systems based on the idea of slack sharing among processors. Takes into consideration tasks with and without precedence constraints.

Chapter 8

Conclusions and Outlook

◊
*"The best
Way to predict
the future is to
invent it"*
◊

A. Kay

This chapter starts by giving a summary of the precedent chapters followed by a summary of the contributions of the thesis. Additionally, it indicates suggestions for future research and development directions that would help to bring the MaSHReC into industrial practice. Finally, the complete and alphabetically sorted bibliography without commentary is produced.

8.1 Summary

This work introduced a novel area of application for real-time systems, Manufacturing Systems underlying real-time constraints, which is especially motivated by the stringent demands of Production Planning and Control Systems. Understanding basic concepts of task scheduling helps the designer engineer to guarantee satisfation of deadlines which is why it is essential to study the scheduling theory. This thesis does the complex interdisciplinary work of transferring theoretical real-time scheduling results to practical, industrial computing applications. In the following the summary of the different chapters is given.

8.1.1 Chapter 1 "Introduction"

The first chapter motivates the research presented in this thesis by emphasizing the impact of related fields which affected my research. A main object of this chapter is to present the basic hypotheses concluded from the different observations in several fields of Real-Time Systems and Production Planning and Control. The evaluation of these hypotheses is the subject of the following seven chapters.

8.1.2 Chapter 2 "The MaSHReC Architecture"

A profound approach is to construct manufacturing planning and control systems such that changes can be more easily incorporated and the guarantee of real-time cosntraints of tasks can be clearly verified. This is the topic of the first part of this thesis. Therefore, Chapter 2 introduces the architecture of a Manufacturing System under Hard Real-Time Constraints using the holonic approach.

8.1.3 Chapter 3 "The MaSHReC Design Model"

Based on the architecture presented in Chapter 2, this chapter develops the design model of a manufacturing system under hard real-time constraints. The Unified Modeling Language (UML) is used to model the system structure, components, relations, data, functions and interactions.

8.1.4 Chapter 4 "A Local Scheduling Algorithm for MaSHReC"

In the second part of the thesis, the on-line aperiodic scheduling of MaSHReC is treated. The purpose of this chapter is to develop a local scheduling algorithm for the model provided in the precedent chapter. Besides developing the local scheduling technique, Chapter 4 derives an extended schedulability test, inclusing changeover time costs, to prove the predictability of the system.

8.1.5 Chapter 5 ”A Distributed Scheduling Algorithm for MaSHReC”

Due to the advances in Flexible Manufacturing Systems technologies, distributed systems have become widespread. It is the purpose of this chapter to study the scheduling problem for distributed manufacturing systems underlying real-time constraints. This scheduling problem in its general form is known to be NP-hard. The study of real-time constraints in such systems increase the complexity of the scheduling problem. Therefore, a heuristic scheduling scheme is developed and implemented in a prototyped manufacturing cell. The performance of the proposed algorithm is examined via simulation studies.

8.1.6 Chapter 6 ”Predictable Monoprocessor Scheduling”

In order to guarantee hard timing constraints, it is necessary that the scheduling solution provides predictable response time performance for tasks with hard deadlines. The unpredicatble nature of aperiodic tasks makes it difficult to create a predictable scheduling solution that is able to meet hard timing constraints. This problem is treated by Chapter 6 and Chapter 7. Chapter 6 provides an extension of a server-based algorithm to decide if an aperiodic task is feasible on one machine in a production shift. The extended algorithm retains the advantages of the previous Total Bandwith Server algorithms, while overcoming their limitations in order to be applied to manufacturing systems, that is including changeover time overheads. The predictability of the system is proven through schedulability analysis techniques.

8.1.7 Chapter 7 ”Predictable Multiprocessor Scheduling”

In chapter 7, the scheduling of hard real-time activities is performed on-line on a uniform multiprocessor platform considering hard periodic and hard aperiodic tasks and the assignment of changeover time costs. The problem is studied using a relaxed notion of competitive analysis, in which the on-line algorithm is allowed faster machines than the off-line algorithm to which it is compared. Schedulability analysis techniques are derived to prove the predictability of the system. The performance of the multiprocessor control algorithm is analyzed through simulations.

8.2 Achievements of the Thesis

The fundamental goals of this thesis are:

1. to structure and design a methodology of a manufacturing system under real-time constraints,
2. to thoroughly examine and compare the literature of real-time scheduling theory and its application to Production Planning and Control Systems, and
3. to develop computational methods for studying the aperiodic scheduling problem in uni-, multiprocessor and distributed manufacturing systems under hard real-time constraints.

The contributions of this thesis, can be summarized in the following. The location of the contributions is provided in chapters 1 to 7.

- Motivation and formulation of the hypotheses for the modeling and scheduling of Manufacturing Systems under Real-Time Constraints (Chap. 1).
- A complete review of holonic manufacturing architectures (Chap. 2).
- Development of an architecture for Manufacturing Systems underlying real-time Constraints (Chap. 2).
- A design methodology of a manufacturing system under real-Time constraints (Chap. 3).
- Definition of a taxonomy for the real-time scheduling problem (Chap. 4).
- Review of holonic manufacturing scheduling schemes (Chap. 4).
- Development of the local scheduling policy and the schedulability test for MaSHReC (Chap. 4).
- Review of recent distributed scheduling techniques (Chap. 5).

- A classification and review of real-time distributed scheduling literature (Chap. 5).
- Development of a distributed scheduling scheme for MaSHReC (Chap. 5).
- A prototype implementation of the scheduling scheme proposed in Chapter 5 (Chap.5).
- Performance evaluation of the scheduling scheme proposed in Chapter 5.
- A review of uniprocessor real-time scheduling algorithms (Chap. 6).
- Review and evaluation of server scheduling mechanisms (Chap. 6).
- Development of a predictable scheduling scheme upon a monoprocessor production stage (Chap. 6).
- A review of multiprocessor real-time scheduling algorithms (Chap. 7).
- Development of a predictable scheduling scheme upon a uniform multiprocessor production stage (Chap. 7).
- Development of a schedulability analysis for the control algorithm provided in chapter 7 (Chap. 7).
- Performance evaluation of the control algorithm provided in chapter 7 (Chap. 7).

8.3 Suggestions for Future Research and Development

Scheduling is a relevant topic for both domains of real-time systems and manufacturing systems. This thesis is a contribution to both research fields, and it provides a conceptual basis, as well as a heuristic and predictable algorithms as starting points for future work. The following extensions might be of particular interest in the future.

- *Assumptions Relaxation*
  Future work is implied from the relaxation of algorithms' assumptions defined in Chapters 4, 5, 6, and 7 (Assumptions sections). These are

not cited in this section. However, new ideas and concepts resulting from this thesis are provided.

- *Real-Time Design of MaSHReC*
  The model presented in Chapter 3 is based on UML. The UML visual modeling facilities do not provide sufficient means to support real-time systems. The extension of UML to support real-time constraints is out of the scope of this thesis. This is reflected in the model presented in Chapter 3. However, this topic remains a subject of further research efforts.

- *Fault Tolerance Support*
  A desirable feature of manufacturing systems underlying real-time constraints is the support of *fault tolerance*. Providing fault tolerance in a system allows executing tasks to survive failures within the system. Without fault tolerance, an application program executing in parallel on multiprocessors in a distributed system could fail entirely if even a single processor executing part of it fails. Therefore, it would be necesssary that critical components of the real-time manufacturing system are designed to be fault tolerant.

- *Overload Support*
  Real-time production planning and control systems may suffer from overload conditions. While overload support is a known problem in the real-time scheduling theory, a future work would be how to provide means to support real-time scheduling of manufacturing systems in overload conditions.

- *Evaluation of the Heuristic Algorithm for Large Manufacturing Systems* (Chap. 5)
  Although numerous experiments have been conducted, further experiments are necessary to investigate the performance of the heuristic algorithm for large systems. Besides, in the the simulation, only production cells with one machine were considered. It remains to be studied how uniform multiprocessor machines in a production shift perform when using the presented scheduling technique.

- *Implication of Transport Costs in the Proposed Predictable Control Scheme*
  This thesis provides a predictable scheduling for deciding if an aperiodic task is feasible on uniform multiprocessors in a production shift.

In a manufacturing system, where a substantial amount of time is spent to transport parts among machines, it would be primary to account for transport overheads.

- *Extension of the Results to Other Real-Time Application Domains*
  The application focused on in this thesis is manufacturing planning and control. However, results (or extended results) of this thesis would be of great interest for a vast array of real-time system applications underlying hard or firm real-time constraints and/or suffering from high changeover time costs such as automotive electronics, air traffic control, railway switching systems, large-scale multiprocessors[1].

- *Involvement of Further Timing Requirements*
  A typical characteristic of real-time systems is the concurrent processing of tasks under strict timing requirements. These timing requirements may impose not only direct constraints, such as deadlines, but also indirect timing constraints in terms of inter-task dependencies, limited buffer capacities, optimization of storage usage, share of common resources. Theoretical developments are also required to involve such timing requirements.

Finally, it is anticipated that despite its maturity and development, the real-time scheduling area will continue to offer challenging open problems for a long time in the future.

[1]As multiprocessors grow in size and complexity, latency tolerance of synchronization faults and remote menory accesses becomes increasingly important. Accounting for these latencies is crucial when evaluating machine utilization.

Bibliography

Abdelzaher, T. and K. Shin (1995, December). Optimal Combined Task and Message Scheduling in Distributed Real-Time Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 162–171.
Presents an optimal algorithm for combined task and message scheduling in distributed hard real-time systems. Uses a branch-and-bound technique to find an optimal solution to the problem.

Abdelzaher, T. and K. Shin (1997, September). Comment on "A Pre-Run-Time Scheduling Algorithm for Hard Real-Time Systems". *IEEE Transactions on Software Engineering 23*(9), 599–600.
Shows that the branch-and-bound implicit enumeration algorithm does not always succeed in finding a feasible solution, and describes the reason why this algorithm might fail.

Abdelzaher, T. and K. Shin (2000, January). Period-Based Load Partitioning and Assignment for Large Real-Time Applications. *IEEE Transactions on Computers 49*(1), 81–87.
Proposes an algorithm (heuristic) to the problem of workload partitioning and assignment of large heterogeneous real-time applications, which groups tasks by period. Evaluates the presented approach using simulations.

Altenbernd, P. and H. Hansson (1998). The Slack Method: A New Method for Static Allocation of Hard Real-Time Tasks. *Real-Time Systems 15*, 103–130.
Presents and evaluates the Slack Method, a constructive heuristic for the allocation of periodic hard real-time tasks to multiprocessor or distributed systems.

Anderson, J. and A. Srinivasan (2000, June). Early-Release Fair Scheduling. In *Proc. of the EuroMicro Conference on Real-Time Systems*, Stockholm, Sweden, pp. 35–43. IEEE Computer Society Press.
Presents a variant of P-fair scheduling, the early-release fair (ERfair). ER-

fair differs from P-fair scheduling in that it has a lower average job response times and run-time costs, particularly in a lightly-loaded systems.

Andersson, B., S. Baruah, and J. Jonsson (2001, December). Static-priority scheduling on multiprocessors. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 193–202.

Considers the problem of preempive scheduling of systems of periodic tasks in multiprocessor identical systems. Proposes a scheduling algorithm for static priority scheduling of such systems based on the extension of the uniprocessor rate monotonic scheduling algorithm.

Atlas, A. and A. Bestravos (1998, December). Statistical Rate Monotonic Scheduling. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain.

Statistical Rate Monotonic Scheduling is a generalization of the classical Rate Monotonic Scheduling results of Liu and Layland. This algorithm allows the scheduling of periodic tasks with variable resource requirement to achieve a requested statistical Quality of Service guarantee. It Yield controllable and predictable Quality of Service unrelated to the period of a given task.

Audsley, N., A. Burns, M. Richardson, and A. Wellings (1991). Hard Real-Time Scheduling: The Deadline Monotonic Approach. In *Eighth IEEE Workshop on Real-Time Operating Systems and Software*, pp. 133–137.

Investigates schedulability tests for sets of periodic processes whose deadlines are permitted to be less than their period. Such a relaxation enables sporadic processes to be directly incorporated without alteration to the process models.

Aydin, H., R. Melhem, D. Mossé, and P. Mejia-Alvarez (2001, December). Dynamic and Agressive Scheduling Techniques for Power-Aware Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 95–105.

Addresses power-aware scheduling of periodic hard real-time tasks using dynamic voltage scaling.

Aydin, H., R. Melhem, S. Mossé, and P. Mejia-Alvarez (1999, December). Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona. IEEE Computer Society Press.

Addresses the periodic reward-based scheduling problem in the context of uniprocessor systems. Focuses on linear and concave reward functions, which adequately represent realistic applications such image and speech processing, time-dependent planning and multimedia presentations.

Baruah, S. (1995, December). Fairness in Periodic Real-Time Scheduling. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 200–209.

Describes a quantitative measure of temporal fairness "pfairness" in periodic real-time scheduling. Presents the Weight-Monotonic scheduling algorithm, a static priority scheduling algorithm for generating "pfair" schedules.

Baruah, S., N. Cohen, C. Plaxton, and D. Varvel (1996, June). Proportionate Progress: A notion of Fairness in Resource Allocation. *Algorithmica 15*(6), 600–625.

Defines a notion of fairness, called P-fairness to be used in a variety of resource allocation problems. Shows that P-fair schedules exists for the resource sharing problem, which is a slight generalization of the periodic scheduling problem.

Baruah, S., R. Howell, and L. Rosier (1990). Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor. *Real-Time Systems 2*, 301–324.

investigates preemptive scheduling algorithms of periodic real-time task systems on one processor.

Baruah, S., A. Mok, and L. Rosier (1990, December). Preemptively Scheduling Hard Real-Time Sporadics Tasks on One Processor. In *Proc. of the Real-Time Systems Symposium*, pp. 182–190. IEEE Computer Society Press.

Gives a necessary and sufficient conditions for a sporadic task system to be preemptively schedulable on one processor.

Bate, I. and A. Burns (1999). A Framework for Scheduling in Safety-Critical Embedded Control Systems. In *Proc. of the 6th International Conference on Real-Time Computing Systems and Apllications.*

Presents a computational model that supports the reuse of legacy systems. Develops timing analysis that features low pessimism and low computational complexity.

Bernat, G. and C. Cayssials (2001, December). Guaranteed On-Line Weakly-Hard Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 25–35.

Presents an on-line scheduling framework called Bi-Modal Scheduler for weakly hard real-time systems. In a normal mode, tasks can be scheduled with a generic scheduler. Weakly hard constraints are guaranteed to be satisfied by switching to a panic mode for which schedulability tests exist.

Bestavros, A. and D. Spartiotis (1993, May). Probabilistic Job Scheduling for Distributed Real-Time Applications. In *Proc. of the First IEEE Workshop on Real-Time Applications*, New York, NY.

Describes a heuristic for the dynamic real-time scheduling in a distributed environment. When a task is submitted to a node, the scheduling software tries to schedule the task locally so as to meet its deadline. If it fails, it tries to locate another node where this could be done with a high probability of success.

Bettati, R. (1994). *End-to-End Scheduling to Meet Deadlines in Distributed Systems.* Ph. D. thesis, University of Illinois at Urbana-Champaign, Zrich.

Presents two algorithms for scheduling flow shops where tasks can be serviced more than once by some processors. Describes a technique to schedule flow shops that consist of periodic tasks and to analyze their schedulability.

Biyabani, S., J. Stankovic, and K. Ramamritham (1988, December). The Integration of Deadline and Criticalness in Hard Real-Time Scheduling. In *Proc. of the 9th Real-Time Systems Symposium*, Los Alamitos, California, pp. 152–169. CS Press.

Presents two heuristric approaches for distributed hard real-time computer systems. The algorithms explicitely account for both deadlines and criticalness of tasks when making scheduling decisions.

Bongaerts, L. (1998). *Integration of Scheduling and Control in Holonic Manufacturing Systems.* Ph. D. thesis, PMA/K.U. Leuven.

Develops shop floor control algorithms based on the notion of holonic manufacturing systems.

Booch, G. (1991). *Object Oriented Design with Applications.* Redwood City, California: Benjamin/Cummings.

Presents a unified notation that incorporates the Boochś notation (first book) and other methods. Includes several examples of projects implemented in C++.

Booch, G., J. Rumbaugh, and I. Jacobson (1999). *The Unified Modeling Language User Guide.* Addison-Wesley.

Presents the UML conceptual model and applies UML to series of modeling problems.

Burns, A. and G. Bernard (2001). Jorvik: A framework for effective scheduling. Technical Report YCS-334, Department of Computer Sci-

ence, University of York.
Presents a collection of mechanisms that together form a framework for the support of flexible scheduling with mix hard and soft tasks, periodic and aperiodic load and intertask relationships.

Burns, A., K. Tindell, and A. Wellings (1995, May). Effective Analysis for Engineering Real-Time Fixed Priority Schedulers. In *IEEE Transactions on Software Engineering*, Volume 21 of *5*, pp. 475–480.
Presents an analysis that enables the cost of the scheduler (clock overheads, queue manipulations and release delays) to be factored into the standard equations of calculating worst-case response times in hard real-time systems.

Buttazzo, G. (1997). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications.* Kluwer academic Publishers.
Introduces basic concepts for real-time computing, with emphasis on predictable scheduling algorithms. Handles periodic and aperiodic task scheduling, tasks with precedence constraints, access protocols to shared resources, schedulability analysis, and handling overload conditions.

Buttazzo, G., M. Spuri, and F. sensini (1995, December). Value vs Deadline Scheduling in Overload Conditions. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 90–99.
Presents a comparative study among scheduling algorithms which use different priority Assignments and different guarantee mechanism during overload conditions.

Caccamo, M., G. Lipari, and G. Buttazzo (1999, December). Sharing Resources among Periodic and Aperiodic Tasks with Dynamic Deadlines. In *Proc. of the IEEE Real-Time Systems Symposium, Phoenix, Arizona*, pp. 284–293.
Addresses the problem of scheduling hybrid real-time task sets consisting of hard periodic and soft aperiodic tasks that may share resources in exclusive mode in a dynamic environment. Considers that resources are accessed through the Stack Resource Policy and aperiodic tasks are serviced by the tunable Total Bandwidth Server.

Caccamo, M. and L. Sha (2001, December). Aperiodic Servers with Resource Constraints. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 161–170.
Addresses the problem of integrating resource constraints in the execution of hybrid task sets including hard period and soft aperiodic task sets using a capacity based server.

Casey, L. (1981). Decentralized Scheduling. *The Australian Computer Journal 13*(2).

Chiu, J.-F. and G.-M. Chiu (2001, December). Placing Forced Checkpoints in Distributed Real-Time Embedded Systems. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Presents a scheme for placing forced checkpoints in a distributed real-time embedded systems so as to make all checkpoints useful for rollback recovery.

Davari, S. and S. Dhall (1986, December). An On line Algorithm For Real-Time Tasks Allocation. In *Proc. of the 7th Real-Time Systems Symposium*, New Orleans, Louisiana, pp. 194–200. IEEE.

Davis, R., K. Tindell, and A. Burns (1993). Scheduling Slack Time in Fixed Priority Pre-emptive Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, pp. 222–231.

Addresses the problem of jointly scheduling tasks with both hard and soft time constraints. Determines the maximum processing time which may be stolen from hard deadline periodic or sporadic tasks, without jeopardising their timing constraints.

Davis, R. and A. Wellings (1995, December). Dual Priority Scheduling. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 100–109.

Presents a strategy for scheduling tasks with soft deadlines in real-time systems containing periodic, sporadic and adaptive tasks with hard deadlines.

Dhall, S. and C. Liu (1978, February). On a Real-Time Scheduling Problem. *Operations Research 26*(1), 127–140.

DiNatale, M. and J. Stankovic (1995, December). Applicability of Simulated Annealing Methods to Real-Time Scheduling and Jitter Control. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 190–199.

Introduces a scheduling approach, which minimizes jitter for periodic tasks in distributed static systems. Presents a general framework consisting of an abstract architecture model and a general programming model are .

Dussa-Zieger, K. (1998). *Model-Based Scheduling and Configuration of Heterogeneous Parallel Systems.* Ph. D. thesis, University Erlangen-Nürnberg.

Studies the problem of scheduling in heterogeneous multiprocessor systems. Uses and implements heuristic algorithms based on genetic algorithms and tabu search.

Eager, D., E. Lazowska, and J. Zahorajan (1986, May). Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Trans. on Software Engineering SE-12*(5), 662–675.

Uses the *system decomposition* technique to evaluate three types of load sharing algorithms. This technique enables the entire system to be modeled in terms of a single node, replying upon the conjecture that the decomposition method is asymptotically exact as the number of nodes, N, becomes larger. Concludes that any redistribution strategy was better than none, and that simple policies were almost as effective as more complex ones.

El-Kebbe, D. A. (2000a, May). Integration of On-Line and Off-Line Systems in Real-Time Manufacturing. In *Proc. of the Workshop of the Informatics Graduate Colleges*, Schloss Dagstuhl, Germany.

Introduces a methodology to integrate off-line and on-line production planning systems to achieve both flexibility and a guarantee for critical production tasks.

El-Kebbe, D. A. (2000b, October). Modeling the Manufacturing System under Hard Real-Time Constraints Using Real-Time UML. In *Workshop on Formal Design Techniques Using Real-Time UML*, York, UK.

Discusses modeling techniques for MaSHReC. Introduces an object oriented holonic model for a manufacturing system under real-time constraints using UML.

El-Kebbe, D. A. (2000c, September). Towards a Manufacturing System under Hard Real-Time Constraints. In *Informatik 2000: 30. Jahrestagung der Gesellschaft für Informatik*, Berlin.

Presents the basic concept of a manufacturing system under hard real-time constraints.

El-Kebbe, D. A. (2001a, April). A Real-Time Holon-Based Architecture for the Production Planning System: Further Results. In *Proc. of the Workshop on Agent Based Simulation II*, Passau, Germany.

Presents the architecture for the manufacturing system under hard real-time constraints (MaSHReC). Models the structure, bahavior and interactions of MaSHReC using UML.

El-Kebbe, D. A. (2001b, March). A UML Model for the MaSHReC Architecture. In *Proc. of the International Congress on Information Science Innovations in Intelligent Automated Manufacturing*, Dubai, United Arab Emirates.

Investigates current holonic manufacturing architectures. The MaSHReC model is designed using the Unified Modeling Language (UML). UML templates are provided to allow the design of the system structure, components, relations, data, functions and interactions.

El-Kebbe, D. A. (2001c, December). Aperiodic Scheduling in a Dynamic Real-Time Manufacturing System. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Addresses the problem of aperiodic scheduling of manufacturing systems under hard real-time constraints. Adopts the extended version of the Total Bandwidth Server algorithm and extends it to include changeover time costs.

El-Kebbe, D. A. (2001d, October). Findings from Adapting Real-Time Aperiodic Tasks to Production Planning Systems. In *Proc. of the IEEE Conference on Emerging Technologies and Factory Automation*, Antibes, France.
Presents some findings from adapting real-time operating systems scheduling theory to production planning and control. Concludes with the introduction of server mechanisms to the aperiodic scheduling of manufacturing systems with the novel feature of including changeover time costs.

El-Kebbe, D. A. (2001e, October). Scheduling of Manufacturing Systems under Hard Real-Time Constraints. In *Proc. of the IEEE Systems, Man and Cybernetics Conference*, Tucson, Arizona.
Presents a real-time distributed scheduling scheme for a manufacturing system underlying real-time constraints based on bidding and focused addressing.

El-Kebbe, D. A. (2002, April). Predictable Multiprocessor Scheduling in Manufacturing Systems underlying hard Real-Time Constraints. In *Proc. of the AIPS Workshop on On-line Planning and Scheduling*, Toulouse, France.
Presents the Total Bandwidth server algorithm and its schedulability analysis upon uniform multiprocessor platforms.

El-Rewini, H. and H. Ali (1995). Scheduling of Conditional Branches in Parallel Program. *Journal of Parallel and Distributed Computing 24*, 41–54.

Feng, W. and J. W.-S. Liu (1997, February). Algorithms for Scheduling Real-Time Tasks with Input Error and End-to-End Deadlines. *Transactions on Software Engineering 23*(2), 93–106.
Describes algorithms for scheduling preemptive, imprecise, composite tasks in a real-time system. Extends the imprecise computation technique to account

for input error and end-to-end timing constraints. Develops five algorithms to minimize the output error of each composite task.

Fidge, C. (1998, January). Real-Time Schedulability Tests for Preemptive Multitasking. *J. of Real-Time Systems 14*(1), 61–93.

A tutorial acting as a guide to the major schedulability tests available for preemptive multi-tasking applications.

Fohler, G. (1995, December). Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 152–161.

Presents an algorithm for the joint scheduling of periodic and aperiodic tasks in statically scheduled distributed real-time systems.

Forbes, H. and K. Schwan (1994). Rapid - a multiprocessor scheduler for dynamic real-time applications. Technical Report GIT-C-94-23, College of Computing, Georgia Institute of Technology.

Investigates and evaluates operating system support for on-line scheduling of real-time tasks on shared memory multiprocessors.

Funk, S., J. Goossens, and S. Baruah (2001, December). On-line Scheduling on Uniform Multiprocessors. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 183–192.

Considers the on-line scheduling of hard real-time systems on multi-processor machines. Results are applied to the scheduling of periodic task systems.

Gai, P., G. Lipari, and M. D. Natale (2001, December). Minimizing Memory Utilization of Real-Time Task Sets in Single and Multi-Processor Systems-on-a-chip. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 73–83.

Presents an algorithm for sharing resources in multi-processor systems by preemptive tasks. This allows to guarantee the schedulability of hard real-time task sets while minimizing RAM usage.

Ghosh, S., R. Melhem, and D. Mousse (1995, December). Enhancing Real-Time Schedules to Tolerate Transient Faults. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 120–129.

Presents a scheme that can be used to guarantee that the execution of real-time tasks can tolerate transient und intermittent faults. The scheme is based on reserving sufficient slack in a schedule such that a task can be re-executed before its deadline without compromising the guaranteed given to other tasks.

Gou, L., T. Hasegawa, P. Luh, S. Tamura, and J. Oblak (1994, October). Holonic Planning and Scheduling for a Robotic Assembly Testbed. In *Proc. of the Rensselear's fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, New York. Rensselear Polytechnique Institute.

Applies the holonic concept to a simple robotic assembly testbed. Establishes cooperation mechanisms for planning and scheduling among holons based on an adaptive consistency algorithm and the Lagrangian relaxation technique.

Gou, L. and P. Luh (1997, June). Holonic Manufacturing Scheduling: Architecture, Cooperation, Mechanism and Implementation. In *Proc. of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Tokyo, Japan.

Models a Holonic Manufacturing System with its key elements such as machines, cells, factories, parts, products, operators, teams, etc. having autonomous and cooperative properties.

Holman, P. and J. Anderson (2001, December). Guaranteeing Pfair Supertasks by Reweighting. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 203–212.

Considers the "supertask" approach, in which a set of Pfair tasks is scheduled as a single task in a multiprocessor system. Presents reweighting rules for both EPDF- and EDF-scheduled component tasks.

Hong, I., M. Potkonjak, and M. Srivastava (1998, November). On-Line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor. In *Proc. of the International Conference on Computer-Aided Design.*

Considers the problem of scheduling hard periodic and firm sporadic tasks on variable voltage processor to optimize power consumption.

Hong, K. and J. Leung (1988, December). On-line Scheduling of real-time tasks. In *Proc. of the Real-Time Systems Symposium*, Huntsville, Alabama, pp. 244–250. IEEE Computer Society Press.

.

Hsueh, C.-W., K.-J. Lin, and N. Fan (1995, December). Distributed Pinwheel Scheduling with End-toEnd Timing Constraints. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 172–181.

Presents an end-to-end scheduling model for real-time distributed system based on the pinwheel scheduling algorithm.

Isovic, D. and G. Fohler (1998). Handling Sporadic Tasks in Off-line Scheduled Distributed Real-Time Systems. In *Proc. of the 11th Euromicro Conference on Real-Time Systems.* IEEE.

Presents an algorithm to handle event-triggered sporadic tasks in the context of time-triggered, off-line scheduled systems. Provides off-line schedulability test for sporadic tasks.

Jonsson, J. (1998). Compile-Time Scheduling of Real-Time Threads on Multi-Level-Context Architectures. In *Proc. of the Seventh Swedish Workshop on Computer System Architecture.*

Addresses the problem of how to schedule periodic, real-time threads on a class of architectures referred to as multi-level-context (MLC) architectures such as real-time operating system architectures.

Kádar, B., L. Monostori, and E. Szelke (1997). An object-oriented framework for developing distributed manufacturing architectures. In L. Monostori (Ed.), *Proc. of the Second World Congress on Intelligent Manufacturing Processes and Systems*, Budapest, Hungary, pp. 548–554.

Kalyanasundaram, B. and K. Pruhs (1995, Oktober). Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science*, Los Alamitos, pp. 214–223. IEEE Computer Society Press.

Introduces the notion of competitive analysis for uniprocessor scheduling, in which the on-line algorithm is allowed more resources than the optimal off-line algorithm.

Koestler, A. (1967). *The Ghost in the Machine.* London: Hutchinson & Co. (Second Edition: Arkana Books, London, 1989)

Observes a dichotomy of wholeness and partness in living organisms and social organisations. Uses the "Janus Effect" as a metaphor for this dichotomy of wholeness and partness. Suggests a new term "holon" to describe the members of these systems.

Kopetz, H. (1997). *Real-Time Systems: Design Principles for Distributed Embedded Applications.* Kluwer Academic Publishers.

This book treats issues of hard real-time distributed systems with fault tolerance aspects.

Koren, G. and D. Sasha (1992, December). D-over: An Optimal On-Line Scheduling Algorithm for Overloaded Real-Time Systems. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona, pp. 290–299. IEEE.

Presents an optimal on-line algorithm for overloaded systems. Optimal means that the algorithm gives the best competitive factor possible relative to an off-line scheduler.

Koren, G. and D. Sasha (1993, November). MOCA: A Multiprocessor On-Line Competitive Algorithm for Real-Time System Scheduling. In *Proc. of the 14th Real-Time Systems Symposium*, pp. 172–181.

Studies on-line scheduling with worst-case guarantees in multi-processor real-time environments. Considers two memory models: a distributed system having a centralized scheduler and a shared memory multiprocessor.

Koren, G. and D. Shasha (1991). An optimal scheduling algorithm with a competitive factor for real-time systems. Technical Report TR572, Department of Computer Science, New York University.

Presents an algorithm for a possibly overloaded system which behaves like the EDF algorithm when the system is underloaded, and obtains at least a quarter of the maximum value that an optimal clairvoyant algorithm could obtain even when the system is overloaded.

Koren, G. and D. Shasha (1995, December). Skip-Over: Algorithms and Complexity for Overloaded Systems That Allow Skips. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 110–117.

Investigates the problem of uniprocessor scheduling of occasionally skippable periodic tasks (tasks with acceptable deadline missing).

Kouiss, K., H. Pierreval, and N. Mebarki (1997). Using multi-agent architecture in FMS for dynamic scheduling. *J. of Intelligent Manufacturing 8*, 41–47.

Presents a multi-agent system for dynamic scheduling in Flexible Manufacturing Systems.

Kriz, D. (1995). *Holonic Manufacturing Systems: Case study of an IMS Consortium.* http://hms.ifw.uni-hannover.de.

Presents motivational and implementation issues for Holonic Manufacturing Systems.

Küster, J. (2001, September). Towards Behavior Consistent Modeling in UML-RT. In *Proc. of the Forum on Design Languages (FDL'01).*

Küster, J. and J. Stroop (2001). Consistent Design of Embedded Real-Time Systems with UML-RT. In *Proc. of the 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001)*, pp. 31–40. IEEE Computer Society.

Küster, J. M. and J. Stroop (2000, October). Towards Consistency of Dynamic Models and Analysis of Timing Constraints. In *Workshop on Formal Design Techniques Using Real-Time UML*, York, UK.

Introduces the notion of consistency for sequence diagrams and state-charts to the domain of real-time modeling using UML/UML-RT.

Lamastra, G., G. Lipari, and L. Abeni (2001, December). A Bandwidth Inheritance Algorithm for Real-Time Task Synchronization in Open Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 151–160.

Presents the BandWidth Inheritance (BWI) scheduling strategy that extends the bandwidth reservation approach to open systems where tasks can interact through shared resources. Off-line sched anal. not possible. Arrival Time unknown.

Larsson, E. (1994). *The Scheduling Tool.* Technical report ProVia-DoCs-94204, Department of Computer Sytems, Uppsala University.

Presents an off-line scheduling tool that maps a set of process graphs onto a particular system configuration. The tool accept as input process graphs of the RED processes to be scheduled, together with a target system description, and produces as output one schedule for each node executing RED processes.

Lawler, E. and C. Martel (1981, Februar). A Note on Preemptive Scheduling of Periodic Real-Time Tasks. *Information Processing Letters 12*(1), 9–12.

Considers a problem in which periodic tasks are to to be preemptively scheduled on a system of parallel processors. Shows that there exists a feasible schedule if and only if there exists a feasible schedule which is cyclic with a period equal in length to the least common multiple of the periods of the individual tasks.

Lee, S., S. Min, C. Kim, C.-G. Lee, and M. Lee (1999). Cache-Conscious Limited Preemptive Scheduling. *Real-Time Systems 17*, 257–282.

Proposes a scheduling scheme, called Limited Preemptive Scheduling, to address the problem of inter-task cache interference due to preemptions in multi-tasking real-time systems.

Lehoczky, J. and S. Ramos-Thuel (1992, December). An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, Phoenix, Arizona, pp. 110–123.

Presents an approach (Slack Stealing) for servicing aperiodic requests within the context of a hard real-time system by making any spare processing time available as soon as possible. A means of determining the maximum amount of slack which may be stolen, without jeopardising the hard timing constraints, is thus the key to the operation of the algorithm.

Leung, J. and M. Merril (1980). A Note on Preemptive Scheduling of Periodic Real-Time Tasks. *Information Processing Letters 11*(3), 115–118.

Leung, J. and J. Whitehead (1982, December). On the complexity of fixed priority scheduling of periodic, real-time tasks. In *Performance Evaluation*, Volume 2 of *4*, Netherlands, pp. 237–250.

Formulates an alternative priority assignment policy, where task deadlines can be less than the period of a task. Provides simple analysis to determine the schedulability of such tasks.

Liu, C. and J. Layland (1973, January). Scheduling algorithms for multiprogramming in a hard real-time environment. *J. of the ACM 20*(1), 46–61.

Describes the different aspects of scheduling of periodic systems. Presents the first proof of the optimality of rate-monotonic and Earliest Deadline First scheduling for periodic systems.

Liu, C., J. Liu, and A. Liestman (1982). Scheduling with Slack-Time. In *Acta Informatica*, Volume 17, pp. 31–41.

Liu, F., P. Luh, and B. Moser (1998). Scheduling and Coordination of Distributed Design Projects. *CIRPS Annals 47*, 111–113.

Studies the scheduling and coordination of distributed design projects with uncertainties while managing design risks. Presents a mathematical optimization model that balances modeling accuracy and computational complexity. Develops a solution methodology that combines Lagrangian relaxation and stochastic dynamic programming.

Liu, J., K.-J. Lin, and S. Natarajan (1987, December). Scheduling Real-Time Periodic Jobs Using Imprecise Results. In *Proc. of the 8th Real-Time Systems Symposium*, San Jose, California, pp. 252–260. IEEE.

Outlines an approach to design general purpose real-time computer systems that can skip non-critical portions of scheduled jobs in order to avoid missed deadline during system overloads.

Liu, J. W. S. and R. Ha (1995). *Advances in Real-Time Systems* (Sang H. Song ed.)., Chapter 9, Efficient Methods of Validating Timing Constraints, pp. 196–220. Prentice Hall.

Presents worst-case bounds and efficient algorithms for determining how late the completion times of independent jobs with arbitrary release times can be in a dynamic multiprocessor or distributed system when their release times and execution times may vary from one instance to another.

Livny, M. and M. Melman (1982, April). Load balancing in homogeneous broadcast distributed systems. In *ACM Computer Network Performance Symposium*, pp. 47–55.

Studies the probability that in a homogeneous distributed computing system, a customer waits for service at one node while at least one node is idle. Shows that for a wide range of system traffic intensity, this probability is close to one.

Lu, C., J. Stankovic, G. Tao, and S. Son (1999, June). Design and Evaluation of a Feedback Control EDF Scheduling Algorithm. In *Proc. of the Real-Time Systems Symposium.*

Presents and evaluates the feedback control real-time scheduling.

Márkus, A., T. Kis, J. Váncza, and L. Monostori (1996). A market approach to holonic manufacturing. *CIRP Annals 45*(1), 433–436.

Introduces a market mechanism for coordinating the activities of intelligent agents that pursue their own interest by operating under bounded rationality in a changing, hardly predictable environment.

Márkus, A. and J. Váncza (1996, September). Are manufacturing agents different? In S. B. S. Albayrak (Ed.), *Proc. of the European Workshop on Agent-Oriented Systems in Manufacturing*, Berlin, Germany, pp. 86–103. Daimler-Benz AG and T.U. Berlin.

outlines a prototype of an order-processing and scheduling system that has been built on a market mechanism.

Marti, P., J. Fuertes, F. Fohler, and K. Ramamritham (2001, December). Jitter Compensation for Real-Time Control Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 39–48.

Proposes an approach for real-time scheduling of control systems by compensating for sampling jitter and sampling-actuation delay through the adjustment of controller parameters.

Martins, E., L. Almeida, and J. Fonseca (2001, December). Integrating Traffic Scheduling and Schedulability Analysis in an FPGA-based Coprocessor. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.

Presents a dedicated coprocessor to support the traffic management in the communication system. The coprocessor integrates both scheduling and schedulability analysis functions.

Moir, M. and S. Ramamurthy (1999, December). Pfair Scheduling of Fixed and Migrating Periodic Tasks on Multiple Resources. In *Proc. of the Real-Time Systems Symposium*, Phoenix, Arizona. IEEE Computer Society Press.

Presents a scheduling scheme of periodic preemptable tasks on multiple resources. Considers a task model that allows arbitrary mixes of fixed and migratable tasks, and prove the existence of an optimal Pfair scheduler in this model.

Mok, A. (1983). *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment.* Ph. D. thesis, Massachusetts Institute of Technology.

(Presents the on-line algorithm Least Laxity Algorithm).

Mok, A. and D. Chen (1997, October). A Multiframe Model for Real-Time Tasks. *IEEE Transactions on Software Engineering 23*(10), 635–645.

Presents a multiframe real-time task model which allows the execution time of a task to vary from one instance to another by specifying the execution time of a task in terms of a sequence of numbers.

Mok, A. and M. Dertouzos (1978, November). Multiprocessor Scheduling in a Hard Real-Time Environment. In *Proc. of the 7th Texas Conference on Computer Systems*, Houston, Texas. IEEE/ACM.

Mok, A. and W. Wang (2001, December). Window-Constrained Real-Time Periodic Task Scheduling. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 15–24.

Shows that the Dynamic Window-Constrained Scheduling fails for arbitrarily low aggregate utilization rates of the packet streams. Defines the notion of Pfairness in relation to the Window-Contrained Scheduling model. Defines an EDF for the Window-Contrained scheduling problem.

Moriwaki, T., N. Sugimura, Y. Martawirya, and S. Wirjomartono (ASME 1992). Production scheduling in autonomous distributed manufacturing system. *Quality Assurance Through Integration of Manufacturing Processes and Systems PED-Vol. 56*, 175–186.

Nishi, T., A. Sakata, S. Hasebe, and I. Hashimoto (2000). Autonomous Decentralized Scheduling System for Just-in-Time Production. In *Proc. of the 7th International Symposium on Process System Engineering*, pp. 345–351.

Proposes an autonomous decentralized scheduling system for just-in-time production. The goal for each sub-system scheduling includes the storage costs for intermediate and final products in addition to the changeover costs and the due date penalties.

Nissanke, N., A. Leulseged, and S. Chillara (2001, December). A Framework for Probabilistic Analysis of Multiprocessor Scheduling Environments. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Sets up a framework for probabilistic analysis of task scheduling in a multiprocessor environment. Assumes that the tasks are not known precisely and that they inherently carry an element of uncertainty or unpredictability.

Oh, Y. and S. Son (1994). Scheduling hard real-time tasks with tolerance of multiple processor failures. *Microprocessing and Microprogramming 40*, 193–206.
Studies the problem of scheduling a set of hard real-time tasks with duplication. Proves that the problem of scheduling a set of non-preemptive tasks on $m \geq 3$ processors to tolerate one arbitrary processor failure is *NP-complete* even when the tasks share a common deadline. A heuritic algorithm is proposed to solve the problem.

Oliveira, R. and J. Fraga (1996). Scheduling Imprecise Computation Tasks with Intra-Task / Inter-Task Dependence. In *Proc. of the 21st IFAC/IFIP Workshop on Real-Time Programming*, pp. 51–56.
Presents two heuristics to be used as admission policy when the system is made of imprecise tasks. The objective of the admission policy is to maximize system utility through the selection of optional parts for execution. These heuristics are supposed to be used combined with off-line schedulability tests and on-line acceptance tests already described in the literature.

Orozco, J., R. Santos, J. Santos, and E. Ferro (2001, December). Hybrid Rate-Monotonic/Reward-Based Scheduling of Real-Time Embedded Sytems. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Studies the on-line scheduling of periodic, independent, preemptable real-time sets of tasks consisting of a mandatory and an optional part. Presents a scheduling method allowing a rearrangement of slack time based on the detection of singular instants during the processing.

Padreiras, P. and L. Almeida (2001, December). A Practical Approach to EDF Scheduling on Controller Area Network. In *Proc. of the IEEE/EE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*, London.
Compares Earliest Deadline First algorithm with the rate monotonic approach on a FTT CAN protocol. Concludes that using EDF, a higher bus

utilization and a reduced jitter and delay time for messages with long periods are achieved.

Parunak, H., A. Baker, and S. Clark (1997, February). The AARIA Agent Architecture. In *Proc. of the International Conference on Autonomous Agents*, Marina del Rey, CA.

Presents an architecture for a full Enterprise Resource Planning anf Manufacturing Execution System.

Peng, D.-T., K. Shin, and T. Abdelzaher (1997, December). Assignment and Scheduling Communicating Periodic Tasks in Distributed Real-Time Systems. *IEEE Transactions on Software Engineering 32*(12), 745–758.

Presents an optimal solution to the problem of allocating communicating periodic tasks in a distributed real-time systems. The task system is modeled with a task graph and are assigned to processing nodes by using a branch & bound search algorithm.

Philips, C. A., C. Stein, E. Torng, and J. Wein (1997, May). Optimal Time-Critical Scheduling via Resource Augmentation. In *Proc. of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, Texas, pp. 140–149.

Considers the problem real-time dynamic scheduling in a preemptive multiprocessor setting, and scheduling to provide good response time in a number of scheduling environments. Uses a relaxed notion of competitive analysis, in which the on-line algorithm is allowed more resources than the optimal off-line algorithm to which it is compared.

Poledna, S., A. Burns, A. Wellings, and P. Barrett (2000, February). Replica Determinism and Flexible Scheduling in Hard Real-Time Dependable Systems. *IEEE Transactions on Computers 49*(2), 100–111.

A method, called timed messages, to avoid the inconsistent order and timing of replicated tasks in real-time distributed systems is presented. The major advantage of timed messages is its efficiency and flexibility while guaranteeing deterministic operation of replicated tasks.

Ramamritham, K. and J. Stankovic (1984). Dynamic Task Scheduling in Hard Real-Time Distributed systems. *IEEE Software 1*(3), 65–75.

Deals with multiprocessor scheduling in hard real-time distributed systems. Uses a uniprocessor scheme for local scheduling, and perform distributed scheduling for tasks which are potentially subject to timing failures at run-time.

Ramamritham, K. and J. Stankovic (1994). Scheduling Algorithms and Operating Systems Support for Real-Time Systems. In *Proc. of the IEEE*, Volume 82 of *1*, pp. 55–67.

Presents the state of the real-time field in the areas of scheduling and operating system kernels.

Ramamritham, K., J. Stankovic, and P.-F. Shiah (1990, April). Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems. *IEEE Transactions on Parallel and Distributed Computing 1*(2), 184–194.

Presents a scheduling algorithm, based on heuristics, to schedule a set of tasks in multiprocessor systems. The approach constructs explicitly task execution plans. It assumes that non-blocking tasks with known worst case execution times and resource requirements are the entities being scheduled.

Ramamritham, K., J. Stankovic, and W. Zhao (1989, August). Distributed Scheduling of Tasks with Deadlines and Resource Requirements. *IEEE Transactions on Computers 38*(8), 1110–1123.

Evaluates four algorithms for cooperation in a distributed real-time system. They differ in the way a node treats a task that cannot be guaranteed locally.

Ramos, C. (1996). A Holonic Approach for Task Scheduling in Manufacturing Systems. In *Proc. of the IEEE International Conference on Robotics and Automation.*

Ramos, C. and P. Sousa (1996, September). Scheduling Orders in Manufacturing Systems using a Holonic Approach. In S. B. S. Albayrak (Ed.), *Proc. of the European Workshop on Agent-Oriented Systems in Manufacturing*, Berlin, Germany, pp. 80–85. Daimler-Benz AG and T.U. Berlin.

Presents an approach to resource allocation in holonic manufacturing based on the contract net protocol.

Randell, B., J.-C. Laprie, H. Kopetz, and B. Littlewood (1995). *Predictably Dependable Computing Systems.* Springer-Verlag.

Contains a selection of papers on main topics in Predictable Dependable Computing Systems: fault prevention, fault tolerance, fault removal, and fault forecasting.

Regehr, J. and J. Stankovic (2001, December). A Framework for Composing Soft Real-Time Schedulers. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 3–14.

Presents a hierarchical CPU scheduling to support applications in open systems. .

Rhee, I. and G. Martin (1995, December). A Scalable Real-Time Synchronization Protocol for Distributed Systems. In *Proc. of the Real-Time Systems Symposium*, Pisa, Italy, pp. 21–27.

Proposes a distributed protocol for the synchronization of real-time tasks that have variable resource requirements. The protocol is intended for large-scale distributed or parallel systems in which processes communicate by message passing.

Ripoll, I., A. Crespo, and A. Garcia-Fornes (1997, June). An Optimal Algorithm for Scheduling Soft Aperiodic Tasks in Dynamic-Priority Preemptive Systems. *IEEE Transactions on Software Engineering 23*(6), 388–400.

Presents the theoretical foundations for the EDF Exact Slack Stealer algorithm which provides a solution to the problem of jointly scheduling hard periodic tasks and aperiodic tasks. Is optimal for periodic tasks since it is based on EDF and optimal for aperiodic tasks since it gives the shortest response time to aperiodic tasks.

Rumbaugh, J., M. Blaha, W. Premerlani, S. Eddy, and W. Lorensen (1991). *Object Oriented Modeling and Design.* New York, USA: Prentice Hall, Englewood Cliffs.

Explores the Object Modeling Technique (OMT), a generic method of representing objects and their relationships.

Santos, J., E. Ferro, J. Orozco, and R. Cayssials (1997). A Heuristic Approach to the Multitask-Multiprocessor Assignment Problem using the Empty Slots Method and Rate Monotonic Scheduling. *Real-Time Systems 13*, 167–199.

Presents a heuristic approach to the problem of assigning a set of preemptable resource-sharing and blockable real-time tasks to be executed in a set of heterogeneous processors communicated through an interprocessor network.

Schneider, U. (1996). *Ein formales Modell und eine Klassifikation für die Fertigungssteuerung.* Ph. D. thesis, Heinz-Nixdorf Institut / Universität Paderborn.

Presents a methodology for the construction of production planning and control systems based on a classification of production control tasks and production control procedures.

Selic, B. (1998). Using UML for modelling complex real-time systems. *Lecture Notes in Computer Science 1474*, 250–260.

Selic, B. (1999, October). Turning Clockwise: Using UML in the Real-

Time Domain. *Communications of the ACM 42*(10), 46–54.
Discusses modeling of real-time systems using UML.

Serlin, O. (1972, May). Scheduling of Time Critical Processes. In *Proc. of the Spring Joint Computer Conference*, Atlantic city, New Jersey, pp. 925–932. Montvale, NJ: American Federation of Information Processing Societies.
(First algorithm to handle periodic tasks.).

Sha, L. and S. Sathaye (1993). Distributed real-time system design: Theoretical concepts and applications. Technical Report CMU/SEI-93-TR-2 ESC-TR-93-179, Software Engineering Institute, Canergie Mellon University.
Describes the use of generalized rate monotonic scheduling theory for the design and analysis of a distributed real-time system.

Shih, W.-K., J. Liu, J.-Y. Chung, and D. Gillies (1989, July). Scheduling Tasks with Ready Times and Deadlines to Minimize Average Error. *Operating Systems Review 23*(3), 14–28.
Presents a preemptive optimal algorithm for scheduling n dependent tasks with rational ready times, deadlines, and processing times on uniprocessor systems. The tasks are logically decomposed into mandatory and optional subtasks.

Shin, K. G. and Y.-C. Chang (1995, December). A reservation-based algorithm for scheduling both periodic and aperiodic real-time tasks. *IEEE Transactions on Computers 44*, 1405–1419.

Shirazi, B., A. Hurson, and K. Kavi (1995). *Scheduling and Load Balancing in Parallel and Distributed Systems.* New York: IEEE Computer Society Press.

Smith, R. (1980, December). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers C-29*(12), 1104–1113.
Pioneers the research in communication among cooperating distributed agents with the contract net protocol.

Sousa, P. and C. Ramos (1997). A Dynamic Scheduling Holon for Manufacturing Orders. In L. Monostori (Ed.), *Proc. of the Second World Congress on Intelligent Manufacturing Processes and Systems*, Budapest, Hungary, pp. 542–547.

Sprunt, B. (1990, August). *Aperiodic Task Scheduling for Real-Time Systems.* Ph. D. thesis, Department of Electrical and Computer Engineering, Canergie Mellon University.

Develops the Sporadic Server Algorithm for scheduling aperiodic tasks in real-time systems. Demonstrates that the Sporadic Server algorithm is able to guarantee deadlines for hard-deadline aperiodic tasks and provides good responsiveness for soft-deadline tasks.

Sprunt, B., J. Lehoczky, and L. Sha (1988, December). Exploiting Unused Periodic Time For Aperiodic Service Using the Extended Priority Exchange Algorithm. In *Proc. of IEEE Real-Time Systems Symposium*, pp. 251–258.

Presents an extended version of the priority exchange server algorithm for exploiting unused periodic time.

Sprunt, B., L. Sha, and J. Lehoczky (1989). Aperiodic Task Scheduling for Hard Real-Time Systems. In *J. of Real-Time Systems*, pp. 27–60.

Develops the Sporadic Server algorithm for scheduling aperiodic tasks in real-time systems. This algorithm extends the rate monotonic algorithm which was designed to schedule periodic tasks. It guarantees deadlines for hard periodic tasks and provide good responsiveness for soft aperiodic tasks.

Spuri, M. and G. Buttazzo (1994, December). Efficient Aperiodic Service under Earliest Deadline Scheduling. In *Proc. of the 15th IEEE Real-Time Systems Symposium*, Portorico, pp. 2–11.

Proposes and studies server algorithms under earliest deadline first scheduling. Studies the performance of the Improved Priority Exchange algorithm.

Spuri, M. and G. Buttazzo (1996). Scheduling aperiodic tasks in dynamic priority systems. *Real-Time Systems Journal 10*, 179–210.

Spuri, M., G. Buttazzo, and F. Sensini (1995, December). Robust Aperiodic Scheduling under Dynamic Priority Systems. In *Proc. of the IEEE Real-Time Systems Symposium*, Pisa , Italy, pp. 210–219.

Extends the Total Bandwidth Server algorithm to handle firm aperiodic tasks and then integrates a guarantee mechanism that allows to achieve degradation in case of transient overload.

Stankovic, J. (1984, June). Simulation of Three Adaptive Decentralized Controlled Job Scheduling Algorithms. *Computer Network 8*(3), 199–217.

Stankovic, J. (1985a, February). An application of bayesian decision theory to decentralized control of job scheduling. *IEEE Transactions on Computers C-34*(2), 117–130.

The delay in transferring state information and tasks makes it impossible for a node scheduler to obtain the necessary data to take an optimal decision. The Bayesian decision based algorithm tries to reduce uncertainty through estimates based on information provided by the exchange of messages.

Stankovic, J. (1985b). Stability and Distributed Scheduling Algorithms. *IEEE Transactions on Software Engineering SE-11*(10), 1141–1152.
Lists two scheduling methods. The first is adaptive with dynamic reassignment, and is based on broadcast messages and stochastic learning automata. The second method uses bidding and one time-assignment in a real-time environment.

Stankovic, J. (1988, October). A Serious Problem for Next-Generation Systems. *Computer 21*(10), 10–19.

Stankovic, J., T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu (2001, December). Feedback Control Scheduling in Distributed Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 59–70.
Presents a framework (Distributed Feedback Control Real-Time Scheduling) for developing software control algorithms based on a theory of feedback control to a distributed open system.

Stankovic, J., C. Lu, S. Son, and G. Tao (1999, June). The Case for Feedback Control Real-Time Scheduling. In *Proc. of the Euromicro Conference on Real-Time Systems.*
Studies the use of feedback control concepts in soft rea-time scheduling systems, with the goal of the development of a theory of feedback control scheduling.

Stankovic, J. and R. Mirchandaney (1986, December). Using stochastic learning automata for job scheduling in distributed processing systems. *J. of Parallel and Distributed Computing 3*, 527–552.

Stankovic, J. and K. Ramamritham (1988a). *Tutorial Hard Real-Time Systems.* IEEE Computer Society Press.

Stankovic, J. and K. Ramamritham (1988b). *Tutorial Hard Real-Time Systems* (J. A. Stankovic and K. Ramamritham ed.)., Paper: Scheduling Algorithms for Hard Real-Time Sytems - A Brief Survey, pp. 150–173. IEEE Computer Society Press.

Stankovic, J. and K. Ramamritham (1988c). *Tutorial Hard Real-Time Systems*, Paper: Evaluation of a Flexible Task Scheduling for Distributed Hard Real-Time Systems, pp. 273–286. IEEE Computer Society Press.

Stankovic, J. and K. Ramamritham (1991, May). The Spring Kernel: A New Paradigm for Real-Time Systems. *IEEE Software 8*(3), 62–72.
Implements and evaluates multiprocessor schedulers running on single, dedicated nodes of small scale parallel embedded systems using a static heuristic function, which may integrate timing, resource and preceding constraints.

Stankovic, J. and K. Ramamritham (1993). *Advances in Real-Time Systems*, Article: Misconceptions About Real-Time Computing, pp. 17–25. IEEE Computer Society Press.

Stewart, D. and P. Khosla (1991, May). Real-Time Scheduling of Sensor-Based Control Systems. In *Proc. of the Eighth IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, pp. 144–150.
Proposes the maximum urgency first algorithm, which is a mixed priority real-time scheduling algorithm (combination of fixed and dynamic priority scheduling). The motivation behind this algorithm is to provide guaranteed soft real-time scheduling.

Streich, H. (1995). TaskPair-Scheduling: An Approach for Dynamic Real-Time Systems. *Int. Journal of Mini & Microcomputers 17*(2), 77–83.
Presents an on-line scheduling approach which merges the concepts of guaranteeing (an activity) and exception handling (due to time outs). It gives a guarantee, when the task is accepted, that the TaskPair will hold its time constraints.

Strosnider, J. (1988, August). *Highly Responsive Real-Time Token Rings.* Ph. D. thesis, Department of Electrical and Computer Engineering, Canergie Mellon University.
Develops the deferrable server algorithm. The execution of the deferrable server is replenished periodically. Unlike a poller, when a deferrable server finds no aperiodic jobs for execution, it preserves its budget.

Sun, J. and J. Liu (1996, May). Synchronization Protocols in Distributed Real-Time Systems. In *Proc. of the 16th International Conference on Distributed Computing Systems*, pp. 38–45.
Focuses on distributed real-time systems that contain independent, periodic tasks scheduled by fixed priority scheduling algorithms. Describes three synchronization protocols together with algorithms to analyze the schedulability of the system when these protocols are used.

Sun, J. and M. G. J. Liu (1997, October). Bounding Completion Times of Jobs with Arbitrary Release Times, Variable Execution Times, and Resource Sharing. *IEEE Transactions on Software Engineer-*

ing 23(10), 603–615.

Presents three algorithms for computing upper bounds on the completion times of jobs that have arbitrary release times and priorities.

Swaminathan, V. and K. Chakrabarty (2000, November). Real-Time Task Scheduling for Energy-Aware Embedded Systems. In *Proc. of the IEEE Real-Time Systems Symposium.*

Presents an approach for scheduling periodic tasks in real-time systems. The presented approach minimizes the total energy consumed by the task set and guarantees that the deadline for every periodic task is met.

Tia, T.-S. and J.-S. Liu (1995). Assigning Real-Time Tasks and Resources to Distributed Systems. *Special Issue of the International Journal of Mini and Microcomputers 17*(1), 18–25.

Presents a method for allocating periodic tasks where different tasks may have different deadlines. Graph based heuristics, which attempt to minimize interprocess communication and based on clustering and graph-bisection, are used for task assignment.

Tindell, K. and J. Clark (1994). Holistic Schedulability Analysis for Distributed Hard Real-Time Systems. *Microprocessing & Microprogramming 40*, 117–134.

Performs an early work on an event-driven model for scheduling of distributed systems. Analyses schedulability for distributed systems where tasks with arbitrary deadlines communicate by message passing and shared data areas. Uses periodic tasks for the first task in the transaction. Subsequent tasks are triggered as sporadic tasks when the preceding task has been completed.

Tokuda, H. and C. Mercer (1989, July). Arts: A Distributed Real-Time Kernel. In *Operating Systems Review*, Volume 23 of *3*, pp. 29–53. ACM Press.

Introduces a real-time object model and the integrated time-driven scheduling model to develop real-time computing systems in a distributed environment. Describes the Advanced Real-Time Technology (ARTS) kernel and the real-time toolset consisting of schedulability analyzer, *Scheduler 1-2-3*, and the real-time monitor/debugger.

Tokuda, H., J. Wendorf, and H. Wang (1987, December). Implementation of a Time-Driven Scheduler for Real-Time Operating Systems. In *Proc. of the 8th Real-Time Systems Symposium*, San Jose, California, pp. 271–280. IEEE.

Tönshoff, H. and M. Winkler (1995). Shop Control for Holonic Manufacturing Systems. In *Proc. of the 27th CIRP International Seminar on Manufacturing Systems*, Michigan, USA, pp. 329–336. Ann-Arbor.

Valckenaers, P., H. V. Brussel, L. Bongaerts, and J. Wyns (1997). Holonic Manufacturing Sytems. *Integrated Computer-aided Engineering 4*(3), 191–201.

Valckenaers, P., J. Wyns, H. V. Brussel, L. Bongaerts, and P. Peeters (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry, Special Issue on Intelligent Manufacturing Systems 37*(3), 255–276.

Presents the PROSA reference architecture for holonic Manufacturing Systems. This architecture consists of three types of holons: product, resource and order with the assistance of staff holons.

Váncza, J. and A. Márkus (1998). Holonic manufacturing with economic rationality. In E. W. G. on IMS & EPFL (Ed.), *Proc. of the European Workshop on Intelligent Manufacturing Systems (IMS-EUROPE-98)*, Lausanne, Switzerland, pp. 383–394.

Wellings, A., L. Beus-Dukic, and D. Powell (1998, December). Real-Time Scheduling in a Generic Fault-Tolerant Architecture. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain.

Presents a real-time scheduling for Generic Upgradable Architecture for Real-Time Dependable Systems (GUARDS). Uses an extended response-time analysis to predict the timing properties of replicated real-time transactions.

Young, M. and L.-C. Shu (1991). Hybrid online/offline scheduling for hard real-time systems. Technical Report SERC-TR-100-P, Software Engineering Research Center, Department of Computer Sciences, Purdue University.

Constructs an off-line scheduler that optimally allocates idle time to improve rate-monotonic schedulability.

Young, S. (1982). *Real-Time Languages: Design and Development.* Ellis Horwood.

Zhao, W., K. Ramamritham, and J. Stankovic (1987a, August). Preemptive Scheduling Under Time and Resource Constraints. *IEEE Transactions on Computers C-36*(8), 949–960.

Discusses the use of a heuristic approach for scheduling atsks with timing and resource constraints. Validates the results through simulation studies.

Zhao, W., K. Ramamritham, and J. Stankovic (1987b, May). Scheduling Tasks with Resource Requirements in Hard Real-Time Systems. *IEEE*

Transactions on Software Engineering SE-13(5), 564–577.

Zhu, D., R. Melhem, and B. Childers (2001, December). Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multi-Processor Real-Time Systems. In *Proceedings of the Real-Time Systems Symposium*, London, pp. 84–94.
Focuses on power-aware scheduling for multi-processor real-time systems based on the idea of slack sharing among processors. Takes into consideration tasks with and without precedence constraints.