RESOURCE AUGMENTATION FOR PERFORMANCE GUARANTEES IN EMBEDDED REAL-TIME SYSTEMS

Abhilash Thekkilakattil

2012
Abstract

Real-time scheduling policies have been widely studied, with many known schedulability and feasibility analysis techniques for different task models, that have advanced the state-of-the-art. Most of these techniques are typically derived under the assumption of negligible runtime overheads which may not be realistic for modern embedded real-time systems, and hence potentially compromises the guarantees on their correct behaviors. This calls for methods to reason about the functioning of the system under the presence of such overheads as well as to predictably control them. Controlling these overheads may place additional performance demands, consequently requiring more resources such as faster processors. At the same time, the need for energy efficiency in these class of systems further complicates the problem and necessitates a holistic approach.

In this thesis, we apply resource augmentation, viz., processor speed-up, to guarantee desired real-time properties even under the presence of runtime overheads. We specifically consider preemptions and faults that, at runtime, manifest as overheads in the system in various ways. Our aim is to provide specified non-preemption and fault tolerance feasibility guarantees in a real-time system. We first propose offline and online methods, that uses CPU frequency scaling, to control the number of preemptions in periodic and sporadic task systems, under a preemptive Fixed Priority Scheduling (FPS) policy. Furthermore, we derive the resource augmentation bound, specifically the upper-bound on the lowest processor speed, that guarantees the feasibility of a specified non-preemption behavior for any real-time task. We show that, for any task \( \tau_i \), the resource augmentation bound that guarantees a non-preemptive execution for a specified duration \( L_i \), is given by \( \frac{4L_i}{D_{\min}} \), where \( D_{\min} \) is the shortest deadline in the task set. Consequently, we show that the upper-bound on the lowest processor speed that guarantees the feasibility of a non-preemptive schedule for the task set is \( \frac{4C_{\max}}{D_{\min}} \), where \( C_{\max} \) is the largest execution time in the task set.
We then propose a method to guarantee specified upper-bounds on the preemption related overheads in the schedule. We first translate the requirements of meeting specified upper-bounds on the preemption related overheads to a set of non-preemption requirements for the task set. The resource augmentation bound in conjunction with a sensitivity analysis is used to calculate the optimal processor speed that guarantees the derived non-preemption requirements, achieving the specified bounds on the preemption related costs. Finally, we derive the resource augmentation bound that guarantees the fault tolerance feasibility of a set of real-time tasks under an error burst of known length. We show that if the error burst length is no longer than half the shortest deadline in the task set, the resource augmentation bound that guarantees fault tolerance feasibility is 6.

Our contribution bounds the extra resources, specifically the required processor speed-up, that provides specified non-preemption and fault tolerance feasibility guarantees in a real-time system. It allows us to quantify the ‘goodness’ of non-preemptive scheduling, referred to as its sub-optimality, as compared to an optimal uni-processor scheduling algorithm, in terms of the required processor speed-up that guarantees a non-preemptive schedule for any uni-processor feasible task set. We intend to extend this work to provide non-preemption and fault tolerance feasibility guarantees in multi-processor systems.
“Lokah Samastah Sukhino Bhavantu.”

"May all beings everywhere be happy and free, and may the thoughts, words, and actions of my own life contribute in some way to that happiness and to that freedom for all.”

---

This Sanskrit verse is an expression of the universal spirit found in the ancient Indian scriptures of Vedas.
Acknowledgements

It is hard to express in words my gratitude towards several people who have contributed directly or indirectly in enriching my life. However, today supposedly being an important day in my academic career, I take a moment to thank them for their kindness, love and affection.

Let me first thank my supervisors, Prof. Sasikumar Punnekkat and Dr. Radu Dobrin, for believing in me, and for their continuous support throughout my bachelors, masters and now the doctoral studies. Prof. Sasi has always supported me and motivated me to continue exploring my ways even when I was doubtful about them. Radu has been a great teacher - he taught me great many 'other' things, besides real-time systems, during the last four years, and we have had a wonderful time together during the many conference trips (I still remember the hunt for the hot peanuts in Hong Kong). Even though we rarely have a consensus regarding any matter, you have always been a role model to me. Let me also thank Vipin Sir for pushing me to pursue higher studies and for being a good teacher and friend, and for patiently listening to me whenever I have something to talk.

Next, I would like to thank my friends Vidhi and Deepthi for being partners in crime all these years and for the wonderful time that we have had together. Many thanks goes to Vidhi, Sebastian, Deepthi, Uma Chechi and Sajith for putting up with the several quick unplanned Amsterdam trips. I really miss you guys. Life would have been hard without the once in awhile 'chumma pingals' from Vidhi and the nice books and the long philosophical discussions from Sebastian. I thank Jezy chechi for motivating me to continue my higher studies at MDH and Paru for being a good friend and for cheering me up quite

---

A licentiate degree is a Swedish graduate degree halfway between MSc and PhD.
My office-mates Adnan, Andreas J, Joe and Hus deserves special mention for the nice time that we have. Furthermore, many thanks goes to Gabriel, Jey, Yue, Andreas G, Fredrik E, Mikael À, Jan C, Rafia, Saad, Guillermo, Svetlana, Moris, Leo, Aida, Hang, Thomas N, Irfan, Omar, Farhang, Nima, Eddy, Cristina, Dag and Barbara for the company during the lunch and/or conference trips. Special thanks to Adnan for helping me out with many practicalities and for being a good friend.

The learning process would have been incomplete without the nice Professors at IDT, especially Hans, Ivica, Damir, Gordana, Kristina and Bjorn. Heartfelt gratitude to Susanne, Carola and the other admin staff for the smooth and 'real-time' handling of the day-to-day practicalities. I also thank the other colleagues at IDT for lighting up the otherwise monotonous (winter) days.

I cannot leave out the football gang: Radu, Nikola, Adis, Murat, Fredrik A, Muhammed, Abdi, Nicho, Irfan, Omar, Alex, Simon, Pedram, Behnam, Federico, Fredrik E, Eddy, Dennis and the other irregulars, for making weekends so lively and fun- thanks to Radu and Nikola for organizing it every week. Also the badminton - pingpong gang: Svetlana, Saad, Gabriel, Dag, Severine, Irfan, Muhammed A, Andreas G., Bob and Leo, for making wednesdays seem like saturdays- thanks to Saad and Svetlana for organizing it.

A special word of thanks to Prof. Sasi, Sunitha Chechi and Kannan, for making me feel at home with the nice company and delicious food and also to Prakash Chettan, Mridula Chechi and little Mira for the good times.

Finally, I thank my parents and my sister for the love and affection and for the continuous support without which this thesis would have been impossible.

Abhi,
Västerås, November, 2012

The work presented in this thesis was partially supported by Vetenskapsrådet (The Swedish Research Council) under the project CONTESSE (2010-4276).
Publications

List of Papers Included in this Thesis¹

**Paper A:** *Reducing the Number of Preemptions in Real-Time Systems Scheduling by CPU Frequency Scaling*, Abhilash Thekkilakattil, Anju S Pillai, Radu Dobrin and Sasikumar Punnekkat, In proceedings of the 18th International Conference on Real-Time and Network Systems, Toulouse, France, November, 2010

**Paper B:** *Probabilistic Preemption Control using Frequency Scaling for Sporadic Real-time Tasks*, Abhilash Thekkilakattil, Radu Dobrin and Sasikumar Punnekkat, In proceedings of the 7th International Symposium on Industrial Embedded Systems, IEEE, Karlsruhe, Germany, June, 2012

**Paper C:** *Quantifying the Sub-Optimality of Non-Preemptive Real-time Scheduling*, Abhilash Thekkilakattil, Radu Dobrin and Sasikumar Punnekkat, Technical Report, Mälardalen Real Time Research Centre, Mälardalen University, Västerås, Sweden, November, 2012

**Paper D:** *Resource Augmentation for Fault-Tolerance Feasibility of Real-time Tasks under Error Bursts*, Abhilash Thekkilakattil, Radu Dobrin, Sasikumar Punnekkat and Huseyin Aysan, In proceedings of the 20th International Conference on Real-Time and Network Systems, ACM, Pont á Mousson, France, November, 2012 (*Shortlisted for Best Student Paper Award*)

¹The included articles are reformatted to comply with the licentiate thesis guidelines.
Other Relevant Papers


5. *Optimizing the Fault Tolerance Capabilities of Distributed Real-Time Systems*, Abhilash Thekkilakattil, Radu Dobrin, Sasikumar Punnekkat and Huseyin Aysan, In proceedings of the 14th International Conference on Emerging Technologies and Factory Automation (WiP), IEEE, Palma de Mallorca, Spain, September, 2009
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Real-time task attributes</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>The dependability tree by Laprie et. al. [1]</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>(A) Preemptive real-time schedule (B) Non-preemptive real-time schedule</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Fault tolerance related overheads for systems that employ temporal redundancy for fault tolerance</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>An offline detected initial preemption</td>
<td>39</td>
</tr>
<tr>
<td>5.2</td>
<td>An off-line detected potential preemption</td>
<td>39</td>
</tr>
<tr>
<td>5.3</td>
<td>Original RM schedule</td>
<td>45</td>
</tr>
<tr>
<td>5.4</td>
<td>RM schedule after eliminating one preemption</td>
<td>47</td>
</tr>
<tr>
<td>5.5</td>
<td>RM schedule after reducing preemptions using LOPF</td>
<td>47</td>
</tr>
<tr>
<td>5.6</td>
<td>Average number of initial preemptions after preemption elimination</td>
<td>49</td>
</tr>
<tr>
<td>5.7</td>
<td>Average number of potential preemptions after preemption elimination</td>
<td>49</td>
</tr>
<tr>
<td>6.1</td>
<td>Example probability mass function</td>
<td>71</td>
</tr>
<tr>
<td>6.2</td>
<td>A part of the original FPS schedule of the sporadic task set</td>
<td>71</td>
</tr>
<tr>
<td>6.3</td>
<td>The sporadic task schedule after preemption control</td>
<td>72</td>
</tr>
<tr>
<td>6.4</td>
<td>Average number of preemptions for various threshold probabilities</td>
<td>73</td>
</tr>
<tr>
<td>6.5</td>
<td>Average power consumption for various threshold probabilities</td>
<td>74</td>
</tr>
<tr>
<td>7.1</td>
<td>The feasibility bucket</td>
<td>88</td>
</tr>
<tr>
<td>7.2</td>
<td>Methodology overview</td>
<td>95</td>
</tr>
</tbody>
</table>
7.3 Non-preemption requirement to enable preemptions only at optimal preemption points .................................................. 97
7.4 Non-preemption requirement to always enable critical section execution inside non-preemptive regions ......................... 98
8.1 The worst case error overhead due to error bursts on a single task 124
8.2 The maximum length of the burst error ........................................ 125
8.3 Error burst hitting multiple jobs .............................................. 127
8.4 EDF schedule .................................................................. 136
8.5 EDF schedule under faults with $T_{\text{length}} = 4$ ...................... 136
8.6 EDF schedule under faults after a speed-up of 2.8 ................... 137
## Contents

I Thesis ........................................... 1

1 Introduction ........................................... 3
   1.1 Real-time Systems ........................................... 3
   1.2 Real-time Scheduling ........................................... 4
   1.3 Energy Awareness in Real-time Systems ...................... 6
   1.4 Dependability in Real-time Systems ......................... 6
   1.5 Resource Augmentation ........................................... 8

2 Motivation and Problem Description .................. 9
   2.1 Motivation .................................................. 9
   2.2 Problem Description ......................................... 12

3 Main Contributions ........................................... 13
   3.1 Summary of Contributions ........................................... 13
       3.1.1 Paper A .................................................. 14
       3.1.2 Paper B .................................................. 14
       3.1.3 Paper C .................................................. 15
       3.1.4 Paper D .................................................. 17
   3.2 Significance of the Contributions .......................... 18

4 Conclusions and Future Work .......................... 21
   4.1 Conclusions .................................................. 21
   4.2 Future Work .................................................. 22

Bibliography ........................................... 25
## II Included Papers

### 5 Paper A: Reducing the Number of Preemptions in Real-Time Systems Scheduling by CPU Frequency Scaling

- **5.1 Introduction** .................................................. 33
- **5.2 Related Work** .................................................. 34
- **5.3 System Model** .................................................. 36
  - 5.3.1 Task Model .................................................. 36
  - 5.3.2 Energy Model ................................................. 37
  - 5.3.3 Execution Time Model ....................................... 37
- **5.4 Methodology** .................................................. 38
  - 5.4.1 Preemption Identification .................................. 40
  - 5.4.2 Preemption Elimination .................................... 40
- **5.5 Example** ....................................................... 44
- **5.6 Performance Evaluation** .................................... 47
  - 5.6.1 Experiment 1 ................................................ 48
  - 5.6.2 Experiment 2 ................................................ 48
  - 5.6.3 Energy Consumption ....................................... 50
- **5.7 Discussion** .................................................... 50
- **5.8 Conclusions and Future Work** ............................ 51
- **5.9 Acknowledgment** ............................................. 52

### Bibliography ....................................................... 53

### 6 Paper B: Probabilistic Preemption Control using Frequency Scaling for Sporadic Real-time Tasks

- **6.1 Introduction** .................................................. 59
- **6.2 Related Work** .................................................. 61
- **6.3 System Model** .................................................. 63
  - 6.3.1 Processor Model ............................................. 63
  - 6.3.2 Task model .................................................. 63
  - 6.3.3 Energy Model ................................................ 64
  - 6.3.4 Execution Time Model ..................................... 64
- **6.4 Methodology** .................................................. 65
  - 6.4.1 Offline Phase ................................................ 65
  - 6.4.2 Online Preemption Control Algorithm .................... 66
- **6.5 Example** ....................................................... 70
- **6.6 Evaluation** .................................................... 72
## Contents

6.7 Conclusions ................................................. 73
Bibliography .................................................. 75

7 Paper C: 
Quantifying the Sub-Optimality of Non-Preemptive Real-time Scheduling 79

7.1 Introduction .............................................. 81
7.2 System Model ............................................. 83
  7.2.1 Task model ........................................... 83
  7.2.2 Scheduling Model .................................... 84
  7.2.3 Execution Time Model .............................. 85
7.3 Feasibility Analysis of Real-time Systems ................... 85
7.4 Quantifying the Sub-Optimality of Non-Preemptive Scheduling .......... 87
7.5 Guaranteeing a Specified Preemption Behavior using Processor Speed-up ......................... 94
  7.5.1 Translating Preemption Cost Control Requirements to Non-Preemption Requirements ............... 96
  7.5.2 Sensitivity Analysis for Preemption Control ............... 99
  7.5.3 Example ............................................... 101
7.6 Discussions ............................................... 102
  7.6.1 Relaxing the Assumption of Linear Speed-up .......... 102
  7.6.2 Relaxing the Assumption of Negligible Preemption Related Overheads at Optimal Preemption Points 105
7.7 Related Work ............................................. 105
7.8 Conclusions .............................................. 108
Bibliography .................................................. 111

8 Paper D: 
Resource Augmentation for Fault-Tolerance Feasibility of Real-time Tasks under Error Bursts 115

8.1 Introduction .............................................. 117
8.2 Related Work ............................................ 119
8.3 System Model ............................................ 121
  8.3.1 Task model ........................................... 121
  8.3.2 Scheduling Model and Fault Tolerance Strategy ....... 122
  8.3.3 Execution Time Model .............................. 122
8.4 Problem Description .................................... 123
## Contents

8.5 Fault Tolerance Feasibility Analysis .................................... 123
8.6 Resource Augmentation for
    FT-Feasibility .................................................. 132
8.7 Example ............................................................... 135
8.8 Conclusions ........................................................... 138
Bibliography ............................................................... 139
I

Thesis
Chapter 1

Introduction

1.1 Real-time Systems

Real-time computing paradigm is being ubiquitously deployed in many areas and is increasingly becoming the backbone of most mission and safety critical systems. A real-time system is a computer system where the correctness of the system depends not only on the functional characteristics of the computations performed, but also on its temporal characteristics [2]. The temporal characteristics of the computations are derived from the temporal properties of the events occurring in the environment that the real-time system interacts with. The events occurring in the environment, that may be periodic, sporadic or aperiodic, are mapped to a set of real-time tasks to perform a desired function within a bounded time interval. The real-time tasks are specified using a set of task attributes, that reflect the timing requirements of the corresponding events. In this thesis, we focus on periodic and sporadic real-time tasks that are characterized by a release time, an exact or a minimum inter-arrival time, a Worst Case Execution Time (WCET) and a relative deadline with respect to its release time, as shown in figure 1.1. Each task generates an infinite sequence of jobs where any two consecutive jobs are separated by the exact/minimum inter-arrival time. All the jobs generated by the real-time tasks have to complete their execution before their respective deadlines.

Depending on the consequences of a deadline miss in the system, real-time systems can be classified as hard real-time systems and soft real-time systems. In hard real-time systems, e.g., aircraft control systems, a deadline miss can potentially cause catastrophic consequences such as loss to life and
4 Chapter 1. Introduction

property. On the other hand, a deadline miss in soft real-time systems, e.g., telecommunication systems, merely leads to a degradation of the service level.

1.2 Real-time Scheduling

Real-time scheduling can be classified as offline scheduling and online scheduling. In offline scheduling, the schedule is computed offline and is stored, typically in a table, and the tasks are executed according to this order. The main advantage of using offline scheduling algorithms is that much of the complexity involved in generating a valid schedule is handled offline and it has a simple runtime mechanism. However, the main disadvantage is that the schedule needs to be regenerated every time a new task is added to the system. On the other hand, in online scheduling, the scheduling decision is taken online, based on a suitable criteria e.g., Earliest Deadline First (EDF) or highest priority first. The main advantage of using an online scheduling algorithm is the flexibility it provides to the scheduler in accounting for dynamically changing factors e.g., new task arrivals. Online scheduling, which is typically priority driven, can be further classified as static priority and dynamic priority scheduling. In static priority scheduling, also known as Fixed Priority Scheduling (FPS), the scheduling decisions are based on the priorities that are determined offline and the

Figure 1.1: Real-time task attributes

![Diagram of real-time task attributes](image)
task priorities does not change online e.g., Rate-Monotonic (RM) scheduling and Deadline Monotonic (DM) scheduling. In dynamic priority scheduling, the scheduling decisions depend on the priorities that change dynamically e.g., Earliest Deadline First (EDF) scheduling and Least Laxity First (LLF) scheduling.

One of the main goals with respect to the design of real-time systems is to provide temporal guarantees to the set of real-time tasks. A schedulability analysis is a technique designed for a particular scheduler that determines whether or not there will be a deadline miss in the schedule generated by that scheduler for any given task set. There exists several utilization based [3], response time based [4][5] and demand bound based [6][7] schedulability tests for various scheduling algorithms such as FPS or EDF. A feasibility analysis on the other hand determines the existence of a real-time schedule for a given task set, independent of the scheduler. However, many of the well known feasibility analysis techniques [6][7] build on the optimality of scheduling algorithms such as EDF [8] to determine the existence of a real-time schedule.

Real-time scheduling can also be classified as preemptive and non-preemptive scheduling. In preemptively scheduled systems, a lower priority task can be preempted in favor of a higher priority task. This enables preemptive schedulers to achieve significantly high processor utilization and preemptive scheduling is known to strictly dominate non-preemptive scheduling with respect to feasibility [9]. On the other hand in a non-preemptively scheduled system, the lower priority task is allowed to complete its execution before a higher priority task is scheduled. Non-preemptive scheduling schemes increase blocking times on higher priority tasks, which may have shorter deadlines, leading to a significantly under utilized system [9]. It can also be shown that, in general, non-preemptive scheduling can be infeasible even at arbitrarily low utilizations [10]. One of the main advantage of using a non-preemptive scheduler is the extend of determinism it provides to the system, due to the absence of preemption related overheads, that are typical of a preemptive scheduler.

Even though preemptive scheduling strictly dominates non-preemptive scheduling with respect to feasibility [9], it suffers from preemption related overheads that may translate as temporal overheads, causing deadline misses in the schedule. These preemption related overheads e.g., cache related preemption delays, may invalidate the arguments in many schedulability analysis schemes e.g., worst case response times at critical instant for FPS [11]. One of the ways to overcome this problem is to control the number of preemptions as well as the points at which preemptions occur, thereby bounding the associated overheads.

In chapter 2, we discuss in detail about the preemption related issues in
real-time systems, the pros and cons of using a non-preemptive scheduler and motivate the need to control preemptions.

1.3 Energy Awareness in Real-time Systems

The increasing use of more powerful processors coupled with the increase in the mobile nature of most real-time applications, necessitated sound methods to conserve energy in the usually processor intensive real-time applications. Also, the advances in battery technology did not keep its pace with the advances in processor technology requiring efficient utilization of the energy resources due to the increase in need for energy. Various methods like switching off the unused devices, selectively switching off certain circuits in the processor and Dynamic Voltage Scaling (DVS), were proposed for managing the power consumption of the system. Many of these methods were adopted fairly quickly by the industry with publication of standards like ACPI [12] which established an open standard for power management. DVS has been applied to energy constrained real-time systems to prolong its operational lifetime, while meeting the real-time requirements. Increasing the processor frequency reduces the task execution times in many applications, however, requiring higher operating voltages. The general strategy adopted for implementing DVS for reducing power consumption is to utilize the slack in the schedule to slow down task executions, by lowering the processor frequency and the applied voltage. This has been shown to provide significant reduction in the processor power consumption, which quadratically decreases with decrease in applied voltage and linearly with the frequency. While applying DVS techniques in real-time systems to save energy, one must ensure that the slowing down of task executions does not cause any deadline misses in the schedule.

The possibility of CPU frequency scaling provides for the ability to control task execution rates in a real-time system to influence the real-time schedule, making it one of the most important research area in real-time computing.

1.4 Dependability in Real-time Systems

Ubiquitous deployment of real-time systems in safety and mission critical applications necessitates high levels of dependability due to the catastrophic consequences of a failure. Dependability, as defined by Laprie et. al. [1], is the ability of a system to deliver a justifiably trusted service. The service delivered by the system is defined as the perceived behavior of the system by a user,
which can be a human or another system, at the system interface. Laprie et. al. [1] identified the threats to dependability, the attributes of dependability and the means to achieve dependability in their famous dependability tree (refer figure 1.2). Availability, reliability, safety, integrity, confidentiality and maintainability together constitutes the attributes of dependability. The threats that affect the dependability of a system include faults, which is the hypothesized cause of an error, that may eventually lead to a failure of the system. An error in a subsystem can be considered as a fault in the system. Hence, the usage of faults and errors depends on the level of abstraction. Previous research on dependable real-time systems [13][14][15] focused on single errors per task/job, which may not be realistic [16]. A more realistic error model may be to consider error bursts e.g., a vehicle passing through a region of electromagnetic interference that makes any useful task execution impossible during that du-
One of the means of achieving dependability is to employ fault tolerance mechanisms, that can mask subsystem failures from the system.

Temporal redundancy is one of the most commonly employed fault tolerance strategy that can be used to tolerate transient and intermittent faults in the system by a simple execution of a recovery computation, which may be a re-execution of the failed computation or the execution of an alternate computation. The failed computations represent a temporal overhead in the system, potentially causing deadline misses, and can be seen as fault tolerance related overhead. In real-time systems, a task is typically considered as an independent unit of failure. Real-time fault tolerance aims at re-executing the failed task or executing an alternate task before a predefined deadline, which is usually the original deadline of the failed task, requiring sufficient slack in the schedule for fault tolerance. The main challenge here is to control these fault tolerance related overheads so that the feasibility of the task set can be guaranteed even under the presence of faults.

1.5 Resource Augmentation

Augmenting the scheduler with extra resource, e.g., a faster processor, can be beneficial for controlling the behavior of the system to guarantee specified requirements e.g., preemption control. While augmenting the scheduler with extra resources to achieve a specified behavior, the amount of extra resources required should be bounded by a reasonable value for it to be useful in practice. Resource augmentation analysis aims at finding the effects of having additional resources in the system and to find upper-bounds on the extra resources, to achieve a certain specified system behavior. Additionally, resource augmentation provides insights into the parameters that affect the satisfaction of the specified goal, giving greater flexibility to the system designer while designing the system. Kalyanasundaram et. al. [17] first introduced resource augmentation and proved that augmenting a non-clairvoyant online scheduler with a faster processor can effectively buy the power of clairvoyance.

In this thesis, we derive resource augmentation bounds that guarantee the feasibility a set of real-time tasks under preemption and fault tolerance related overheads and propose methods to control these overheads.
Chapter 2

Motivation and Problem Description

2.1 Motivation

The unpredictable and costly run time overheads in real-time systems, e.g., pre-emption related overheads and fault tolerance related overheads, can be con- 
trolled by changing the task attributes [18]. In this thesis, we control the task worst case execution times (WCET) and consequently the real-time schedule, by controlling the processor speed. Preemption related overheads can be con- 
trolled by controlling the number of preemptions and the points at which these preemptions occur. This can be achieved by controlling the task execution times such that preemptions occur only at specified points in the task code e.g., speed up the task such that it completes before a preemption. Fault tolerance feasibility under a given fault hypothesis can be achieved by using a faster pro- cessor which ensures that sufficient slack exists in the schedule to successfully schedule the recoveries of the failed tasks under an error burst of known length without causing deadline misses.

The detrimental impact of preemptions on task schedulability has received considerable attention from the research community [11] [19] and the need for preemption control is widely recognized. The most commonly considered preemption related overheads are:

1. **Context Switch Related Overheads**: Whenever a lower priority task is preempted by a higher priority task, the context of the preempted task
is saved to enable its execution from the same point when it resumes its
execution at a later time. The time taken to perform this context switch
manifests as a temporal overhead in the schedule, potentially causing
deadline misses.

2. **Cache Related Preemption Delays:** Preemptions can also displace data
from the cache memory, which has to be reloaded when the preempted
task resumes its execution. This translates to a temporal overhead, which
can vary with the point of preemption [11]. The cache related preemp-
tion delays can be of the order of hundreds of micro seconds for a single
preemption [20]. Consequently, the total temporal overhead on a task,
due to preemptions, can be very high depending on the number of pre-
emptions and the points at which the preemptions occur. Hence, bound-
ing the number preemptions, as well as restricting them to only certain
points in the task code is highly desirable.

3. **Pipeline Related Overheads:** Preemption on a lower priority task flushes
the instruction pipelines, to load the instructions of the higher priority
task. When the preempted task resumes its execution, the pipeline has
to be refilled. This flushing and refilling of the instruction pipeline man-
ifest as a temporal overhead in the system, leading to potential deadline
misses.

Non-preemptive scheduling on the other hand increases blocking times on
higher priority tasks leading to a significantly underutilized system. Conse-
quently, non-preemptive scheduling can be prohibitively expensive for most
applications due to space and cost constraints. Hence, controlling preemptions in a real-time schedule can be an attractive option, i.e., enable non-preemptive execution of tasks as long as necessary, to reduce the indeterminism in the task execution times. As seen earlier, the preemption related overheads in a schedule depend on the number of preemptions in the schedule and the points at which the preemptions occur. Therefore, by controlling the number of preemptions and the points at which these preemptions occur, the associated overheads can be controlled.

Figure 2.1 shows examples of a preemptive and a non-preemptive schedule along with the associated temporal overheads.

Implementing temporal redundancy requires that, in any time interval under a given fault hypothesis, there exists enough slack in the schedule to achieve one successful execution of all the tasks, that have their release times and deadlines within that interval. If enough slack does not exist during some time interval, there is a possibility of a deadline miss during that time interval. As mentioned earlier, the real-time tasks must respond to events occurring in the environment in a timely manner and hence, most of the real-time task attributes e.g., task periods and deadlines, are derived from the characteristics of these events, and are therefore strict. Consequently, one way of ensuring the existence of enough slack in the schedule to guarantee fault tolerance by temporal redundancy, is to use a faster processor. Finding such a processor speed can be done by performing a simple search among the set of available processor speeds. However, before performing a search, it may be desirable to verify if there exists a bound on the lowest processor speed that can guarantee fault tolerance. Hence, we need to find an upper-bound on the lowest processor speed.
that guarantees fault tolerance feasibility using temporal redundancy under a
given fault assumption. Figure 2.2 shows an example of the fault tolerance
related overheads under an error burst, in systems that employ temporal re-
dundancy.

Hence, in this thesis, we examine the use of resource augmentation to con-
trol the preemption and fault tolerance related overheads, to guarantee a desired
preemption behavior and fault tolerance feasibility.

2.2 Problem Description

Our aim is to use resource augmentation, specifically a faster processor, to
achieve preemption control and fault tolerance feasibility. Specifically, we con-
sider the following questions:

Q1 How can we control preemptions in a periodic task system using CPU
frequency scaling?

Q2 How can we control preemptions in a sporadic task system using CPU
frequency scaling?

Q3 What are the resource augmentation bounds that guarantees the feasibil-
ity of a specified non-preemption behavior?

Q4 What are the resource augmentation bounds that guarantees fault toler-
ance feasibility of a set of real-time tasks under an error burst of known
length?

We present methods to provide non-preemption and fault tolerance feasibility
guarantees using resource augmentation. We use analytical and experimental
approaches to evaluate our proposed methods.
Chapter 3

Main Contributions

Our contributions are presented in four papers which address various aspects of the problem. We first present an offline method to control preemptions in periodic task systems using CPU frequency scaling. We also present a combined offline-online approach to perform preemption control in sporadic task systems. We then derive the resource augmentation bound that guarantees the feasibility of a certain specified preemption behavior for periodic and sporadic task systems. Consequently, we derive the resource augmentation bound that guarantees the feasibility of a fully non-preemptive schedule. We also present a method to translate the requirements of meeting a specified upper-bound on the preemption related costs to a set of non-preemption requirements on the tasks. Later, we use the resource augmentation bound in a sensitivity analysis to calculate the exact processor speed that guarantees the feasibility of the derived non-preemption requirements for any task set. Finally, we apply resource augmentation to find upper-bounds on the processor speed-up that guarantees fault tolerance feasibility of a set of real-time tasks under an error burst of known length.

3.1 Summary of Contributions

In this section, summarize our solutions to the research questions raised in the previous chapter.
3.1.1 Paper A

Reducing the Number of Preemptions in Real-Time Systems Scheduling by CPU Frequency Scaling, Abhilash Thekkilakattil, Anju S Pillai, Radu Dobrin and Sasikumar Punnekkat, In proceedings of the 18th International Conference on Real-Time and Network Systems\(^1\), Toulouse, France, November, 2010

Summary: Controlling the number of preemptions in real-time systems is highly desirable in order to achieve an efficient system design in multiple contexts. For example, the delays due to context switches account for high preemption overheads which detrimentally impact the system schedulability. Preemption avoidance can also be potentially used for the efficient control of critical section behaviors in multi-threaded applications. At the same time, modern processor architectures provide for the ability to selectively choose operating frequencies, primarily targeting energy efficiency as well as system performance. In this paper, we propose the use of CPU Frequency Scaling for controlling the preemptive behavior of real-time tasks. We present a framework for selectively eliminating preemptions, that does not require modifications to the task attributes or to the underlying scheduler. We evaluate the proposed approach by four different heuristics through extensive simulation studies.

My contributions: Main author of the paper and performed simulations. Anju S Pillai implemented the task generator and helped with the literature survey. All the co-authors contributed by participating in the discussions and in reviewing the paper.

Relation to the research questions: This paper, which proposes an offline preemption control method for periodic task systems, addresses research question Q1.

3.1.2 Paper B

Probabilistic Preemption Control using Frequency Scaling for Sporadic Real-time Tasks, Abhilash Thekkilakattil, Radu Dobrin and Sasikumar Punnekkat, In proceedings of the 7th International Symposium on Industrial Embedded Systems, IEEE, Karlsruhe, Germany, June, 2012

\(^1\)This paper is also published in IEEE as "Preemption Control Using Frequency Scaling in Fixed Priority Scheduling" in the IEEE/IFIP Embedded and Ubiquitous Computing Conference since RTNS-2010 had no copyright.
Summary: Preemption related costs are major sources of unpredictability in the task execution times in a real-time system. We examine the possibility of using CPU frequency scaling to control the preemption behavior of real-time sporadic tasks scheduled using a preemptive Fixed Priority Scheduling (FPS) policy. Our combined offline-online method provides probabilistic preemption control guarantees by making use of the release time probabilities of the sporadic tasks. The offline phase derives the probability related deviation from the minimum inter-arrival time of tasks. The online algorithm uses this information to calculate appropriate CPU frequencies that guarantees non-preemptive task executions while preserving the overall system schedulability. The online algorithm has a linear complexity and does not lead to significant implementation overheads. Our evaluations demonstrate the effectiveness of the method as well as the possibility of energy-preemption trade offs. Even though we have considered FPS, our method can easily be extended to dynamic priority scheduling schemes.

My contributions: Main author of the paper and performed simulations. All the co-authors contributed by participating in the discussions and in reviewing the paper.

Relation to the research questions: This paper, which proposes a combined offline-online preemption control method for sporadic task systems, addresses research question Q2.

3.1.3 Paper C

Quantifying the Sub-Optimality of Non-Preemptive Real-time Scheduling, Abhilash Thekkilakattil, Radu Dobrin and Sasikumar Punnekkat, Technical Report, Mälardalen Real Time Research Centre, Mälardalen University, Västerås, Sweden, November, 2012

Summary: Many preemptive real-time scheduling algorithms, such as the Earliest Deadline First (EDF), are known to be optimal on a uni-processor. However, no such algorithms exist under the non-idling non-preemptive scheduling paradigm. Hence preemptive schemes strictly dominate non-preemptive schemes with respect to feasibility. However, the ’goodness’ of non-preemptive schemes in successfully scheduling feasible task sets when compared to uni-processor optimal preemptive scheduling schemes such as the EDF, which can
also be referred to as its sub-optimality, is unknown. In this paper, we apply resource augmentation, specifically processor speed-up, to quantify the sub-optimality of non-preemptive scheduling with respect to an optimal uni-processor scheduling scheme such as the EDF. We also present a method to guarantee user specified upper-bounds on the preemption related costs in the schedule.

We prove that the speed-up required to guarantee the feasibility of a non-preemptive execution of any task $\tau_i$, for a duration of $L_i$, is upper-bounded by $\frac{4L_i}{D_{\text{min}}}$, where $D_{\text{min}}$ is the smallest relative deadline in the task set. Consequently, we show that the upper-bound on the processor speed that guarantees the feasibility of a non-preemptive schedule for the task set is $\frac{4C_{\text{max}}}{D_{\text{min}}}$, where $C_{\text{max}}$ is the largest execution time in the task set. The derived upper-bound is used in a sensitivity analysis based method to calculate the optimal processor speed that guarantees a specified upper-bound on the preemption related costs in the schedule. For this, we first present a method to translate the system-level requirements of meeting specified upper bounds on the preemption related costs to task level non-preemption requirements. We then use sensitivity analysis technique to calculate the optimal processor speed that guarantees the feasibility of the derived task level non-preemption requirements, which in its turn guarantees the desired bounds on the preemption related overheads.

Our contribution quantifies the sub-optimality of non-preemptive scheduling in terms of the processor speed-up required to successfully schedule all the uni-processor feasible task sets. It also enables a system designer to use a faster processor to guarantee specified upper-bounds on the preemption related overheads.

**My contributions:** Initiator and main author of the paper. All the co-authors contributed by participating in the discussions and in reviewing the paper.

**Relation to the research questions:** This paper derives upper-bounds on the lowest processor speed that guarantees the feasibility of a specified non-preemption behavior of a set of real-time tasks. Hence, we can obtain the upper-bound on the processor speed that can guarantee the feasibility of a fully non-preemptive schedule. Using these upper-bounds in a sensitivity analysis, exact processor speed-up that guarantees the feasibility of a specified non-preemption behavior is derived. The paper addresses research question Q3, while also providing a solution for questions Q1 and Q2.
3.1 Summary of Contributions

3.1.4 Paper D

Resource Augmentation for Fault-Tolerance Feasibility of Real-time Tasks under Error Bursts, Abhilash Thekkilakattil, Radu Dobrin, Sasikumar Punnekkat and Huseyin Aysan, In proceedings of the 20th International Conference on Real-Time and Network Systems, ACM, Pont á Mousson, France, November, 2012 (Shortlisted for Best Student Paper Award)

Summary: Dependability is a vital system requirement, particularly in safety critical and mission critical real-time systems, due to the potentially catastrophic consequences of failures. In most critical applications different fault tolerance mechanisms using redundancy are employed to prevent possible failures. In the case of real-time systems the system designer must ensure that the task set is feasible even under faults, which we refer to as fault tolerance feasibility. Due to cost considerations, often temporal redundancy has been prevalently used to meet this objective. In this paper we focus on guaranteeing fault-tolerance feasibility under error bursts on uni-processor systems by the usage of resource augmentation, specifically through processor speed-up. Firstly, we derive a processor demand bound based sufficient condition for a set of real-time tasks to be fault tolerance feasible under an assumption that no more than one error burst occurs during the hyper-period of the task set. Subsequently, we derive the necessary resource augmentation bounds (i.e., the processor speed-up), that guarantees the fault tolerance feasibility, if the sufficient test fails. Finally, we prove that, if the error burst length is no more than half the shortest relative deadline of the task set, the processor speed-up required to guarantee fault tolerance feasibility is upper-bounded by 6.

My contributions: Initiator and main author of the paper. All the co-authors contributed by participating in the discussions and in reviewing the paper. The motivation of the work builds on Huseyin Aysan’s PhD thesis [16].

Relation to the research questions: This paper derives upper-bounds on the processor speed-up required, that can guarantee fault tolerance feasibility of a set of real-time tasks under an error burst of known length. It addresses the research question Q4.
3.2 Significance of the Contributions

The optimality of preemptive and non-idling non-preemptive Earliest Deadline First (EDF) scheduling, under the respective assumptions, is well known [8] [21]. However, not all the task sets schedulable by preemptive EDF is schedulable by non-idling non-preemptive EDF, as preemptive scheduling strictly dominates non-preemptive scheduling [9]. Despite the domination of preemptive scheduling over non-preemptive scheduling, due to preemption related overheads, a preemptive schedule may not always be feasible and limiting preemptions [9] [22] may be necessary to guarantee feasibility. However, the limited preemption model [9] [22] may not always guarantee a specified preemption behavior e.g., guarantee a specified upper-bound on the number of preemptions or on the preemption related costs, because the largest non-preemptive region per task in any time interval is bounded by the processor demand in that interval. The dependence of the largest non-preemptive region of a task on the processor demand enables us to use a faster processor to control its length. Our upper-bound on the processor speed-up, guarantees a specified non-preemption behavior per task (ranging from preemptive to fully non-preemptive), achieving a certain generality. Hence, it opens up the possibility of augmenting the scheduler with a faster processor to guarantee the feasibility of the set of all uni-processor feasible task sets by a non-idling non-preemptive EDF scheduler. Consequently, it provides significant insights into developing a utilization based test for non-preemptive feasibility of periodic and sporadic real-time tasks.

A yet another interesting contribution of this thesis is the quantification of the sub-optimality of the non-preemptive scheduling scheme with respect to a uni-processor optimal preemptive scheduling scheme. As mentioned earlier in this section, preemptive scheduling strictly dominates non-preemptive scheduling with respect to feasibility. However, the 'goodness' of non-preemptive scheduling, when compared to a uni-processor optimal preemptive scheduling scheme, which is referred to as its sub-optimality, is unknown. In this thesis, we derive the resource augmentation bound that guarantees the feasibility of a non-preemptive schedule, for any set of real-time tasks that is uni-processor feasible. This allows us to quantify the sub-optimality of non-preemptive scheduling in terms of the processor speed-up required to guarantee the non-preemptive feasibility of any uni-processor feasible task set.

Bounding the processor speed-up that guarantees the fault tolerance feasibility of real-time tasks under an error burst of known length simplifies many design decisions, while building safety and mission critical real-time systems.
In cases where the original task set was feasible upon a uni-processor under an error free scenario, but is not feasible under an error burst, a system designer can opt for a faster processor to achieve fault tolerance feasibility under the error burst. We have shown that if the error burst length is no longer than half the shortest deadline of the task set, the lowest processor speed-up required to guarantee fault tolerance feasibility is upper-bounded by a constant 6. Hence, in this case, by increasing the processor speed by only a constant factor, the fault tolerance feasibility of a real-time task set can be guaranteed.
Chapter 4

Conclusions and Future Work

4.1 Conclusions

In this thesis, we present methods to control preemption related and fault tolerance related overheads in a real-time system, using resource augmentation. We first present an offline preemption control method for periodic task systems, that derive individual task instance frequencies which reduces the number of preemptions in the schedule. Such an offline method is however not possible in a sporadic task system, when the task release times are not known offline. We hence propose a combined offline-online approach to control preemptions in a sporadic task system where the probabilities of the task releases are known. We first find the probability related deviation from the minimum inter-arrival times of the task. We then use this information in an online preemption control algorithm that, at any time instant, determines the maximum time for which the outstanding computations can execute non-preemptively and performs CPU frequency scaling to avoid a preemption. We hence demonstrate the feasibility of using resource augmentation to control preemptions in periodic and sporadic systems.

We derive the upper-bound on the processor speed-up required that guarantees the feasibility of a specified preemption (or alternately non-preemption) behavior of a set of real-time tasks. We show that the upper-bound on the processor speed-up that guarantees a non-preemptive region of length $L_i$ for
any task \( \tau_i \) is given by \( \frac{4L_i}{D_{\text{min}}} \), where \( D_{\text{min}} \) is the smallest deadline in the task set. Consequently, we show that the upper-bound on the processor speed that guarantees the feasibility of a fully non-preemptive schedule is given by \( \frac{4C_{\text{max}}}{D_{\text{min}}} \), where \( C_{\text{max}} \) is the largest execution time in the task set. However, changing the processor speed changes the required length of the non-preemptive region as well as its largest possible length. We finally use the derived bound to guarantee a user specified upper-bound on the preemption related costs. In this approach, we first present a method to derive a set of non-preemption requirements per task \( \tau_i \), which is the required length \( L_i \) of the non-preemptive region for \( \tau_i \), that guarantees the specified upper-bounds on the preemption related costs. Then, using a sensitivity analysis based approach, we derive the exact processor speed-up that guarantees the feasibility of the derived non-preemption requirement \( L_i \) for any such task \( \tau_i \).

Finally, we apply resource augmentation to guarantee the fault tolerance feasibility (FT-feasibility) of a set of real-time tasks under an error burst of known length during the hyper-period (LCM of time periods) of the task set. To calculate the bound, we first derive a sufficient condition that guarantees the feasibility of a set of real-time tasks under the error burst. We then derive the necessary bounds on the lowest processor speed-up required that guarantees FT-feasibility if the sufficient condition fails. We then show that if the error burst length is no longer than half the shortest deadline in the schedule, the required lowest processor speed-up is upper-bounded by a constant 6. Thus, by only a constant times increase in the processor speed, the FT-feasibility of a real-time task set can be guaranteed when the error burst length is no longer than half the shortest deadline.

### 4.2 Future Work

Some possible future work include:

- **Extensions to multi-processor scheduling**: Derive resource augmentation bounds that guarantees a specified non-preemption behavior for multi-processor systems.

- **Utilization based tests**: Develop utilization based tests for non-preemptive scheduling, and to determine the FT-feasibility of a real-time task set.

- **Redefining the notion of feasibility of real-time tasks**: Augmenting the notion of feasibility of real-time task sets with the extend of processor
speed-up required to guarantee the existence of a schedule even under the presence of overheads.

- **Contracts for preemption control and fault tolerance:** Incorporating the possibility of processor speed-up to derive online/offline contracts to achieve preemption control and fault tolerance feasibility in component based real-time systems.
Bibliography


