Best-case response times of real-time tasks under fixed-priority scheduling with preemption thresholds

Rivera Verduzco, H.J.

Award date:
2017

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
Best-case response times of real-time tasks under fixed-priority scheduling with preemption thresholds

Master Thesis

Héctor Joao Rivera Verduzco
ID nr. 0977393
h.j.rivera.verduzco@student.tue.nl

Supervisor:
Dr. ir. Reinder J. Bril

Committee:
Dr. ir. Reinder J. Bril
Dr. Rudolf H. Mak
Prof. Dr. Kees G.W. Goossens

Final version

Eindhoven, July 2017
Abstract

Nowadays, real-time systems are found in a wide range of applications, from small consumer electronic devices to large critical applications in the area of robotics, aeronautics, automotive, etc. The main purpose of a real-time system is to provide a correct response within certain time bounds. Therefore, it is imperative to determine bounds on the response time of real-time tasks to guarantee that they meet the timing constraints imposed by the system.

Since in a processor normally a set of tasks execute concurrently, the order in which the tasks are scheduled has an impact on the response time of real-time tasks. There exist many scheduling approaches to determine the order in which tasks execute on a processor. One of the most well known is fixed-priority preemptive scheduling (FPPS) that is supported by most real-time operating systems. However, fixed-priority scheduling with preemption thresholds (FPTS) has been viewed as a potential successor to FPPS as a de facto standard in industry, where it is already supported by both OSEK and AUTOSAR compliant operating systems. Compared to FPPS, FPTS reduces memory requirements, the cost of arbitrary preemptions, and it has been shown to improve the ratio of the task-sets that can meet their upper timing constraints.

In this thesis, we present and prove a novel exact best-case response time analysis for independent real-time periodic tasks scheduled using FPTS. As an initial step towards the best-case analysis for FPTS, we first prove that the best-case response time of non-preemptive tasks is equal to their best-case computation times. Next, we formalize the exact analysis based on facts derived from experimentation related with the best-case behaviour of a task scheduled using FPTS.

Unlike FPPS, the exact analysis for FPTS that we propose has exponential time complexity. Therefore, we also propose and prove a novel lower bound for the best-case response time of a task that provides a tractable schedulability test for tasks. In addition, we present an evaluation of our exact analysis for FPTS with the novel lower bound and other trivial lower bounds for the response time. Such trivial lower bounds include the best-case response time analysis for FPPS when ignoring the tasks that cannot preempt a task. For the majority of the cases considered in the evaluation, our novel lower bound and the best-case response time analysis for FPPS yield a tight lower bound for the response time of a task scheduled using FPTS. However, in some cases, our novel lower bound achieves better results than the best-case response time for FPPS.
Preface and Acknowledgments

“Una vez que empieces tu viaje, nunca querrás detenerte.”
“Once you start your journey, you will never want to stop.”

— Tuila Aída Martinez Garza.

This Master’s thesis is the result of the graduation project for the Embedded Systems program at Eindhoven University of Technology (TU/e). This project has been conducted within the System Architecture and Networking group in the department of Mathematics and Computer Science.

The original topic of this project was related to the application of limited preemptive techniques, such as fixed-priority scheduling with preemption thresholds (FPTS), to control tasks in order to evaluate its advantages and disadvantages in terms of jitter. During the feasibility study of the project, we identified the need of an exact best-case response time analysis for FPTS in order to bound the response jitter of tasks as tightly as possible. Since the best-case response time analysis for FPTS is already an interesting topic on its own, we divided the project into two possible directions. The first one related with the exact best-case analysis for FPTS, and the second direction concerning a comparative evaluation in terms of jitter of FPTS with other scheduling approaches using extensive simulations rather than the exact analysis. Being the exact best-case analysis the most interesting of the two, we decided to start with the analysis for FPTS and only change the direction if the corresponding proof and formalization of the analysis resulted to be out-of-reach for the given time. In fact, the best-case analysis resulted to be more challenging than what I initially expected. Mainly due to the fact that the best-case behaviour of FPTS presents some properties that are neither intuitive nor immediately apparent. At the end, it was possible to successfully address the route with the proof for the best-case response time of FPTS.

I would like to express my sincere gratitude to my supervisor dr.ir. Reinder J. Bril for his guidance during the realization of this Master’s project. He always provided valuable feedback in order to improve the quality of the work. Furthermore, he was always willing to discuss subjects, and to provide advice whenever I needed it. I have no doubt that, without his valuable support, I would not have been able to achieve the current results.

I would like to thank the members of the assessment committee dr.ir. Reinder J. Bril, dr. Rudolf H. Mak and prof.dr. Kees G.W. Goossens for their interest and their time to examine this thesis. Furthermore, I would also like to thank dr. Dip Goswami for his initial feedback and interest during the feasibility study of this Master’s project.

I am gratefull to the National Council of Science and Technology (CONACYT) and the State Council of Science and Technology of Jalisco (COECYTJAL) for granted me a scholarship to study in the Netherlands during the past two years. Receiving such a scholarship greatly motivated me to complete my degree.

My special thanks go to my parents and two older sisters for their continuous support. Besides the time difference between Mexico and the Netherlands, we always found the time to talk. The same holds true for my girlfriend Nelly, who has supported me during my entire study at TU/e.

Héctor Joao Rivera Verduzco
Eindhoven, July 2017
# Contents

1 Introduction ................................................................. 1  
   1.1 Background and related work ........................................ 1  
   1.2 Motivation ............................................................ 2  
      1.2.1 Control system ................................................. 2  
      1.2.2 Distributed systems ........................................... 2  
   1.3 Problem statement ................................................... 3  
   1.4 Research questions and research approach .......................... 3  
   1.5 Contributions ....................................................... 3  
   1.6 Thesis outline ..................................................... 4  

2 Real-Time Scheduling Model ............................................. 5  
   2.1 Basic task model ..................................................... 5  
   2.2 Fixed priority scheduling algorithms ............................... 5  
      2.2.1 Basic scheduling concepts .................................... 5  
      2.2.2 FPPS Scheduling ............................................... 5  
      2.2.3 Refined model for FPTS ....................................... 6  
   2.3 Related notions and schedulability .................................. 6  
      2.3.1 Derived notions ............................................... 6  
      2.3.2 Schedulability criterion ...................................... 7  
      2.3.3 Hyperperiod ................................................... 7  
   2.4 An example ........................................................... 7  

3 Recap of Existing Analysis ............................................. 8  
   3.1 Worst-case response time analysis for FPPS ......................... 8  
      3.1.1 A critical instant ............................................ 8  
      3.1.2 Level-i active period ....................................... 9  
      3.1.3 Worst-case response times .................................. 10  
   3.2 Worst-case response time analysis for FPTS .......................... 10  
   3.3 Best-case response time analysis for FPPS .......................... 12  

4 Non-preemptive Tasks and Lower Priority Tasks ....................... 14  
   4.1 Best-case response time for non-preemptive tasks .................. 14  
   4.2 Influence of lower priority tasks .................................. 15  
   4.3 Influence of thresholds of higher priority tasks ................... 18  

5 Towards an Optimal Instant .............................................. 19  
   5.1 Introductory examples .............................................. 19  
   5.2 Tasks influencing the best-case response time ..................... 22  
   5.3 A property for an optimal instant .................................. 22  
   5.4 The phasing of delaying and extra preempting tasks ................ 25  

Best-case response time of real-time tasks under FPTS ................. v
# 6 Exact Best-Case Response Time Analysis

6.1 Determining the hold times of a job .................................................. 26
6.2 Best-case interval for the execution of a task \( r_i \) ................................. 28
6.3 Shortest response time of a job .............................................................. 30
6.4 Best-case response times ................................................................. 31
6.5 An example ......................................................................................... 31
6.6 Some optimizations to the analysis ...................................................... 32
   6.6.1 Number of jobs to explore before an optimal instant ..................... 32
   6.6.2 A constraint on the hold time of a task ....................................... 32

# 7 A Lower-Bound for the Best-Case Response Time

7.1 A minimal preempting instant ............................................................ 34
7.2 Artificial hold time of a job ............................................................... 35
7.3 Best-case interval for the execution of a task experiencing an artificial hold time ........................................................................................................... 37
7.4 Shortest response time of a job experiencing an artificial hold time 39
7.5 Best-case response time lower bound ................................................. 40
7.6 An algorithm for computing the lower bound for the best-case response time ........................................................ ......................... 41
7.7 An example of the algorithm to derive \( BR_{lb}^i \) ................................. 42

# 8 Evaluation

8.1 Set-up for the evaluation .................................................................. 45
8.2 Constraint deadlines ....................................................................... 45
8.3 Arbitrary deadlines ........................................................................ 46

# 9 Conclusions

Appendix

A Auxiliary Lemmas ........................................................................ 53
B Additional Proofs ........................................................................... 54
   B.1 Proof for Lemma 4 ................................................................... 54
   B.2 Proof for Lemma 5 ................................................................... 55
   B.3 Proof for Theorem 4 .................................................................. 55
C Tools ................................................................................................. 56
   C.1 Grasp tool ............................................................................... 56
   C.2 FPTS scheduler tool ................................................................ 56
   C.3 Simulation tool .......................................................................... 57
   C.4 Exploration tool ........................................................................ 57
   C.5 Evaluation tool ........................................................................ 58
D Additional Examples ..................................................................... 59
   D.1 Example 1 ............................................................................. 59
   D.2 Example 2 ............................................................................. 60
   D.3 Example 3 ............................................................................. 61
List of Figures

2.1 Basic model for a periodic task $T_1$. .......................................................... 6
2.2 Timeline for $T_{2,1}$ when $\varphi_1 = 1$, $\varphi_2 = 2$ and $\varphi_3 = 0$. Curly red arrows indicate that a job missed its worst-case deadline. .......................................................... 7
3.1 Timeline for $T_{3,1}$ with a critical instant at time $t = 0$ for task $T_3$. ........... 9
3.2 Timeline for $T_{3,4}$ with a critical instant at time $t = 0$ for task $T_3$. ........... 12
3.3 Timeline for $T_{5,6}$ with an optimal instant at time $t = 0$ for task $T_3$. ........... 13
4.1 A timeline for $T_{4,1}$. The activations of the tasks are repeated in every interval with length $P_{T_{4,1}} = 12$. .......................................................... 15
4.2 A timeline for $T_{4,1}$. After pushing the activations of $T_2$, the second job can immediately start. .......................................................... 15
4.3 A timeline for $T_{4,2}\backslash\{r_3,r_4\}$ where the best-case response time for $T_2$ is found in the time interval $[0,6]$. .......................................................... 17
4.4 A timeline for $T_{4,2}\backslash\{r_4\}$ where the best-case response time of $T_2$ is preserved after introducing a job of $T_3$ in the time interval $[-6,0)$ before the job of $T_2$ starting at time $t = 0$. Note that this interval has exactly an amount of time $C_3 = 4$ available for $T_3$. .......................................................... 17
4.5 A timeline for $T_{4,2}$ where the best-case response time of $T_2$ is preserved after introducing a job of $T_4$ in the time interval $[-17,0)$. Note that this interval has exactly an amount of time $C_4 = 3$ available for $T_4$. .......................................................... 18
5.1 A timeline for $T_{5,1}$ depicting a level-3 active period. The best-case response time of $T_3$ is assumed by the last job in the level-3 active period. .......................................................... 20
5.2 A timeline for $T_{5,2}$ depicting a level-4 active period. A simultaneous activation of preemptive tasks $T_1$ and $T_2$ coincides with a completion of the last job of $T_4$ in the level-4 active period at time $t = 0$. .......................................................... 21
5.3 A timeline for $T_{5,2}$ depicting a level-4 active period where the best-case response time of task $T_4$ is assumed by the first job. .......................................................... 21
5.4 A timeline for $T_{5,3}$ showing a job $k_{br}$ of $T_1$ that assumes the best-case response time. All activations of higher priority tasks after $s_{i,k_{br}}$ occur after $s_{i,k_{br}} + \Delta$ as well. .......................................................... 23
5.5 A timeline for $T_{5,3}$ depicts the case when the delaying task $T_d$ is pushed till one activation coincides with $s_{i,k_{br}} + \Delta$. .......................................................... 23
5.6 A timeline for $T_{5,3}$ depicts the case when the delaying task $T_d$ is pushed till one activation coincides with $s_{i,k_{br}} + \Delta$. .......................................................... 24
5.7 A timeline for $T_{5,3}$ depicts the case when the delaying task $T_d$ is pushed till one activation coincides with $s_{i,k_{br}} + \Delta$. .......................................................... 24
6.1 Two timelines for $T_{6,1}\backslash\{r_3\}$ showing the hold times for a job of $T_4$ when experiencing extra preemptions by $T_1$ and minimal preemptions by $T_2$. Hence, $H_d((T_1)) = (32,54)$. .......................................................... 28
6.2 Timeline for $T_{6,1}$ depicting the generalized best-case interval $GI_4(2, BC_4, h, E_4) = 106$, where $h = 32$ and $E_4 = \{T_1\}$. .......................................................... 30
6.3 Two timelines for $T_{6,4}$. In a) $\tau_2$ is ignored from the task-set and $\tau_1$ is a minimal preemptive task of $\tau_3$. In b) task $\tau_2$ is introduced, affecting the schedule for $\tau_3$.  

7.1 A timeline for $T_{7,1}$ depicting the minimal preemption instant for the last job $i_{4,-1}$ of task $\tau_4$. Preempting tasks $\tau_1$ and $\tau_2$ are simultaneously activated at time $f_{4,-1} = 0$. Furthermore, delaying task $\tau_3$ is activated a sufficiently small amount of time after $s_{4,-1}$.  

7.2 Timeline for $T_{7,1}$ where an artificial hold time $H_{4,-1}^{ar} = 24$ is assumed by the last job $i_{4,-1}$ activated at time $t = -36$.  

7.3 Timeline for $T_{7,1}$ where a sufficiently large artificial hold time is assumed by the last job of $\tau_4$ to reduce its response time till the start of the artificial hold time.  

7.4 Timeline for $T_{7,3}$ during the first iteration of $bcrt\text{LowerBound}(T_{7,3,4})$.  

7.5 Timeline for $T_{7,3}$ showing the lower-bound for the best-case response time of task $\tau_4$ returned by $bcrt\text{LowerBound}(T_{7,3,4})$.  

8.1 Fraction of the times the best-case response time of the lowest priority task is equal to a lower bound versus the utilization of the task-set considering $n = 10$ tasks and $WD_i = T_i$.  

8.2 Fraction of the times the best-case response time of the lowest priority task is equal to a lower bound for different values of $\alpha$ considering $n = 10$ tasks and task-set's utilizations of $UT = 0.98$.  

8.3 Closeness of the lower bounds to the actual best-case response time versus task-set's utilization. Deadlines vary according to $\alpha = 0.7$, and $n = 10$ tasks are considered per system.
Chapter 1

Introduction

This thesis concerns the analysis of real-time scheduling with limited preemptions. Real time systems are computing system that, besides providing a correct logical answer, have to response to events within some time bounds. If the system fails to respond within such time bounds, the result may not be useful anymore or, in the worst-case, it may have catastrophic effects. In the past years, the use of real-time computing has increased considerably due to the increase of embedded computing in a wide range of applications. In fact, the applicability of real-time computing spans from small consumer electronic devices to large critical applications in the area of robotics, medical systems, automotive, etc.

In order to guarantee that a real-time system meets its timing constraints, it is important for the system to be predictable. This means that whenever the processor starts executing a real-time task, it should be possible to bound its response time. Since a processor may execute a set of concurrent tasks, the order in which tasks are scheduled may have an influence on the response time of a task. In this thesis, we investigate fixed-priority scheduling with preemption thresholds (FPTS) and, in particular, the lower bound (best-case) for the response time of a task under such a scheduling algorithm because exact worst-case response time analysis has already been provided.

In the remaining of this chapter, we present some background and related work regarding FPTS scheduling and its response time analysis. We also motivate the need of determining tight lower bounds for the response time of a task and we explicitly mention the problem statement. Furthermore, we present the research questions as well as the research approach taken to answer such questions. Finally, we mention the contributions addressed by this thesis.

1.1 Background and related work

FPTS was first introduced in the ThreadX real-time operating system [22] and then analyzed by Wang and Saksena in [24]. In particular, FPTS limits the number of preemptions that real-time tasks may experience compared to fixed-priority fully-preemptive scheduling (FPPS). In order to do so, a task may raise its priority after acquiring the CPU leading to a reduction in preemption overheads and memory requirements of the system. After the task finishes its execution, the priority is lowered to its original value. Although fixed-priority non-preemptive scheduling (FPNS) also presents the aforementioned benefits, it has been shown in [6] that FPTS improves the ratio of the task-sets that can meet their worst-case deadlines, i.e. the task-sets that are schedulable from a worst-case perspective. In addition, FPTS has been viewed as a potential successor to FPPS as a de facto standard in industry. An important reason for its success in industry is that switching from FPPS to FPTS can be done without any change to the task's code. The increase of the priority of a task after acquiring the CPU can be done at integration time. For example, FPTS is already supported by both OSEK [14] and AUTOSAR [1] compliant operating systems in the form of internal resources.
Special attention has been paid in the literature to the worst-case response time analysis for FPTS to provide guarantees for real-time tasks meeting their upper (worst-case) deadlines \[24\][16][10]. The analysis for worst-case response time is based on a critical instant, i.e. the time instant at which the worst-case response time of a task is assumed by one of its jobs. Although the worst-case analysis for FPTS has been already addressed in the literature, no research has been conducted yet regarding its best-case counterpart. Since FPTS is a generalization of FPPS and FPNS, it is natural to first consider the best-case response time analysis for FPPS and FPNS before making the step towards the corresponding best-case analysis for FPTS.

The best-case response time analysis for FPPS has been addressed in \[15\] for deadlines at most equal to periods (i.e. constraint deadlines). However, for deadlines smaller than, equal to or larger than periods (i.e. arbitrary deadlines), the analysis in \[15\] yields a lower bound that is not tight for the response time. In \[20\], an extension of the analysis is given that provides an exact best-case response time analysis for arbitrary deadlines. Moreover, these analyses are based on an optimal instant, i.e. the time instant at which the best-case response time of a task is assumed by one of its jobs. The work in \[20\] also shows the non-dual nature of the best-case and worst-case analysis for FPPS. Therefore, FPTS inherently has such a non-duality as well.

The best-case response time analysis for FPNS has received considerably less attention than FPPS. FPNS is widely used in message-based protocols under a network environment. For example, the Controller Area Network (CAN) uses a priority based arbitration and non-preemptive scheduling for message transmission. Furthermore, worst-case analysis for CAN has been addressed in \[7\], and for FPNS in \[21\]. Regarding its best-case analysis, the work in \[4\] presents a lower bound for the best-case response time under fixed-priority scheduling with deferred preemptions (FPDS). Since FPNS is a special case of FPDS, this analysis also applies for the former. In particular, this analysis gives the trivial result that the best-case computation time of a task is a lower bound for the best-case response time of such a task under FPNS. However, we would also like to investigate whether this lower bound is tight for all cases.

### 1.2 Motivation

The best-case response time of a task becomes important whenever timing constraints impose lower bounds on response times to events. A common example is an airbag that, upon a collision, has to be inflated neither too earlier nor too late. Another example in which the best-case response time is required is for the calculation of response jitter, i.e. the fluctuation in the response time of a task. Whenever it is necessary to bound the response jitter of a task as tightly as possible, determining its best-case response time becomes relevant. We now describe two applications in which response jitter is required.

#### 1.2.1 Control system

One of the goals of designing a control system is to provide a system that remains stable and achieves the desired control performance under some possible disturbance. In a control system, sampling and actuation is normally performed periodically. Furthermore, when a controller is implemented as a task, it can experience preemptions from other tasks leading to a variation in the time a control signal is sent to the actuator of the system; hence, leading to response jitter. It is well known that such variations in the response time of the controller can degrade the performance of the system and, in the worst case, can jeopardize its stability \[23\]. Therefore, it is important to take response jitter into consideration in the design of reliable control systems.

#### 1.2.2 Distributed systems

The need for best-case response time analysis in the area of distributed systems has been identified in \[11\]\[8\]. In distributed multiprocessors systems in which a task may trigger other ones, the time variation in the response time of a triggering task generates fluctuation in the activation of the
triggered task, i.e. activation jitter. Furthermore, activation jitter has a direct impact on the worst-case response time of lower priority tasks. It therefore becomes relevant to bound the response jitter of the triggering task as tightly as possible, in order to determine the worst-case response time of such lower priority tasks without introducing pessimism.

1.3 Problem statement

We are interested in investigating the best-case response time analysis for independent strictly periodic real-time tasks with arbitrary deadlines scheduled using FPTS, i.e. for tasks with deadlines smaller than, equal to, or larger than periods. Furthermore, we aim to evaluate tightness of some existing lower bounds for the response time of a task with its actual best-case response time.

1.4 Research questions and research approach

This thesis addresses the following research questions:

- **RQ1.** What is the exact best-case response time analysis for tasks scheduled using FPNS?
- **RQ2.** What is the exact best-case response time analysis for tasks scheduled using FPTS?
- **RQ3.** What are appropriate lower bounds for the best-case response time of a task scheduled using FPTS?

In order to answer the aforementioned research questions, we started by building a suite of tools. These tools allowed us to explore the behavior of FPTS in best-case situations and, in addition, to investigate the best-case response time of tasks scheduled using FPTS by means of brute-force. Based on such a tool-suite, we were able to identify some facts related with the best-case behaviour of a task scheduled using FPTS and, in addition, to determine how it differs with respect to the best-case behaviour for FPPS. The tool-suite therefore saved much manual work as well as facilitated the investigation of interesting examples that would be hard to find otherwise. Based on those facts and examples resulting from the experimentation, we were able to formalize the analysis. Hence, the combination of experimentation with formalization was imperative throughout the process of deriving the current results of this Master's project.

1.5 Contributions

This thesis presents three major contributions. Firstly, it proves that the best-case response time of non-preemptive tasks scheduled using FPTS or FPNS is equal to its best-case computation time. This contribution was already accepted as a work-in-progress (WiP) session paper for the 22nd IEEE International Conference in Emerging Technologies and Factory Automation (ETFA) [18].

Our second major contribution concerns the best-case response time of independent strictly periodic real-time tasks with arbitrary deadlines scheduled using FPTS. We present and prove an exact novel best-case analysis for such tasks. In addition, we perform an evaluation of the tightness of two trivial lower bounds for the response time of a task with respect to its exact best-case response time. Such trivial lower bounds are the best-case computation time of a task and the best-case response time analysis for FPPS when ignoring the tasks that cannot preempt another task. This contribution was submitted as a conference paper to the 25th International Conference on Real-Time Networks and Systems (RTNS) and it is currently under review [17].

Finally, this thesis also presents and proves a novel lower bound for the best-case response time tasks scheduled using FPTS that achieves a better time complexity than the exact best-case analysis. We also provide an evaluation for such a novel lower bound compared to the aforementioned trivial lower bounds. In particular, the novel lower bound achieves better results for arbitrary deadlines.
1.6 Thesis outline

This thesis is organized as follows. Chapter 2 introduces the real-time scheduling models for FPPS and FPTS that we analyze as well as some basic terminology. In Chapter 3, we briefly recapitulate the existing worst-case analysis for FPPS and FPTS. In addition, we also recapitulate the exact best-case response time for FPPS. The best-case response time for non-preemptive tasks is addressed in Chapter 4, as well as the influence of lower priority tasks in the best-case response time of another task scheduled using FPTS. Furthermore, in Chapter 5, we introduce some facts related with the best-case behaviour of a task scheduled using FPTS and, in addition, we derive a property for an optimal instant based on such facts. An exact best-case response time analysis for FPTS is presented and proved in Chapter 6. Next, a novel lower bound for the response time of a task is presented in Chapter 7. Chapter 8 presents the results of the evaluation of our novel lower bound with other trivial lower bounds for the response time of a task with respect to its exact best-case response time. We conclude this Master's thesis in Chapter 9. Finally, in the Appendix, we include some auxiliary lemmas and proofs for additional theorems. In addition, we briefly describe the tools that were developed during the realization of the Master's project and include some examples applying our novel best-case analysis.
Chapter 2

Real-Time Scheduling Model

In this chapter, we present the real-time scheduling models for FPPS and FPTS along with some related notions.

2.1 Basic task model

We assume a single processor and a set \( T \) of \( n \) independent periodic tasks \( \tau_1, \tau_2, ..., \tau_n \) with unique and fixed priorities \( \pi_1, \pi_2, ..., \pi_n \in \mathbb{N}^+ \). We assume that tasks are given in order of decreasing priority, i.e. \( \tau_1 \) has the highest priority whereas \( \tau_n \) has the lowest priority. A higher priority is represented by a higher value, i.e. \( \pi_1 > \pi_2 > ... > \pi_n \).

Each task \( \tau_i \) generates an infinite sequence of jobs \( i_i, k \) with \( k \in \mathbb{Z} \). In addition, each task is characterized by a period \( T_i \in \mathbb{R}^+ \), a worst-case computation time \( WC_i \in \mathbb{R}^+ \), a best-case computation time \( BC_i \in \mathbb{R}^+ \), where \( BC_i \leq WC_i \), a phasing \( \varphi_i \in \mathbb{R} \), a (relative) worst-case deadline \( WD_i \in \mathbb{R}^+ \), and a (relative) best-case deadline \( BD_i \in \mathbb{R}^+ \cup \{0\} \), where \( BD_i \leq WD_i \). We assume arbitrary deadlines; hence, deadline \( WD_i \) may be smaller than, equal to, or larger than period \( T_i \). A job of a task activated at time \( t = \varphi_i \) serves as a reference activation, and it is referred as the job zero. The set of phasings \( \varphi_i \) is termed the phasing \( \varphi \) of the task-set \( T \). We also assume that we do not have control over the initial phasing \( \varphi \), i.e. arbitrary phasing may occur. In addition, we adopt the same basic assumptions as [13], i.e. tasks do not suspend themselves, jobs do not start before the completion of previous jobs of the same task, and the overhead of context switching and task scheduling is ignored. We sometimes use \( C_i \) to express the computation time of a task \( \tau_i \) when \( BC_i = WC_i \).

2.2 Fixed priority scheduling algorithms

In this section, we present the fixed priority scheduling models for FPPS and FPTS.

2.2.1 Basic scheduling concepts

A schedule is an assignment of tasks into a processor. A schedule for a set \( T \) of \( n \) independent tasks can be defined as a step function \( \sigma : \mathbb{R} \rightarrow \{0, ..., n\} \) [5] in which \( \sigma(t) = i \) with \( i > 0 \) indicates that task \( \tau_i \) is being executed at time \( t \) whereas \( \sigma(t) = 0 \) indicates that the processor is idle at that time. A preemptive schedule allows a task to be interrupted at any moment in time to assign the processor to another task according to a predefined rule. The set of rules that determine which task executes in the processor at any given time is called scheduling algorithm.

2.2.2 FPPS Scheduling

In fixed-priority preemptive scheduling (FPPS), the scheduler guarantees that, at any given moment in time, the processor executes the task with the highest priority among all the tasks with pending

Best-case response time of real-time tasks under FPTS
work. FPPS scheduling is de facto standard in the industry, and most of the available real-time operating systems (RTOS) implement preemptive scheduling.

2.2.3 Refined model for FPTS

For fixed-priority scheduling with preemption thresholds (FPTS), each task \( \tau_i \) has an additional property called preemption threshold denoted by \( \theta_i \in \mathbb{N}^+ \), where \( \pi_1 \geq \theta_i \geq \pi_i \). A task \( \tau_i \) scheduled under FPTS can be preempted by a higher priority task \( \tau_h \) if and only if (iff) \( \pi_h > \theta_i \). Note that FPPS is a special case of FPTS when the preemption threshold of each task is equal to its priority, i.e. when \( \forall 1 \leq i \leq n \theta_i = \pi_i \). Furthermore, FPTS becomes equivalent to FPNS when all the preemption thresholds are set to the highest priority \( \pi_1 \), i.e. when \( \forall 1 \leq i \leq n \theta_i = \pi_1 \).

Compared to FPPS, FPTS reduces memory requirements, the cost of arbitrary preemptions, and it has been shown to improve the schedulability ratio of task-set [6]. FPTS has been viewed as a potential successor to FPPS as a de facto standard in industry.

2.3 Related notions and schedulability

In this section, we formally introduce some concepts regarding the execution of a task, such as its activation, starting time and finalization. Furthermore, we assume that deadlines are hard, i.e. each job of a task must be completed within its best-case and worst-case deadlines. We therefore, introduce an schedulability criterion to determine whether a task-set \( \mathcal{T} \) is schedulable.

2.3.1 Derived notions

The response time \( R_{i,k} \) of a job \( k \) of task \( \tau_i \) is defined as the time elapsed between its (absolute) finalization time \( f_{i,k} \) and (absolute) activation time \( a_{i,k} \), i.e. \( R_{i,k} = f_{i,k} - a_{i,k} \). The (absolute) start time \( s_{i,k} \) is the time at which job \( k \) of \( \tau_i \) starts its execution. Furthermore, the relative start time \( S_{i,k} \) is the start time relative to the activation of a job, i.e. \( S_{i,k} = s_{i,k} - a_{i,k} \). Finally, we define the hold time \( H_{i,k} \) as the time elapsed between the start of a job and its completion, this is expressed as \( H_{i,k} = f_{i,k} - s_{i,k} \). In general, under this model, it always holds that \( R_{i,k} = S_{i,k} + H_{i,k} \). Figure 2.1 shows the basic task model with these notions.

![Figure 2.1: Basic model for a periodic task \( \tau_i \).](image)

The worst-case response time \( WR_i \) and the best-case response time \( BR_i \) of a task \( \tau_i \) are defined as the longest and the shortest response times of its jobs respectively, i.e.

\[
WR_i \triangleq \sup_{\varphi} R_{i,k}(\varphi) \quad (2.1)
\]

\[
BR_i \triangleq \inf_{\varphi} R_{i,k}(\varphi) \quad (2.2)
\]

where \( R_{i,k}(\varphi) \) denotes a dependency of response time \( R_{i,k} \) on phasing \( \varphi \).
Finally, we define the worst-case utilization $U_T$ and the best-case utilization $BU_T$ as the worst and best fraction of the processor time spent on the execution of $T$ respectively [13], i.e.

$$U_T^\text{def} = \sum_{1 \leq i \leq n} \frac{WC_i}{T_i}, \quad (2.3)$$

$$BU_T^\text{def} = \sum_{1 \leq i \leq n} \frac{BC_i}{T_i}. \quad (2.4)$$

### 2.3.2 Schedulability criterion

We say that a set $T$ of $n$ periodic tasks is schedulable iff the jobs of all tasks complete within their lower (best-case) and upper (worst-case) deadlines, i.e. iff

$$\forall_{1 \leq i \leq n} (BD_i \leq BR_i \land WR_i \leq WD_i). \quad (2.5)$$

### 2.3.3 Hyperperiod

The hyperperiod is defined as the length of the shortest time interval in which an schedule repeats itself after an initial start-up. For a set of $n$ periodic tasks $T$, Leung and Merrill proved in [12] that the hyperperiod $P^T$ can be calculated as the least common multiple (lcm) of the periods of the tasks in $T$, i.e. $P^T = \text{lcm}(T_1, \ldots, T_n)$.

### 2.4 An example

Table 2.1 shows a basic example of a set $T_{2.1}$ of three tasks. As can be seen, $\tau_1$ has the highest priority and can preempt to tasks $\tau_2$ and $\tau_3$. Furthermore, $\tau_2$ cannot preempt $\tau_3$ because $\theta_3 = \pi_2$. Figure 2.2 shows a timeline for the execution of the tasks in $T_{2.1}$. As can be seen, the schedule repeats itself every hyperperiod $P_{T_{2.1}} = 24$. Furthermore, note that the first job of $\tau_2$ has a response time $R_{2.0} = 9$ and misses its worst-case deadline $WD_2 = 8.5$. Therefore, the set of tasks $T_{2.1}$ is not schedulable.

<table>
<thead>
<tr>
<th>task $\tau_i$</th>
<th>$T_i$</th>
<th>$WD_i$</th>
<th>$BD_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>8</td>
<td>8.5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>24</td>
<td>28</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{T_{2.1}} = 24$ and $U_{T_{2.1}}^T = BU_{T_{2.1}}^T = 0.875$.

Figure 2.2: Timeline for $T_{2.1}$ when $\varphi_1 = 1$, $\varphi_2 = 2$ and $\varphi_3 = 0$. Curly red arrows indicate that a job missed its worst-case deadline.
Chapter 3

Recap of Existing Analysis

In this chapter, we briefly recapitulate on the existing worst-case response time analysis for FPPS and FPTS. In addition, we also recapitulate the exact best-case response time analysis for FPPS.

3.1 Worst-case response time analysis for FPPS

3.1.1 A critical instant

A critical instant is defined as the (hypothetical) instant that leads to the worst-case response time of a task [13]. For FPPS, a critical instant for a task \( \tau_i \) occurs when it experiences a simultaneous activation with all its higher priority tasks. Furthermore, for constraint deadlines, i.e. for worst-case deadlines at most equal to periods, the worst-case response time of \( \tau_i \) in an schedulable task-set is always found in the first job after a critical instant because such a job experiences the largest interference by its higher priority tasks. On the other hand, when assuming arbitrary deadlines, the first job after a critical instant may induce some delay in the start time of subsequent jobs of \( \tau_i \) and, therefore, the worst-case response time may be assumed by one of such subsequent jobs. As an example of this situation, consider the set of three tasks with characteristics as described in Table \( T_3 \).

Figure 3.1 shows a critical instant for task \( \tau_3 \). Note that there is a simultaneous activation of \( \tau_3 \) with its higher priority tasks at time \( t = 0 \). As can be seen, the first job of \( \tau_3 \) has a response time \( R_{3,0} = 9 \). On the other hand, its third job has a response time \( R_{3,2} = 11 \). Therefore, we can conclude that for arbitrary deadlines, it is necessary to explore more than one job after the critical instant in order to determine the worst-case response time of a task.

Table 3.1: Task characteristics of \( T_3 \).

<table>
<thead>
<tr>
<th>task</th>
<th>( T_i )</th>
<th>( WD_i )</th>
<th>( C_i )</th>
<th>( \pi_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>11</td>
<td>10</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>8</td>
<td>14</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P_{T_3} = 792 \) and \( U_{T_3} \approx 0.95 \).

It only remains to determine how many jobs after a critical instant is sufficient to explore in order to find the worst-case response time of a task. Since the schedule repeats itself every hyperperiod, a possible answer is the number of jobs in an hyperperiod. For our previous example, this would be \( \frac{P_{T_3}}{\tau_3} = \frac{792}{3} = 99 \) jobs for task \( \tau_3 \). However, when periods are irrational numbers, the hyperperiod does not necessarily exists.

A more suitable number for the jobs to explore is bounded by the worst-case number of jobs in a level-i active period. Informally, a level-i active period is a time interval in which there is always pending work by \( \tau_i \) or higher priority tasks. In Figure 3.1, the level-3 active period after the
critical instant is \([0, 32)\). Note that at time \(t = 32\), there is no pending work by a job activated before that time; hence, the level-3 active period ends. In the following section, we formally introduce the definition of level-\(i\) active period and present a method to derive its worst-case length.

### 3.1.2 Level-\(i\) active period

In order to formally define the notion of level-\(i\) active period [21], we first introduce the notion of pending load. The pending load \(P_{\tau_i}(t)\) of a task \(\tau_i\) is defined as the amount of processing at time \(t\) that still needs to be completed for the jobs with a priority higher than or equal to task \(\tau_i\) that are activated before time \(t\). A level-\(i\) active period is then defined as the time interval \([t_s, t_e)\) such that \(P_{\tau_i}(t_s) = 0\), \(P_{\tau_i}(t_e) = 0\), and \(P_{\tau_i}(t) > 0\) for all \(t \in (t_s, t_e)\).

For FPPS, the worst-case length \(WL_{\tau_i}\) of a level-\(i\) active period is the smallest \(x \in \mathbb{R}^+\) satisfying

\[
x = \sum_{1 \leq j \leq i} \left\lfloor \frac{x}{T_j} \right\rfloor WC_j.
\]

(3.1)

\(WL_{\tau_i}\) can be found by the following iterative procedure.

\[
\begin{align*}
WL_{\tau_i}^{(0)} &= \sum_{1 \leq j \leq i} WC_j, \\
WL_{\tau_i}^{(l+1)} &= \sum_{1 \leq j \leq i} \left\lfloor \frac{\WL_{\tau_i}^{(l)}}{T_j} \right\rfloor WC_j, \quad l = 0, 1, \ldots
\end{align*}
\]

(3.2)

The procedure is stopped when the same value is found for two successive iterations of \(l\), yielding \(WL_{\tau_i}\). This procedure is guaranteed to terminate when the utilization of the task-set \(U_T\) is less than one, i.e. \(U_T < 1\), or when the utilization is equal to one and the least common multiple of the periods of all the tasks in \(T\) exists.

Finally, the worst-case number of jobs \(w\ell_{\tau_i}\) of a task \(\tau_i\) in a level-\(i\) active period is given by

\[
w\ell_{\tau_i} = \left\lceil \frac{WL_{\tau_i}}{T_i} \right\rceil.
\]

(3.3)

As an example, consider again task-set \(T_{3,1}\) in Table 3.1. Applying the analysis in this section, we find that the worst-case length of a level-3 active period is \(WL_{\tau_3} = 32\) and the worst-case number of jobs in a level-3 active period is \(w\ell_{\tau_3} = \left\lceil \frac{32}{8} \right\rceil = 4\). Hence, it is sufficient to consider at most four jobs of task \(\tau_3\) after a critical instant in order to find its worst-case response time. Table 3.2 shows the intermediate results of the iterative procedure to determine \(WL_{\tau_3}\). Furthermore, when considering the first four jobs of \(\tau_4\) after a critical instant in Figure 3.1, the one with the largest response time is the third job starting at time \(t = 16\) with a response time \(R_{3,2} = 11\). Hence, this is the worst-case response time of \(\tau_3\).

![Figure 3.1: Timeline for \(T_{3,1}\) with a critical instant at time \(t = 0\) for task \(\tau_3\).](image)

Table 3.2: Intermediate results of the iterative procedure to determine \(WL_{\tau_3}\) of \(T_{3,1}\).

<table>
<thead>
<tr>
<th>iteration</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(WL_{\tau_3})</td>
<td>9</td>
<td>11</td>
<td>14</td>
<td>18</td>
<td>20</td>
<td>23</td>
<td>27</td>
<td>29</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

The result is found when the same value is given by two consecutive iterations.
3.1.3 Worst-case response times

The worst-case response time $WR_i$ of task $\tau_i$ is always found in the level-$i$ active period that starts at a critical instant. We now can express the worst-case response time as the longest response time among all the jobs in a level-$i$ active period, i.e.

$$WR_i = \max_{0 \leq k < w_i} WR_{i,k},$$

where $WR_{i,k}$ denotes the worst-case response time of job $t_{i,k}$. The latter is given by

$$WR_{i,k} = F_{i,k} - k \cdot T_i,$$

where $F_{i,k}$ is the worst-case finalization time of job $t_{i,k}$ relative to the start of the level-$i$ active period. $F_{i,k}$ can be found by the smallest $x \in \mathbb{R}^+$ satisfying

$$x = (k + 1)WC_i + \sum_{1 \leq j < i} \left\lceil \frac{x}{T_j} \right\rceil WC_j.$$  

This recursive equation can be solved using a similar approach as the one given for (3.1), namely by means of an iterative procedure starting with a lower bound.

We now apply the analysis presented in this section in our example. Table 3.3 shows the intermediate results of the worst-case analysis for task $\tau_3 \in \mathcal{T}_3$, i.e. the finalization times and worst-case response times for all the jobs in the level-3 active period after the critical instant. The resulting worst-case response time for task $\tau_3$ is therefore $WR_3 = 11$. Furthermore, it is assumed by the third job after the critical instant as shown in Figure 3.1.

Table 3.3: Intermediate results in the worst-case response time analysis for task $\tau_3 \in \mathcal{T}_3$.  

<table>
<thead>
<tr>
<th>$k$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{3,k}$</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>$WR_{3,k}$</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

The worst-case response time is highlighted in bold.

3.2 Worst-case response time analysis for FPTS

In this section, we recapitulate on the exact worst-case response time analysis for FPTS given in [10]. Similar to FPPS, in order to determine the worst-case response time of a task $\tau_i$ using FPTS scheduling, it is necessary to consider all the jobs that occur in the level-$i$ active period after a critical instant. In FPTS, a critical instant of a task $\tau_i$ occurs when one of its jobs has a simultaneous activation with all its higher priority tasks and, in addition, it experiences a worst-case blocking time. The worst-case blocking time $B_i \in \mathbb{R}^+ \cup \{0\}$ of a task $\tau_i$ is the longest time interval that such a task may be blocked by lower priority tasks and it is given by

$$B_i = \max(0, \max_{\theta_l \geq \pi_i > \pi_l} WC_l).$$

The outer max in (3.7) deals with the situations when the set of values in the inner max is empty.

We can now extend the notion of worst-case length of a level-$i$ active period with the notion of worst-case blocking time as follows. For FPTS, the worst-case length $WL_i$ of a level-$i$ active period is the smallest $x \in \mathbb{R}^+$ satisfying

$$x = B_i + \sum_{1 \leq j \leq i} \left\lceil \frac{x}{T_j} \right\rceil WC_j.$$

Similar to (3.1), $WL_i$ can be found by an iterative procedure starting with a lower bound. The worst-case number of jobs $w_\ell_i$ in a level-$i$ active period is then given by (3.3).
Given a task \( \tau_i \) scheduled using FPTS, its higher priority tasks in \( \mathcal{D}_i = \{ \tau_d \in T | \pi_d \geq \pi_i \} \) cannot preempt \( \tau_i \) but at most can delay its start time. We therefore term these tasks as delaying tasks. On the other hand, the higher priority preempting tasks in \( \mathcal{P}_i = \{ \tau_p \in T | \pi_p > \pi_i \} \) can delay the start time of \( \tau_i \) and, in addition, can preemt it. Note that the start of a job in a level-\( i \) active period after a critical instant can be affected by both preempting tasks and delaying tasks. On the other hand, once a job has started, only preempting tasks can increase its response time. Therefore, in order to determine the worst-case response time of a job, the analysis is split in two parts. First, the worst-case start time of a job is determined, following by its finalization time.

The worst-case start time \( S_{i,k} \) of a job \( \tau_{i,k} \) with \( 0 \leq k < w_l_i \) of a task \( \tau_i \) relative to the start of the level-\( i \) active period after a critical instant is given by the smallest \( x \in \mathbb{R}^+ \) satisfying

\[
    x = \begin{cases} \newline
    B_i + kWC_i + \sum_{k,T_h \in \mathcal{P}_i \cup \mathcal{L}_i} \left( \left\lfloor \frac{x}{T_h} \right\rfloor WC_h \right) & \text{for } B_i > 0 \\
    kWC_i + \sum_{k,T_h \in \mathcal{P}_i \cup \mathcal{L}_i} \left( \left\lceil \frac{x}{T_h} \right\rceil + 1 \right) WC_h & \text{for } B_i = 0
    \end{cases}
\]

The worst-case finalization time \( F_{i,k} \) of job \( k \) with \( 0 \leq k < w_l_i \) of a task \( \tau_i \) relative to the start of the level-\( i \) active period is then given by the smallest \( x \in \mathbb{R}^+ \) that satisfies

\[
    x = \begin{cases} \newline
    S_{i,k} + WC_i + \sum_{p:T_h \in \mathcal{P}_i} \left( \left\lfloor \frac{x}{T_h} \right\rfloor - \left\lfloor \frac{S_{i,k}}{T_h} \right\rfloor \right) WC_h & \text{for } B_i > 0 \\
    S_{i,k} + WC_i + \sum_{p:T_h \in \mathcal{P}_i} \left( \left\lfloor \frac{x}{T_h} \right\rfloor - \left\lceil \frac{S_{i,k}}{T_h} \right\rceil + 1 \right) WC_h & \text{for } B_i = 0
    \end{cases}
\]

Finally, similar to the worst-case analysis for FPPS, we can use (3.5) to determine the worst-case response time of a job in the level-\( i \) active period and (3.4) to determine the actual worst-case response time of task \( \tau_i \).

<table>
<thead>
<tr>
<th>Table 3.4: Task characteristics of ( T_{3,4} ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
</tr>
<tr>
<td>( \tau_2 )</td>
</tr>
<tr>
<td>( \tau_3 )</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P^{T_{3,4}} = 1400 \) and \( U^{T_{3,4}} \approx 0.98 \).

As an example of the analysis, consider the task-set \( T_{3,4} \) described in Table 3.4. Based on the analysis presented in Section 3.1.2, we find a worst-case length \( WL_3 = 200 \) of the level-3 active period with \( w_l_3 = \frac{200}{70} = 3 \) activations of task \( \tau_3 \). Furthermore, after applying the worst-case response time analysis for FPTS, we find that the worst-case blocking time for task \( \tau_3 \) is \( B_3 = 0 \) and its worst-case response time is \( WR_3 = 80 \). Table 3.5 shows the intermediate results in the analysis. In addition, Figure 3.2 shows a timeline for \( T_{3,4} \) where the critical instant for task \( \tau_3 \) occurs at time zero. As can be seen, the worst-case response time for task \( \tau_3 \) is assumed for the second jobs in the level-3 active period.

<table>
<thead>
<tr>
<th>Table 3.5: Intermediate results in the worst-case response time analysis for task ( \tau_3 \in T_{3,4} ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
</tr>
<tr>
<td>( S_{3,k} )</td>
</tr>
<tr>
<td>( F_{3,k} )</td>
</tr>
<tr>
<td>( WR_{3,k} )</td>
</tr>
</tbody>
</table>

The worst-case response time is highlighted in bold.
3.3 Best-case response time analysis for FPPS

The best-case response time analysis for FPPS was presented in [20] for arbitrary deadlines. Similar to the worst-case analysis, the best-case analysis is based on an (hypothetical) instant that leads to the best-case response time, i.e. an optimal instant. For FPPS, an optimal instant of a task $\tau_i$ occurs when the completion of one of its jobs coincides with the simultaneous activation of all its higher priority tasks.

It is worth noting that, unlike the worst-case analysis, the job $i_{k^\text{br}}$ experiencing the optimal instant is the job of $\tau_i$ that assumes the best-case response time and, in addition, it also has the shortest hold time. Furthermore, in case of arbitrary deadlines, such a job $i_{k^\text{br}}$ may also experience interference by its previous jobs $i_{\ell}$ with $\ell < k^\text{br}$. Therefore, it is necessary to look at more than one job of $\tau_i$ when determining its best-case response time. To this end, we have to determine intervals of minimal length where complete jobs of $\tau_i$ can execute.

**Definition 1.** The best-case interval $BI_i(y)$ is defined as the length of the shortest interval in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of a task $\tau_i$.

$$x = y + \sum_{1 \leq j < i} \left[ \left\lceil \frac{x}{T_j} \right\rceil - 1 \right] BC_j,$$

where the notation $w^+$ stands for $\max(w, 0)$. $BI_i(y)$ can be found by an iterative procedure starting with an upper bound, e.g. the worst-case response time of $\tau_i$ for a computation time $y$.

In FPPS, the best-case response time of a task $\tau_i$ is always found for the last job of a level-$i$ active period. Therefore, in order to determine its best-case response time, it is necessary to investigate whether such a job may experience interference induced by its previous jobs $i_{\ell}$. Similarly to the worst-case response time analysis, it is sufficient to investigate at most $w_i \ell_i$ jobs.

Therefore, the best-case response time $BR_i$ of a task $\tau_i$ can be found as follows:

$$BR_i = \max_{1 \leq k \leq w_i \ell_i} (BI_i(k \cdot BC_i) - (k - 1)T_i).$$

As an example of the analysis, consider the task-set $T_{3.6}$ described in Table 3.6. The values for $w_i \ell_i$ were determined using the analysis in Section 3.1.2. Based on the analysis presented in this section, we find that the best-case response time of task $\tau_3$ is $BR_3 = 30$. Table 3.7 shows the intermediate results in the analysis.

**Table 3.6: Task characteristics of $T_{3.6}$.**

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$w_i \ell_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>40</td>
<td>20</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>50</td>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>70</td>
<td>20</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{T_{3.6}} = 1400$ and $U_{T_{3.6}} \approx 0.98$. 

Figure 3.2: Timeline for $T_{3.4}$ with a critical instant at time $t = 0$ for task $\tau_3$. 
Table 3.7: Intermediate results in the worst-case response time analysis for task $\tau_3 \in T_{3.6}$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BI_3(k \cdot BC_3)$</td>
<td>20</td>
<td>90</td>
<td>170</td>
</tr>
<tr>
<td>$BI_3(k \cdot BC_3) - (k - 1)T_i$</td>
<td>20</td>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>

The best-case response time is highlighted in bold.

Figure 3.3 shows a timeline for $T_{3.6}$ where an optimal instant for task $\tau_3$ occurs at time $t = 0$. As can be seen, the best-case response time for task $\tau_3$ is assumed for the job experiencing the optimal instant. Furthermore, the job of $\tau_3$ activated at time $t = -170$ induces some delay in the job experiencing the optimal instant. It is worth noting that the response time of the last job of $\tau_3$ cannot be reduced because three jobs of $\tau_3$ cannot fit in an interval smaller than $(-170, 0)$.

![Timeline for $T_{3.6}$](image-url)

Figure 3.3: Timeline for $T_{3.6}$ with an optimal instant at time $t = 0$ for task $\tau_3$. 

Best-case response time of real-time tasks under FPTS
Chapter 4

Non-preemptive Tasks and Lower Priority Tasks

Since FPNS is a special case of FPTS when thresholds of tasks are set to the highest priority, we first investigate the best-case response time for non-preemptive tasks as an intermediate step towards the best-case response time analysis for FPTS. In this chapter, we prove that the best-case response time of a non-preemptive task is equal to its best-case computation time. In addition, we prove that the best-case response time of a task scheduled using FPTS is not influenced by its lower priority tasks nor by the thresholds of its higher priority tasks.

4.1 Best-case response time for non-preemptive tasks

In this section, we present our results given in [18] regarding the best-case response time of non-preempting tasks.

**Lemma 1.** Let $\tau_i$ be a non-preemptive task, i.e. $\theta_i = \pi_1$, and let $BU_T < 1$ or $BU_T = 1$ and the lcm of the periods exists. The best-case response time of $\tau_i$ is always equal to its best-case computation time, i.e. $BR_i = BC_i$.

**Proof.** Given a task-set $T$ and a non-preemptive task $\tau_i \in T$, assume that all jobs of $\tau_i$ have a computation time equal to $BC_i$. Since $\tau_i$ is non-preemptive, the hold time of all its jobs is simply equal to $BC_i$, i.e. $H_{i,k} = BC_i$. Therefore, in order to prove the lemma, it is sufficient to show that it is always possible to schedule task $\tau_i$ in such a way that the relative start time of a job $\iota_{i,k}$ of $\tau_i$ becomes zero, i.e. $S_{i,k} = 0$. We divide this proof in two cases:

1. **Case $BU_T < 1$.** Given an arbitrary schedule for $T$, let $[t_s, t_e)$ with $t_e > t_s$ be an idle interval in such a schedule, where an idle interval is a time interval in which there is no pending load. Note that it is always possible to find an idle interval because $BU_T < 1$. Furthermore, let $i_{i,k}$ be the first job of $\tau_i$ activated at or after time $t_e$, i.e. $t_e \leq a_{i,k}$. We now show that, by pushing the activation of $i_{i,k}$ to occur in the interval $[t_s, t_e)$, the relative start time of $i_{i,k}$ becomes zero.

   First observe that there is no other job of $\tau_i$ between the idle interval $[t_s, t_e)$ and $i_{i,k}$. Hence, after pushing the activations of $\tau_i$ till $a_{i,k}$ occurs in $[t_s, t_e)$, job $i_{i,k}$ does not experience blocking by a previous job. Furthermore, since $[t_s, t_e)$ was originally idle, job $i_{i,k}$ can start upon activation, leading to $S_{i,k} = 0$.

2. **Case $BU_T = 1$ and the lcm of the periods exists.** First, observe that since the lcm of the periods exists, the hyperperiod $P^T$ exists as well. Let $n_i = P^T / T_i$ be the number of jobs of $\tau_i$ in every hyperperiod $P^T$ and assume that no job of $\tau_i$ starts upon activation. Hence, for each interval $[a_{i,k}', s_{i,k'})$ with $1 \leq k' \leq n_i$, there are only higher priority tasks of $\tau_i$ and/or at most one blocking lower priority task executing within such an interval. As an example of this situation, consider the timeline...
depicted in Figure 4.1 for task-set $T_{4.1}$ described in Table 4.1. As can be seen, no job of $\tau_2$ can start upon activation.

Table 4.1: Task characteristics of $T_{4.1}$.

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$WD_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{T_{4.1}} = 12$ and $BU_{T_{4.1}} = 1$.

Now let $i_{i,b}$ be the job of $\tau_i$ with the shortest relative start time among all jobs of $\tau_i$ in an hyperperiod. Since no job of $\tau_i$ starts upon activation, we can push the activation of all jobs of $\tau_i$ to a later moment in time, without modifying the schedule, till job $i_{i,b}$ starts upon activation. After pushing the activations of $\tau_i$, it holds that $S_{i,b} = 0$ therefore concluding the proof. Figure 4.2 shows an example where the schedule shown in Figure 4.1 is preserved after pushing the activations of $\tau_2$ till its second job starts upon activation.

Figure 4.1: A timeline for $T_{4.1}$. The activations of the tasks are repeated in every interval with length $P_{T_{4.1}} = 12$.

Figure 4.2: A timeline for $T_{4.1}$. After pushing the activations of $\tau_2$, the second job can immediately start.

4.2 Influence of lower priority tasks

In FPTS, a lower priority task $\tau_b$ may affect the response time of a higher priority task $\tau_i$ when $\theta_b > \pi_i$. In this section, we prove that lower priority tasks do not have an influence in the best-case response time of a task however.

**Lemma 2.** Given a task-set $\mathcal{T}$ of $n$ independent tasks scheduled using FPTS, and a job $i_{i,b}$ of a task $\tau_i \in \mathcal{T}$. Let $\tau_{n+1}$ be a new task which priority is lower than all the priorities of the tasks in $\mathcal{T}$. Furthermore, let $U^{r_{\mathcal{T} \cup \{\tau_{n+1}\}} < 1}$ or $U^{r_{\mathcal{T} \cup \{\tau_{n+1}\}} = 1}$ and the lcm of the periods exists. It is always possible to schedule a task $\tau_{n+1}$ in such a way that the schedule for job $i_{i,b}$ does not change after introducing $\tau_{n+1}$; hence, its response time $R_{i,b}$ remains the same.
Proof. Given a schedule for task-set \( T \), observe that job \( i, k \) executes in the interval \([s_{i,k}, f_{i,k}]\). Therefore, in order to prove the lemma, it is sufficient to prove that the schedule in \([s_{i,k}, f_{i,k}]\) remains the same after introducing \( \tau_{n+1} \).

Let \( \beta_i = \{ \beta_b \in T | \theta_b \geq \pi_i > \pi_b \} \) be the set of lower priority tasks that cannot be preempted by task \( \tau_i \), i.e. the blocking tasks of \( \tau_i \). Furthermore, assume that task \( \tau_{n+1} \) is scheduled in such a way that a job \( k' \) of \( \tau_{n+1} \) executes within an interval \([t_{1,i,k}, s_{i,k}]\) where there is exactly an amount of time \( C_{n+1} \) available for the execution of \( \tau_{n+1} \). After introducing \( \tau_{n+1} \), the schedule in \([s_{i,k}, f_{i,k}]\) can be guaranteed to remain unchanged if there is no pending load at time \( s_{i,k} \) from higher priority tasks of \( \tau_i \), i.e. if \( P_{t-1}(s_{i,k}) = 0 \). In addition, all jobs of blocking tasks and previous jobs of \( \tau_i \) starting before \( s_{i,k} \) should finalize before or at that time. Finally, there should not be pending load at time \( s_{i,k} \) from a job of \( \tau_{n+1} \) that has already started. We show that these conditions are always met for the given assumptions.

\( [P_{t-1}(s_{i,k}) = 0 \text{ and previous jobs of } \tau_i \text{ as well as its blocking jobs that start before } s_{i,k} \text{ finalize before or at that time} \). First note that, since there is exactly an amount of time \( C_{n+1} \) available for the execution of \( \tau_{n+1} \) in the interval \([t_{1,i,k}, s_{i,k}]\), job \( k' \) of \( \tau_{n+1} \) can fit exactly in such an interval. Therefore, the higher priority tasks executing in \([t_{1,i,k}, s_{i,k}]\) will still finalize before or at \( s_{i,k} \) after introducing \( \tau_{n+1} \) into the schedule, and consequently the pending load \( P_{t-1}(s_{i,k}) \) will still be zero. Furthermore, the same holds for blocking jobs and previous jobs of \( \tau_i \) executing in such an interval \([t_{1,i,k}, s_{i,k}]\); hence, proving this case.

\( \{ \text{Pending load of already started jobs of } \tau_{n+1} \text{ at time } s_{i,k} \text{ is zero} \} \). Since we assumed that the job \( k' \) of \( \tau_{n+1} \) executes within \([t_{1,i,k}, s_{i,k}]\) and therefore it finalizes before or at \( s_{i,k} \), we only have to prove that next jobs of \( \tau_{n+1} \) cannot start before \( s_{i,k} \). Note that such a job \( k' \) already occupies all the time that was available in \([t_{1,i,k}, s_{i,k}]\). Hence, there is no time available for a job of \( \tau_{n+1} \), other than job \( k' \), in the interval \([t_{1,i,k}, f_{i,k}]\). We therefore conclude that the start time of any job of \( \tau_{n+1} \) after job \( k' \) activated before \( s_{i,k} \) will be postponed after \( f_{i,k} \).

We have proven that whenever a job \( k' \) of \( \tau_{n+1} \) executes within an interval \([t_{1,i,k}, s_{i,k}]\) where there is exactly an amount of time \( C_{n+1} \) available for the execution of \( \tau_{n+1} \), the schedule in \([s_{i,k}, f_{i,k}]\) is preserved. It only remains to prove that it is always possible to schedule a job of \( \tau_{n+1} \) in such an interval \([t_{1,i,k}, s_{i,k}]\). Note that the length of this interval is not necessarily equal to \( C_{n+1} \) but it has to contain exactly an amount of time \( C_{n+1} \) available for \( \tau_{n+1} \).

Since \( U^{[t_{1,i,k}]} < 1 \) or \( U^{[\tau_{n+1}]} = 1 \) and the lcm of the periods exists, it is always possible to find an interval \([t_{1,i,k}, s_{i,k}]\) before \([s_{i,k}, f_{i,k}]\) where there is exactly \( C_{n+1} \) time available for the execution of \( \tau_{n+1} \). Furthermore, the only reason for which a job \( k' \) of \( \tau_{n+1} \) may not be able to execute within \([t_{1,i,k}, s_{i,k}]\) is that it could experience some interference induced by its previous jobs that delays its start time \( s_{n+1,k} \). Therefore, we only have to show that it is always possible to find enough space before \([t_{1,i,k}, s_{i,k}]\) for the execution of previous jobs of \( \tau_{n+1} \) that may delay \( s_{n+1,k} \). In this way, job \( k' \) can execute within \([t_{1,i,k}, s_{i,k}]\) without interference of its previous jobs.

Note that the jobs of \( \tau_{n+1} \) that can delay the start time of job \( k' \) of \( \tau_{n+1} \) are the jobs in its level \((n+1) \) active period. In addition, note that, under the given assumptions, the level \((n+1) \) active period is always finite and, therefore, the number of jobs that can delay \( s_{n+1,k} \) is finite as well. Since it is always possible to find an interval before \([t_{1}, t_{2}]\) with enough processing time available for the execution of a finite number of jobs of \( \tau_{n+1} \), we conclude that job \( k' \) can always be scheduled within \([t_{1}, s_{i,k}]\). \[\square\]
Theorem 1. Let all tasks of a set $T$ be strictly periodic and let $U^T < 1$ or $U^T = 1$ and the lcm of the periods exists. Under FPTS, the best-case response time $BR_i$ of a task $τ_i \in T$ is not influenced by its lower priority tasks.

Proof. Assume that we find a schedule for a subset of task-set $T$ defined as $T_i = \{τ_a \in T | π_a ≥ π_i\}$ where the best-case response time $BR_i$ of $τ_i$ is assumed by its job $k_{br}$, i.e. $R_{i,k_{br}} = BR_i$. Note that task-set $T_i$ contains the same tasks as $T$ without the tasks with a lower priority than $τ_i$. As an example of this situation, consider the set $T_{4,2}$ of four tasks with characteristics as described in Table 4.2. Figure 4.3 shows a timeline for $T_{4,2} \setminus \{τ_3, τ_4\}$ in which the best-case response time $BR_2 = 6$ of task $τ_2$ is assumed by the job starting at time $t = 0$. Given Lemma 2, it is always possible to construct a schedule for a task-set $T_{i+1}$ where task $τ_{i+1}$ is scheduled in such a way that the response time $R_{i,k_{br}}$ remains the same. Figure 4.4 shows this scenario when introducing task $τ_3$ into the schedule for $T_{4,2}$. Applying Lemma 2 again, we can find a schedule for a task-set $T_{i+2}$ where the response time $R_{i,k_{br}}$ is preserved after introducing a task $τ_{i+2}$. Since we can continue constructing schedules preserving the response time $R_{i,k_{br}}$ until $T_{i+l} = T$, we conclude that the best-case response time of a task is not affected by its lower priority tasks. Figure 4.5 shows that indeed the best-case response time of $τ_2$ is preserved after introducing the lowest priority task $τ_4$ into the schedule for $T_{4,2}$. 

Table 4.2: Task characteristics of $T_{4,2}$.

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$π_i$</th>
<th>$θ_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$τ_1$</td>
<td>10</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$τ_2$</td>
<td>16</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$τ_3$</td>
<td>40</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$τ_4$</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{T_{4,2}} = 80$ and $U_{T_{4,2}} = 0.975$.

Figure 4.3: A timeline for $T_{4,2} \setminus \{τ_3, τ_4\}$ where the best-case response time for $τ_2$ is found in the time interval $[0, 6]$.

Figure 4.4: A timeline for $T_{4,2} \setminus \{τ_4\}$ where the best-case response time of $τ_2$ is preserved after introducing a job of $τ_3$ in the time interval $[-6, 0)$ before the job of $τ_2$ starting at time $t = 0$. Note that this interval has exactly an amount of time $C_3 = 4$ available for $τ_3$. 

Best-case response time of real-time tasks under FPTS
Figure 4.5: A timeline for $T_{1.2}$ where the best-case response time of $\tau_2$ is preserved after introducing a job of $\tau_4$ in the time interval $[-17,0)$. Note that this interval has exactly an amount of time $C_4 = 3$ available for $\tau_4$.

4.3 Influence of thresholds of higher priority tasks

In this section, we investigate the influence of the thresholds of higher priority tasks in the best-case response time.

**Lemma 3.** Let $\mathcal{T}$ be a set of strictly periodic tasks with a task $\tau_i \in \mathcal{T}$. The thresholds of the higher priority tasks of $\tau_i$ do not influence the response time of its jobs.

**Proof.** In order to prove Lemma 3, first note that the response time $R_{i,k}$ of a job $\iota_{i,k}$ of $\tau_i$ is equal to $S_{i,k} + H_{i,k}$. Therefore, it is sufficient to show that the thresholds of higher priority tasks do not have an influence on the start time $S_{i,k}$ and the hold time $H_{i,k}$.

**Influence on $S_{i,k}$**. Under FPTS, a necessary condition for a job $\iota_{i,k}$ to start at a time $t$ after its activation is that the priority of $\tau_i$ is the highest among all tasks with pending load at time $t$. Since this condition only depends on the priority of the tasks, higher priority tasks will always affect the start time $S_{i,k}$ of $\iota_{i,k}$ independently of their thresholds.

**Influence on $H_{i,k}$**. Under FPTS, a job $\iota_{i,k}$ of a task $\tau_i$ can only be preempted by a higher priority task $\tau_h$ iff $\pi_h > \theta_i$. Since this condition is independent of the threshold $\theta_h$ of $\tau_h$, we conclude that the thresholds of higher priority tasks of $\tau_i$ do not have an influence on the hold time $H_{i,k}$ of $\iota_{i,k}$.
Chapter 5

Towards an Optimal Instant

Recall that an optimal instant is the time instant that leads to the best-case response time. The best-case response time analysis for FPPS is based on such an optimal instant that occurs when the finalization of a job of a task coincides with the simultaneous activation of all its higher priority tasks. In this chapter, we derive a property for an optimal instant of a task scheduled using FPTS. To this end, we first introduce some facts regarding the best-case behaviour of a task scheduled using FPTS. Afterwards, we formally identify the tasks that influence on the best-case response time of another task and we prove a property for an optimal instant.

5.1 Introductory examples

In this section, we introduce some examples that show that the best-case response time analysis for FPTS is most likely a non-trivial extension of the existing analysis for FPPS.

In FPTS, only tasks with a priority higher than the threshold \( \theta_i \) can preempt a task \( \tau_i \). In Section 3.2, we denoted this type of tasks as preempting tasks. The remaining higher priority tasks in \( \mathcal{D}_i = \{ \tau_d \in \tau | \theta_i \geq \pi_d > \pi_i \} \), however, can at most delay the start time of \( \tau_i \). From Lemma 1, we can conclude that the best-case response time of a non-preemptive task is independent of higher priority tasks. Based on this, it could be intuitive to think that, given a task \( \tau_i \) scheduled using FPTS, the best-case response time of \( \tau_i \) would be independent of all its higher priority tasks that cannot preempt it, i.e. the tasks in \( \mathcal{D}_i \). However, this is not the case for FPTS as we show in the following example.

Consider the set \( T_{5,1} \) of three tasks with deadlines equal to periods with characteristics as described in Table 5.1. The best-case response times in this table were found using brute force. This is, by considering the phasings \( \varphi_1 \in [0,18) \) and \( \varphi_2 \in [0,24) \) with a granularity of one time unit in the simulations, while fixing the phasing of \( \tau_3 \) to \( \varphi_3 = 0 \). For a more detailed description about the brute force approach in the simulations see Appendix C.3. Furthermore, based on the analysis in Section 3.1.2 and Section 3.2, we found the values for the worst-case number of jobs \( w\ell_i \) in a level-\( i \) active period and the worst-case response times \( WR_i \). Note that task \( \tau_1 \) can preempt task \( \tau_3 \) because \( \pi_1 > \theta_3 \). On the other hand, task \( \tau_2 \) cannot preempt task \( \tau_3 \) because \( \pi_2 = \theta_3 \).

Table 5.1: Task characteristics of \( T_{5,1} \) with worst-case deadlines equal to periods.

<table>
<thead>
<tr>
<th>task</th>
<th>( T_i = WD_i )</th>
<th>( C_i )</th>
<th>( \pi_i )</th>
<th>( \theta_i )</th>
<th>( w\ell_i )</th>
<th>( BR_i )</th>
<th>( WR_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>18</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>24</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>24</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>45</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>38</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P_{T_{5,1}} = 360 \) and \( U_{T_{5,1}} = 0.99 \).
Figure 5.1 shows a timeline for task-set $T_{5,1}$ where the best-case response time of task $\tau_3$ is assumed by its last job in the level-3 active period. As can be seen, such a job cannot immediately start upon activation. On the other hand, when ignoring task $\tau_2$ from task-set $T_{5,1}$, the best-case response time of task $\tau_3$ is reduced to $BR'_3 = 7$. Based on this observation, we conclude that, in general, the best-case response time of a task $\tau_i$ is not independent of the higher priority tasks that cannot preempt $\tau_i$. Therefore, we formulate the following fact.

**Fact 1.** In FPTS, the best-case response time of a task $\tau_i$ can also be affected by a higher priority task that cannot preempt $\tau_i$, i.e. by delaying tasks in $D_i = \{\tau_d \in T | \theta_i \geq \pi_d > \pi_i\}$.

Based on the optimal instant for FPPS, one may think that the best-case response time of a task $\tau_i$ under FPTS is found when the simultaneous activation of all high priority tasks that can preempt $\tau_i$ coincides with the completion of a job of $\tau_i$. Figure 5.1 seems to suggest that this is the case because an activation of $\tau_1$ coincides with the completion of the job of $\tau_3$ that assumes the best-case response time at time $t = 0$. However, we now introduce an example that refutes this initial intuition.

**Table 5.2: Task characteristics of $T_{5,2}$.**

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$WD_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$\theta_i$</th>
<th>$\omega \ell_i$</th>
<th>$WR_i$</th>
<th>$BR_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>35</td>
<td>20</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>35</td>
<td>20</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>50</td>
<td>65</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>62</td>
<td>20</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>70</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>66</td>
<td>27</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{5,2} = 350$ and $U_{5,2} = 1$.

Consider the set $T_{5,2}$ of four tasks with characteristics as described in Table 5.2. Similarly to previous example, the values for the best-case response times were found using brute force. Furthermore, note that tasks $\tau_1$ and $\tau_2$ can preempt $\tau_4$. On the other hand, $\tau_3$ cannot preempt $\tau_4$ because $\pi_3 = \theta_4$. In order to investigate whether the best-case response time of $\tau_4$ can be found when all higher priority preemptive tasks are activated simultaneously, we repeated the simulation using brute force but now fixing the phasings of $\tau_1$ and $\tau_2$ to be always equal, i.e. $\varphi_1 = \varphi_2$. After performing the experiment, we found that the shortest response time of $\tau_4$ under these conditions is $BR'_4 = 32$. Figure 5.2 depicts a timeline where this response time is found for the first job of $\tau_4$ in the level-4 active period. As can be seen, a simultaneous activation of tasks $\tau_1$ and $\tau_2$ coincides with the completion of a job of $\tau_4$ at time $t = 0$. However, the best-case response time $BR_4 = 27$ is not assumed by any of the jobs of $\tau_4$ in the level-4 active period. Figure 5.3 shows a timeline for the same task-set $T_{5,2}$ where the best-case response time of $\tau_4$ is assumed by its first job. Note that the activation of the preempting task $\tau_2$ does not coincide with the finalization time of the first job of $\tau_4$ where the best-case response time is found. Therefore, we propose the following fact as a first witness of dissimilarity between the best-case response time behavior of FPTS and FPPS.
Fact 2. In FPTS, the best-case response time of a task $\tau_i$ is not necessarily found when the simultaneous activation of all higher priority tasks that can preempt task $\tau_i$ coincides with the completion of one of its jobs.

![Figure 5.2](image-url): A timeline for $T_{5,2}$ depicting a level-4 active period. A simultaneous activation of preemptive tasks $\tau_1$ and $\tau_2$ coincides with a completion of the last job of $\tau_4$ in the level-4 active period at time $t = 0$.

Fact 3. The best-case response time under FPTS is not necessarily assumed by the job of $\tau_i$ with the shortest hold time.

![Figure 5.3](image-url): A timeline for $T_{5,2}$ depicting a level-4 active period where the best-case response time of task $\tau_4$ is assumed by the first job.

From Figure 5.3, it is also interesting that, even that the first job of $\tau_4$ experiences a preemption by task $\tau_2$, it still assumes the best-case response time. On the other hand, the last job of $\tau_4$ in Figure 5.2 ending at time $t = 0$ assumes the shortest hold time but not the best-case response time. This differs from the best-case behavior of FPPS, where the best-case response time is always assumed by the job with the shortest hold time. Hence, this is a second witness of dissimilarity.

Fact 4. In FPTS, the best-case response time of a task $\tau_i$ is not necessarily assumed by the last job in a level-4 active period.

Best-case response time of real-time tasks under FPTS
5.2 Tasks influencing the best-case response time

According to Theorem 1, lower priority tasks have no influence on the best-case response time of a task \( \tau_i \). Therefore, we ignore lower priority tasks in the rest of this document. In addition, we distinguish between two types of tasks than can influence the best-case response time of a task \( \tau_i \). These types of tasks are the set \( P_i \) of preempting tasks in \( \{ \tau_p \in T | \pi_p > \theta_i \} \), and the set \( D_i \) of delaying tasks given by Fact 1. We termed the latter type as delaying tasks because they cannot preempt \( \tau_i \) but at most can delay its start time. Furthermore, Fact 3 shows that a job \( k_{br} \) of task \( \tau_i \) that assumes the best-case response time does not necessarily have the shortest hold time. Instead, some preempting tasks may give rise to more preemptions in \( k_{br} \). Note that this is the case for the first job of \( \tau_4 \) starting at time \( t = 0 \) in Figure 5.3. This job assumes the best-case response time; however, it experiences a preemption from \( \tau_2 \). Based on this observation, we divide the set \( P_i \) of preempting tasks of \( \tau_i \) into the set of preempting tasks that give rise to extra preemptions denoted as \( E_i \subseteq P_i \), and the remaining preempting tasks that we call minimal preempting tasks in the set \( M_i = P_i \setminus E_i \).

5.3 A property for an optimal instant

We now formulate a property for an optimal instant for FPTS based on the tasks influencing the best-case response time of a task \( \tau_i \).

**Theorem 2.** In FPTS, an optimal instant for a task \( \tau_i \), where the best-case response \( BR_i \) is assumed by a job \( i_{k_{br}} \), occurs when the completion of \( i_{k_{br}} \) coincides with a simultaneous activation of all its minimal preempting tasks in \( M_i \). Furthermore, the activation of all delaying tasks in \( D_i \) and all extra preempting tasks in \( E_i \), coincides with \( s_{i_{k_{br}}} + \Delta \), where \( \Delta \) is a sufficiently small amount of time.

**Proof.** Let \( k_{br} \) be the job of a task \( \tau_i \) that assumes its best-case response time, i.e. \( R_{i,k_{br}} = BR_i \). Furthermore, let \( \Delta \in \mathbb{R}^+ \) be a sufficiently small amount of time such that all jobs of higher priority tasks in \( P_i \cup D_i \) activated after \( s_{i,k_{br}} \) are activated after \( s_{i,k_{br}} + \Delta \) as well. Figure 5.4 shows an example of a job of \( \tau_i \) that assumes its best-case response time for task-set \( T_{5.3} \). We will now show that it is always possible to construct an instant with the property given in Theorem 2 for such a job \( i_{k_{br}} \) without increasing its response time. To this end, first recall that \( R_{i,k_{br}} = S_{i,k_{br}} + H_{i,k_{br}} \); hence, we will show that the start time and the hold time of \( k_{br} \) do not increase when constructing such an instant with the property for an optimal instant.

Table 5.3: Characteristics of task-set \( T_{5.3} \).

<table>
<thead>
<tr>
<th>task</th>
<th>( T )</th>
<th>( C )</th>
<th>( \pi )</th>
<th>( \theta )</th>
<th>( BR )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_{p1} )</td>
<td>40</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>( \tau_{p2} )</td>
<td>40</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>( \tau_d )</td>
<td>60</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>( \tau_i )</td>
<td>50</td>
<td>20</td>
<td>1</td>
<td>2</td>
<td>27</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P_{5.3} = 600 \) and \( U_{5.3} = 0.975 \).
Let \( k_d \) be the first job of a delaying task \( \tau_d \) activated after \( s_{i,br} + \Delta \); hence, it holds that \( a_{i,k_d} > s_{i,br} + \Delta \). Furthermore, note that the pending load \( P_{i-1}(s_{i,br}) \) of higher priority tasks at time \( s_{i,br} \) is zero; otherwise, job \( i_{br} \) would not be able to start at that moment. Clearly, if we push the activations of \( \tau_d \) to an earlier moment in time till \( a_{d,k_d} = s_{i,br} + \Delta \), the pending load \( P_{i-1}(s_{i,br}) \) remains zero. Hence, the start time cannot increase because job \( i_{br} \) can still start at the same time. Note that \( i_{br} \) cannot start earlier because this would reduce its response time, which is impossible because it assumes the best-case response time. In addition, the hold time cannot increase either because delaying tasks cannot preempt \( i_{br} \). We therefore conclude that changing the phasing of all delaying tasks in such a way that one of its activations occur at time \( s_{i,br} + \Delta \) does not change the response time of \( i_{br} \). Figure 5.5 depicts this situation. As can be seen, the response time of the job of \( \tau_i \) is preserved after pushing task \( \tau_d \) till one of its activations occurs at time \( s_{i,br} + \Delta \).

Similarly, we will now show that it is always possible to modify the phasing of all preempting tasks till one of its activations occurs at time \( s_{i,br} + \Delta \) or at time \( f_{i,br} \); without affecting the response time. To this end, let \( k_e \) be the first job activated after \( s_{i,br} + \Delta \) of a preempting task \( \tau_p \); hence, \( a_{p,k_e} > s_{i,br} + \Delta \). In addition, let \( k_m \) be the first job of \( \tau_p \) activated at or after time \( f_{i,br} \); hence, \( a_{p,k_m} \geq f_{i,br} \). We now divide the proof in two cases.

\( \text{Case:} a_{p,k_e} - (s_{i,br} + \Delta) > a_{p,k_m} - f_{i,br} \) For this case, note that it is possible to push the activations of \( \tau_p \) to an earlier moment in time till \( a_{p,k_m} = f_{i,br} \). After doing so, it still holds that \( a_{p,k_e} > s_{i,br} + \Delta \). Therefore, \( i_{br} \) still experiences the same number of preemptions by \( \tau_p \) and its hold time \( H_{i,br} \) remains unchanged. Furthermore, similarly to the case for delaying tasks, \( i_{br} \) can still start at the same time because the pending load \( P_{i-1}(s_{i,br}) \) remains zero. Therefore, the start time \( S_{i,br} \) do not increase either. Figure 5.6 shows that indeed the response time of the job of \( \tau_i \) remains the same after pushing task \( \tau_h \).
Figure 5.6: A timeline for $T_{5,3}$ depicts the case when the delaying task $\tau_d$ is pushed till one activation coincides with $s_{i,kr} + \Delta$.

\[ \text{Case } a_{p,ke} - (s_{i,kr} + \Delta) \leq a_{p,km} - f_{i,kr} \] Similar to previous case, we push the activations of $\tau_p$ earlier in time till $a_{p,ke} = s_{i,kr} + \Delta$. Note that it still holds that $a_{p,km} \geq f_{i,kr}$. Hence, the hold time $H_{i,kr}$ will remain unchanged because $t_{i,kr}$ still experiences the same number of preemptions. Furthermore, since it still holds that $a_{p,ke} > s_{i,kr}$, job $i_{kr}$ can still start at the same time because there is no pending load of higher priority tasks at that time. Figure 5.7 shows that, after pushing the activations of $\tau_{h2}$, the response time of the job of $\tau_i$ is preserved.

We conclude that the phasing of all preempting tasks can be changed in such a way that, for each of its tasks, one of its activations occurs at time $s_{i,kr} + \Delta$ or at time $f_{i,kr}$ without affecting the response time $R_{i,kr}$.

Figure 5.7 shows an example of this property for an optimal instant under FPTS for task $\tau_i$. As can be seen, the activation of the minimal preempting task $\tau_{p1}$ coincides with the completion of the job of task $\tau_i$ at time $t = 0$. Furthermore, delaying task $\tau_d$ and extra preempting task $\tau_{p2}$ are activated an instant $\Delta$ after the start of $\tau_i$.

It is worth noting that, from the property for optimal instant given above, it is not clear how to partition the set of preempting tasks $\mathcal{P}_i$ into the (sub-)sets of extra preempting $\mathcal{E}_i$ and minimal preempting $\mathcal{M}_i$ tasks. In fact, we could have chosen $\mathcal{E}_i = \{\tau_{p1}\}$ and $\mathcal{M}_i = \{\tau_{p2}\}$ in the example depicted in Figure 5.7, and we would still have obtained an option that is in accordance with Theorem 2. However, this new option does not lead to the best-case response time. Therefore, we can conclude that in FPTS there may be multiple candidates of optimal instants that we have to explore in order...
to find the best-case response time of a task \( r_i \). Clearly, this differs from the best-case analysis for FPPS, where the way to construct an optimal instant is always unique. Given this observation, our best-case analysis is based on exploring all possible partitions of preempting tasks.

### 5.4 The phasing of delaying and extra preempting tasks

According to Theorem 2, the simultaneous activation of minimal preempting tasks has to take place at time \( t_1 = f_{i,k^{br}} \), where \( k^{br} \) is the job of \( r_i \) that assumes the best-case response time. Furthermore, the simultaneous activation of delaying and extra preempting tasks occurs at time \( t_2 = s_{i,k^{br}} + \Delta \). Therefore, the phasing \( \varphi_r \) for delaying and extra preempting tasks relative to the activation of minimal preempting tasks is given by

\[
\varphi_r = t_2 - t_1 = (s_{i,k^{br}} + \Delta) - f_{i,k^{br}} = -H_{i,k^{br}} + \Delta.
\]  

(5.1)

Note that the hold time of job \( k^{br} \) is needed to determine the phasing of delaying and extra preempting tasks. We therefore derive the following corollary.

**Corollary 1.** *In order to construct an instant with the property given by Theorem 2, it is necessary to first determine the hold time \( H_{i,k^{br}} \) of the job that assumes the best-case response time.*
Chapter 6

Exact Best-Case Response Time Analysis

In this chapter, we present and prove a novel exact best-case response time analysis for independent real-time periodic tasks with arbitrary deadlines scheduled using FPTS. Our analysis is based on the property for an optimal instant presented in Theorem 2. More precisely, we construct an instant with such a property for all possible partitions of preempting tasks into sets of extra and minimal preempting tasks. For each set of extra preempting tasks of a task \( \tau_i \), we first determine the hold time of a job \( k \) of \( \tau_i \) when experiencing such extra preemptions. Based on such a hold time, we can now construct an instant with the property for an optimal instant and subsequently derive its shortest response time. Afterwards, the best-case response time is simply determined by the minimum of all shortest response times that were explored.

6.1 Determining the hold times of a job

Based on Corollary 1, we have to determine the hold time of the job that assumes the best-case response time in order to construct an instant with the property described in Theorem 2. Note that this job is not necessarily the one with the shortest hold time (Fact 3). Hence, we propose to explore all possible hold times of a job of a task \( \tau_i \) where the best-case response time can be found. Furthermore, note that the hold time of a job is only influenced by the amount of preemptions that it experiences by its extra and minimal preempting tasks. Therefore, we derive all the possible hold times given a set of extra preempting tasks. To this end, we first introduce the notions of worst-case and best-case hold intervals.

**Definition 2.** The worst-case hold interval \( WH_i(y, \mathcal{E}) \) is defined as the worst-case hold time of an artificial task \( \tau'_i \) with a computation time of \( y \in \mathbb{R}^+ \) in the task-set \( \mathcal{E} \cup \{\tau'_i\} \) given that all jobs in the interval assume their best-case computation times.

The worst-case hold interval \( WH_i(y, \mathcal{E}) \) follows directly from the formula for worst-case hold time for FPTS given in [3]. The only difference is that best-case computation times are assumed for the worst-case hold interval. Hence, \( WH_i(y, \mathcal{E}) \) is given by the smallest \( x \in \mathbb{R}^+ \) satisfying

\[
x = y + \sum_{e : \pi_e \neq \emptyset, \tau_e \in \mathcal{E}} \left\lceil \frac{x}{T_e} \right\rceil BC_e.
\]  

**Definition 3.** The best-case hold interval \( BH_i(y, \mathcal{M}) \) is defined as the best-case hold time of an artificial task \( \tau'_i \) with a computation time of \( y \in \mathbb{R}^+ \) in the task-set \( \mathcal{M} \cup \{\tau'_i\} \) given that \( \mathcal{M} \) contains only preempting tasks.

Since Definition 3 assumes that \( \mathcal{M} \) only contains preempting tasks, we can use the same formula for the best-case hold time for FPPS given in [19] to calculate \( BH_i(y, \mathcal{M}) \). Hence, function \( BH_i(y, \mathcal{M}) \)

---

1 In [19], the term best-case hold time is referred as best-case execution time.
returns the largest positive solution of $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{m:n_m > \theta_i, w_{\ell_i} \in M} \left( \left\lceil \frac{x}{T_m} \right\rceil - 1 \right)^+ BC_m.$$  

(6.2)

Finally, we introduce the set of hold times of a task $\tau_i$ based on its extra and minimal preemptions.

**Definition 4.** The set of hold times $\mathcal{H}(E)$ is defined as the set of all possible hold times that a job $i_{1,k}$ of $\tau_i \in \mathcal{T}$ can have when experiencing extra preemptions by the tasks in $E \subseteq \mathcal{P}$, and minimal preemptions by the tasks in $M = \mathcal{P} \setminus E$. This is, when the tasks in $E$ are simultaneously activated at time $s_{i,k} + \Delta$, where $\Delta$ is a sufficiently small amount of time to experience such extra preemptions. Furthermore, delaying tasks of $\tau_i$ in $\mathcal{T}$ are ignored.

Clearly, a hold time $h \in \mathcal{H}(E)$ is given by the following equation:

$$h = \beta_m + \beta_e + BC_i,$$  

(6.3)

where $\beta_m, \beta_e \in \mathbb{R}^+ \cup \{0\}$ are the amount of time spent by preemptions of minimal preempting and extra preempting tasks in the hold time $h$ respectively. Furthermore, the values of $\beta_m$ and $\beta_e$ are given by the following set of recursive equations:

$$\beta_e = WH_i(\beta_m + BC_i, E) - \beta_m - BC_i,$$

$$\beta_m = BH_i(\beta_e + BC_i, M) - \beta_e - BC_i.$$  

(6.4)

Note that the first equation in (6.4) maximizes the influence of extra preempting tasks using the worst-case hold interval $WH_i(y, E)$. On the other hand, the second equation minimizes the influence of minimal preempting tasks using the best-case hold interval $BH_i(y, M)$.

It is worth noting that each possible solution for $\beta_e$ and $\beta_m$ using the set of equations in (6.4) gives rise to a new possible hold time in Equation (6.3). Therefore, the set of hold times $\mathcal{H}(E)$ is simply given by all such possible solutions.

As an example where more than one hold time can be found, consider the task-set $T_{6,1}$ with characteristics as described in Table 6.1. Similarly to the previous example, the best-case response time was found using brute force. Let task $\tau_1$ be an extra preempting task of $\tau_4$, and $\tau_2$ a minimal preempting task, i.e. $E = \{\tau_1\}$. For this task-set, the set of possible hold times for $\tau_4$ is $\mathcal{H}(E) = \{32, 54\}$. Figure 6.1 depicts two timelines in which these hold times are found for a job of task $\tau_4$. As can be seen, in both cases $\tau_1$ is activated just after the start time of the job of $\tau_4$; hence, $\tau_1$ is an extra preempting task. In addition, an activation of $\tau_2$ coincides with the completion of the job of $\tau_4$ also for both cases. Note that task $\tau_3$ is not considered because it cannot preempt $\tau_4$.

<table>
<thead>
<tr>
<th>Task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$w_{\ell_i}$</th>
<th>$BR_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>35</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>49</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>56</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{6,1} = 1960$ and $U_{6,1} \approx 0.98$.

In order to derive all possible solutions for the set of recursive equations in (6.4), we can first derive the smallest solutions $\beta^e_m$ and $\beta^e_p$ using an iterative procedure starting with a lower bound. Similarly, we can derive the largest solutions $\beta^p_m$ and $\beta^p_p$ starting with an upper bound, e.g. the amount of preemptions by preempting tasks in a job with worst-case hold time. Afterwards, if the smallest and largest solutions are equal then there is a single solution for the equation. Otherwise, we evaluate all intermediate values $\beta^e_m \leq \beta_e \leq \beta^p_e$ and $\beta^e_m \leq \beta_m \leq \beta^p_m$ in (6.4) to determine whether they are valid solutions. Note that it is possible to evaluate all intermediate values because $\beta_e$ and
\( \beta_m \) depend on the number of preemptions that extra and minimal preempting tasks generate in a job \( k \) respectively. Since these number of preemptions is integer and finite, we can always evaluate all possible values. For instance, in Figure 6.1.a, tasks \( \tau_4 \) experiences only one preemption by \( \tau_1 \); hence, \( \beta^e_4 = 10 \) and \( \beta^m_4 = 0 \). Furthermore, in Figure 6.1.b, \( \tau_4 \) experiences two preemptions by task \( \tau_1 \) and one by \( \tau_2 \), i.e. \( \beta^e_4 = 20 \) and \( \beta^m_4 = 12 \). Therefore, for this case, the only two possible intermediate solutions that we have to evaluate in (6.4) are the values for \( \beta_e \) and \( \beta_m \) when experiencing only two preemptions by \( \tau_1 \) (i.e. \( \beta_e = 20 \) and \( \beta_m = 0 \)), or when experiencing one preemption by both, \( \tau_1 \) and \( \tau_2 \) (i.e. \( \beta_e = 10 \) and \( \beta_m = 12 \)). Neither of these cases satisfies Equations (6.4); hence, the hold times depicted in Figure 6.1 are the only possible hold times for a job of \( \tau_4 \) when experiencing extra preemptions by task \( \tau_1 \).

### 6.2 Best-case interval for the execution of a task \( \tau_i \)

Similar to the best-case response time analysis with arbitrary deadlines for FPPS, a job of a task \( \tau_i \) scheduled using FPTS and experiencing its optimal instant may still experience interference by its previous jobs provoking a delay on its start time (see Figure 5.1). Therefore, we have to look to previous jobs of task \( \tau_i \) to determine its shortest response time. In order to do so, we have to determine intervals of minimal length with enough processing time to execute complete jobs of \( \tau_i \).

**Definition 5.** Given a job \( i_{i,k} \) of a task \( \tau_i \) with a hold time \( h \in H_i(E_i) \), where \( E_i \) is a set of extra preempting tasks, the generalized best-case interval \( GI_i(y,h,E_i) \) is defined as the length of the shortest interval \( [t_s,t_e] \) before the completion of \( i_{i,k} \), i.e. \( t_e = f_{i,k} \), in which exactly an amount of time \( y \in \mathbb{R}^+ \) is available for the execution of task \( \tau_i \).

Note that \( GI_i(y,h,E_i) \) is similar to the notion of best-case interval \( BI_i(y) \) given in [20]. The main difference is that, for \( GI_i(y,h,E_i) \), it is considered that the interval ends with a job \( i_{i,k} \) with hold time \( h \in H_i(E_i) \). Recall that it is necessary to specify the hold time of \( i_{i,k} \) in order to know the phasing at which extra preempting and delaying tasks must be activated and consequently derive its shortest response time. Therefore, we propose the following theorem.
Theorem 3. The generalized best-case interval \( G_{\ell}(y, h, \Xi) \) is given by the largest \( x \in \mathbb{R}^+ \) satisfying

\[
x = y + \sum_{m : r_m \in \mathcal{M}_i} \left( \frac{x}{T_m} \right)^+ BC_m + \sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{x - h}{T_e} \right]^+ BC_e + \beta_e(h) \tag{6.5}
\]

where \( \mathcal{M}_i = \mathcal{T}_1 \setminus \mathcal{E}_i \) is the set of minimal preempting tasks of \( \tau_i \), and \( \mathcal{D}_1 = \{ \tau_d \in \mathcal{T} \mid \pi_d > \pi_i \} \) is the set of delaying tasks. Furthermore, \( \beta_e(h) \) is the amount of preemptions by extra preempting tasks in the hold time \( h \), which can be calculated as \( \beta_e(h) = \sum_{e \tau, e \in \mathcal{D}_2} \left[ \frac{h}{T_e} \right] BC_e \).

Proof. The general best-case interval \( G_{\ell}(y, h, \Xi) \) consists of two parts: the amount of time \( y \in \mathbb{R}^+ \) that corresponds to the first term in the RHS of Equation (6.5), and the interference that higher priority tasks induce in the interval \( [t_e, t_e] \) of length \( G_{\ell}(y, h, \Xi) \). Since the set \( \mathcal{M}_i \) contains the minimal preempting tasks of \( \tau_i \), their influence on the interval \( [t_e, t_e] \) must be minimal, and it is obtained when there is a simultaneous activation of all minimal preempting tasks at time \( t_e \).

Therefore, the shortest amount of time reserved for minimal preempting tasks in the interval is given by

\[
\sum_{m : r_m \in \mathcal{M}_i} \left[ \frac{t_e - t_e}{T_m} \right]^+ BC_m = \sum_{m : r_m \in \mathcal{M}_i} \left[ \frac{G_{\ell}(y, h, \Xi)}{T_m} \right]^+ BC_m.
\]

This corresponds to the second term in Equation (6.5).

Note that Definition 5 assumes that a job \( t_{i,k} \) of task \( \tau_i \) with a hold time \( h \in \mathcal{N}_i(\Xi) \) executes at the end of \( [t_e, t_e] \), i.e. \( f_{i,k} = t_e \) and \( s_{i,k} = t_e - h \). The amount of time reserved for extra preempting tasks in the interval \( [s_{i,k}, t_e] \) of size \( h \) is given by the worst number of extra preempting jobs that execute in such an interval; hence, it is given by \( \beta_e(h) \). This corresponds to the last term in the RHS of Equation (6.5). On the other hand, delaying tasks cannot preempt \( \tau_i \); hence, there is no time reserved for delaying tasks in \( [s_{i,k}, t_e] \). It only remains to determine the amount of interference by extra preempting and delaying tasks in the interval \( [t_e, s_{i,k}] \).

From the property for an optimal instant given in Theorem 2, a simultaneous activation of delaying and extra preempting tasks occurs at time \( t = s_{i,k} + \Delta \). Note that delaying tasks activated before \( s_{i,k} \) have to be completed before \( s_{i,k} \) as well. If not, job \( t_{i,k} \) would not be able to start at that time. Hence, all jobs of delaying tasks activated in the interval \( [t_e, s_{i,k}] \) will give rise to interference in such an interval. The amount of processing time spent on the execution of delaying and extra preempting tasks in \( [t_e, s_{i,k}] \) is therefore given by

\[
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{(s_{i,k} + \Delta) - t_e}{T_e} \right]^+ BC_e = \\
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{t_e - h + \Delta - t_e}{T_e} \right]^+ BC_e = \\
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{G_{\ell}(y, h, \Xi) - h + \Delta}{T_e} \right]^+ BC_e
\]

Given Lemma 9 in the Appendix, when \( \Delta \) approaches zero the equation is simplified as follows.

\[
\lim_{\Delta \to 0} \sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{G_{\ell}(y, h, \Xi) - h + \Delta}{T_e} \right]^+ BC_e = \\
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \lim_{\Delta \to 0} \frac{G_{\ell}(y, h, \Xi) - h + \Delta}{T_e} \right]^+ BC_e = \\
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ \frac{G_{\ell}(y, h, \Xi) - h + \Delta}{T_e} \right]^+ BC_e = \\
\sum_{e \tau, e \in \mathcal{D}_1 \cup \mathcal{D}_2} \left[ G_{\ell}(y, h, \Xi) - h \right]^+ BC_e.
\]

This corresponds to the third term in Equation (6.5). \(\square\)
GI(y, h, Ei) can be found by an iterative procedure starting with an upper bound, e.g. the worst-case response time of τi for a computation time y. It is worth noting that the generalized best-case interval GI(y, h, Ei) specializes to the best-case interval BI(y) for FPPS when there are no delaying and extra preempting tasks.

Figure 6.2 shows an example of a generalized best-case interval for task τ4 ∈ T6.1 when considering the time that is available for the execution of two jobs, i.e. y = 2·BC4. Furthermore, τ1 is chosen as the extra preempting task and the hold time of the last job is h = 32.

Figure 6.2: Timeline for T6.1 depicting the generalized best-case interval GI4(2·BC4, h, E4) = 106, where h = 32 and E4 = {τ1}.

6.3 Shortest response time of a job

Given the notion of generalized best-case interval, we first introduce a lower bound for the shortest response time of a job experiencing a hold time h ∈ H(Ei). Afterwards, we present the exact shortest response time for such a job scheduled using FPTS.

Lemma 4. Let all tasks of a set T be strictly periodic and let wℓi exist. Furthermore, let job i−1 of τi have a hold time equal to h ∈ H(Ei). For every ℓ ∈ N a lower bound for the shortest response time SRi of i−1 is given by

\[ SR_i(h, E_i) \geq \max_{1 \leq k \leq \ell} (GI_i(k \cdot BC_i, h, E_i) - (k-1)T_i) \] (6.6)

Proof. The proof is similar to the proof of Lemma 4 in [20]. For completeness, the proof can be found in Appendix B.1.

We now introduce a Lemma for the number of jobs to explore in order to find the highest lower bound for SRi(h, Ei) using (6.6).

Lemma 5. To determine the highest lower bound using (6.6), it is sufficient to consider at most wℓi jobs.

Proof. The proof follows directly from the proof of Lemma 5 in [20]. For completeness, the proof can be found in Appendix B.2.

Theorem 4. Let job i−1 of τi have a hold time equal to h ∈ H(Ei) and let wℓi exist. The shortest response time SRi(h, Ei) of job i−1 scheduled under FPTS is given by

\[ SR_i(h, E_i) = \max_{1 \leq k \leq \ell} (GI_i(k \cdot BC_i, h, E_i) - (k-1)T_i). \] (6.7)

Proof. Similar to Theorem 1 in [20]. For completeness, the proof can be found in Appendix B.3.
6.4 Best-case response times

Given that Theorem 4 yields the shortest response time of a job of \( \tau_i \) with a hold time equal to \( h \in \mathcal{H}_i(\mathcal{E}_i) \) for a set of extra preemting tasks \( \mathcal{E}_i \), we now propose to explore all possible hold times in order to find the best-case response time. Note that it is necessary to explore, in the worst case, all possible hold times because, unlike FPPS, the best-case response time is not necessarily found in the job experiencing the shortest hold time. Therefore, the best-case response time could potentially be found for any hold time. Furthermore, it is not clear how to partition the set of preempting tasks \( \mathcal{F}_i \) into sets of extra and minimal preempting tasks. Hence, the best-case response time is found by considering all hold times for all partitions of preempting tasks.

**Corollary 2.** Let all tasks of a set \( \mathcal{T} \) be strictly periodic and let \( w^\ell_i \) exist. Furthermore, let \( \hat{\mathcal{F}}_i = \{ r_p \in \mathcal{T} \mid \pi_p > \theta_i, \exists r_d : \theta_i \geq \pi_d > \pi_i \} \) be the set of preempting tasks of a task \( \tau_i \in \mathcal{T} \) when at least one delaying task exists. The best-case response time of task \( \tau_i \) is given by

\[
BR_i = \min_{E \in 2^{\hat{\mathcal{F}}_i}} \min_{h \in \mathcal{H}_i(E)} (SR_i(h, E)). \tag{6.8}
\]

Note that Corollary 2 specializes to the best-case response time analysis for FPPS when there are no delaying tasks. For this special case, the set of extra preempting tasks is empty and, therefore, all higher priority tasks are considered as minimal preempting tasks. Finally, since the best-case analysis given by Corollary 2 explores all possible partitions of preempting tasks, the analysis translates to an algorithm with exponential time complexity \( \mathcal{O}(2^n) \) for \( n \) number of tasks.

6.5 An example

We revisit the example given in Section 6.1 to determine the best-case response time of task \( \tau_4 \in \mathcal{T}_6 \) using our novel analysis. To this end, we first determine the set of hold times \( \mathcal{H}_4(\mathcal{E}_4) \) that a job of \( \tau_4 \) can assume using the analysis in Section 6.1. Table 6.2 shows such hold times for all possible partitions of preempting tasks.

<table>
<thead>
<tr>
<th>( \mathcal{E}_4 )</th>
<th>{ }</th>
<th>{ \tau_1 }</th>
<th>{ \tau_2 }</th>
<th>{ \tau_1, \tau_2 }</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{H}_4(\mathcal{E}_4) )</td>
<td>(22)</td>
<td>(32,54)</td>
<td>(44)</td>
<td>(66)</td>
</tr>
</tbody>
</table>

Based on a hold time of a job of \( \tau_4 \), we can derive its shortest response time \( SR_4(h, \mathcal{E}_4) \) using Theorem 4. From Table 6.1, note that the worst-case number of jobs of \( \tau_4 \) in a level-4 active period is \( w^\ell_4 = 4 \). The intermediate results for \( SR_4(h, \mathcal{E}_4) \) are depicted in Table 6.3. Furthermore, the maximum of such intermediate results is the shortest response time for the given configuration. The best-case response time is then simply the minimum of such shortest response times according to Corollary 2, i.e. \( BR_4 = \min\{40,36,54,44,66\} = 36 \).

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SR_4(22,{ } ) )</td>
<td>22</td>
<td>36</td>
<td>18</td>
<td>40</td>
</tr>
<tr>
<td>( SR_4(32,{ \tau_1 } ) )</td>
<td>32</td>
<td>36</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>( SR_4(54,{ \tau_1 } ) )</td>
<td>54</td>
<td>6</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>( SR_4(44,{ \tau_2 } ) )</td>
<td>44</td>
<td>36</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>( SR_4(66,{ \tau_1, \tau_2 } ) )</td>
<td>66</td>
<td>18</td>
<td>40</td>
<td>30</td>
</tr>
</tbody>
</table>

The value selected as the shortest response time \( SR_4(h, \mathcal{E}_4) \) is highlighted in bold. That is the maximum of all its intermediate results.
Figure 6.2 shows a timeline where the best-case response time $BR_4 = 36$ of $τ_4$ is assumed by the job ending at time $t = 0$. As can be seen, this best-case response time is found for a job with hold time $h = 32$ and when experiencing extra preemptions by $τ_1$.

In Appendix D, we present additional examples applying the novel best-case response time analysis for FPTS. More precisely, the example in D.1 shows that the best-case response time can also be found in a job assuming its worst-case hold time, i.e. when all its preempting tasks are extra preempting tasks. Furthermore, the example in D.2 shows that the largest hold time in a set of hold times $H_i(E_i)$ can also lead to the best-case response time. These examples therefore illustrate the need of exploring multiple partitions of preempting tasks.

6.6 Some optimizations to the analysis

In this section, we present some optimizations that can be done to speed-up the calculation of the best-case response time of a task.

6.6.1 Number of jobs to explore before an optimal instant

Lemma 5 states that it is sufficient to consider at most $wℓ_i$ jobs in order to determine the shortest response time of a job. We propose to improve this bound on the number of jobs to explore as follows.

**Lemma 6.** Let $wℓ'_i$ be the worst-case number of jobs of task $τ_i$ in a level-$i$ active period when excluding lower priority tasks of $τ_i$. To determine the highest lower bound using (6.6), it is sufficient to consider at most $wℓ'_i$ jobs.

**Proof.** We first observe from Lemma 5 that it is sufficient to consider $wℓ_i$ jobs to determine the highest lower bound using (6.6). Now, recall from Theorem 1 that lower priority tasks do not affect the best-case response time of a task $τ_i$. Therefore, the highest lower bound using (6.6) for $τ_i ∈ T$ should be equal to the highest lower bound for $τ_i$ when excluding lower priority tasks. Since it is sufficient to consider at most $wℓ'_i$ jobs for the latter case, we conclude that the same number jobs are sufficient for task-set $T$. \(\square\)

6.6.2 A constraint on the hold time of a task

As mentioned in Section 6.1, a set of extra preempting (and minimal preempting) tasks can give rise to multiple possible hold times. Furthermore, we aim to determine the shortest response times of a job when assuming such hold times. In this section, we show that a set of extra preempting (and minimal preempting) tasks can give rise to hold times that are unfeasible.

Consider the task-set $T_{6.4}$ given in Table 6.4. Note that $τ_1$ is a preemptive task of $τ_3$, whereas $τ_2$ is a delaying task. Suppose that we want to investigate the response time of a job $ι_{3,k}$ of $τ_3$ when it does not experience any extra preemption, i.e. when $E_3 = \{\}$ and $M_3 = \{τ_1\}$. From Definition 4 and after solving equation (6.3), we found that the set of possible hold times for $ι_{3,k}$ is given by $H_i(E_3) = \{6\}$. However, we will now show that it is impossible to construct a schedule for task-set $T_{6.4}$ in which a job $ι_{3,k}$ assumes a hold time $H_{3,k} = 6$.

**Table 6.4: Characteristics of task-set $T_{6.4}$.**

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$π_i$</th>
<th>$θ_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$τ_1$</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$τ_2$</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$τ_3$</td>
<td>40</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The hyperperiod is $P^{T_{6.4}} = 20$ and $U^{T_{6.4}} = 0.8$.

Figure 6.3.a shows a timeline for $T_{6.4} \setminus \{τ_2\}$ in which $τ_3$ has a hold time equal to $H_{3,k} = 6$ for the job $ι_{3,k}$ starting at time $t = -6$. Note that $τ_1$ is scheduled as a minimal preemptive task because...
one of its activations coincides with the completion of $τ_{3,k}$ at time $t = 0$. In addition, note that $τ_{3,k}$
can only start at time $t = -6$ because there is no pending load at that time, i.e. $P_3(-6) = 0$. Figure
6.3.b shows the timeline for $T_{6,4}$ when introducing delaying task $τ_2$ into the schedule. For this case,
no matter how $τ_2$ is scheduled, there is always pending load at time $t = -6$ and, consequently, the
hold time $H_{3,k} = 6$ cannot be assumed. Therefore, we conclude that such a hold time in $H_6(E_3)$ with
$E_3 = \{\}$ is not a valid hold time for task $τ_3$.

From the previous example, note that the reason for which the hold time $H_{3,k} = 6$ cannot be
assumed is that there is not enough space available in any time interval of that length for the
execution of one job of $τ_3$. Therefore, in order to determine whether a hold time $h$ in the set $H_i(E_i)$
is feasible for a task $τ_i$, we only have to verify whether there exists a time interval of length equal
to $h$ in which a complete job of $τ_i$ can execute. This can be expressed using the notion of generalized
best-case interval for a single job of $τ_i$. In fact, if we investigate the best-case interval for a single
job of $τ_3$ with hold-time $h = 6$ when experiencing minimal preemptions by task $τ_1$, i.e. when $E_3$ is
empty, we obtain that an interval of length $GI(BC_3, 6, \{\}) = 11$ is needed for the execution of such a
job with hold time $h = 6$. This clearly is a contradiction because such a job should be able to execute
completely during its hold time $h = 6$. Therefore, we propose the following constraint on the hold
time of a task.

**Corollary 3.** Given a task-set $T$ and a set of extra preempting tasks $E_i$ for a task $τ_i \in T$, it is possible
to schedule a job $i_{1,k}$ of $τ_i$ with a hold time $h \in H_i(E_i)$ if and only if $h = GI_i(BC_i, h, E_i)$ holds.

We can use Corollary 3 as a quick test to determine whether a hold time of a job is feasible. If
not, there is no need to determine its shortest response time.

---

**Figure 6.3:** Two timelines for $T_{6,4}$. In a) $τ_2$ is ignored from the task-set and $τ_1$ is a minimal preemtive task of $τ_3$. In b) task $τ_2$ is introduced, affecting the schedule for $τ_3$. 

---

**Best-case response time of real-time tasks under FPTS**
Chapter 7

A Lower-Bound for the Best-Case Response Time

In the previous chapter, we introduced an exact best-case response time analysis for real-time tasks scheduled using FPTS. This analysis, explores all possible partitions of preempting tasks into the sets of extra preempting and minimal preempting tasks in order to derive the best-case response time. Therefore, it translates into an algorithm of exponential time complexity. In this chapter, we propose a novel lower bound for the best-case response time of a task scheduled using FPTS with the aim of providing a tractable schedulability test for a task. To this end, we eliminate the need of partitioning the set of preempting tasks. The novel lower bound is based on an instance of the property for an optimal instant that we termed minimal preempting instant and, in addition, on a new concept that we termed artificial hold time.

We start this chapter by introducing the minimal preempting instant and the notion of artificial hold time. Next, we formalize the lower bound for the best-case response and present an algorithm to calculate it. We finalize this chapter by applying the novel algorithm to an example.

7.1 A minimal preempting instant

First, recall that there are two types of tasks that may affect the best-case response time of a task $\tau_i$. Those are the delaying tasks in $D_i = \{ \tau_d \in \tau | \theta_i \geq \pi_d > \pi_i \}$ and the preempting tasks in $P_i = \{ \tau_p \in \tau | \pi_p > \theta_i \}$. Furthermore, recall that lower priority tasks of $\tau_i$ have no impact on its best-case response time, and they are therefore ignored. Based on the tasks that may influence the best-case response time, we already presented a property for an optimal instant. In particular, such a property for an optimal instant holds when some preempting tasks are activated a sufficiently small amount of time after the start of the job experiencing the optimal instant, i.e. extra preempting tasks. Therefore, the exact analysis is based on exploring all possible partitions of preempting tasks into sets of extra and minimal preempting tasks.

In order to construct a lower bound for the best-case response time of a task $\tau_i$, we explore the case when there are no extra preempting tasks, i.e. when the completion of a job of $\tau_i$ coincides with the simultaneous activation of all its preempting tasks. We call such an instance a minimal preempting instant.

**Definition 6.** A minimal preempting instant of a task $\tau_i$ occurs when the simultaneous activation of all its preempting tasks coincides with the completion of a job $\iota_{i,k}$ of $\tau_i$.

As an example, consider the task-set $\mathcal{T}_{7,1}$ with characteristics as described in Table 7.1. The values for the best-case and worst-case response times in the table were found using the analyses given in previous sections. Figure 7.1 shows a timeline for task-set $\mathcal{T}_{7,1}$ depicting a minimal preempting instant for the last job $\iota_{4,-1}$ of task $\tau_4$. As can be seen, all preempting tasks are simultaneously ac-
tivated at time $f_{4,-1} = 0$. Furthermore, the last job of $\tau_4$ assumes its best-case hold time $H_{4,-1} = 22$ and a shortest response time of $R_{4,-1} = SR_i(22, \{ \}) = 36$.

Table 7.1: Characteristics of task-set $\mathcal{T}_{7.1}$.

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$q_i$</th>
<th>$w_i$</th>
<th>$\ell_i$</th>
<th>WR$_i$</th>
<th>BR$_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>35</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>35</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>50</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>62</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>66</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{\mathcal{T}_{7.1}} = 350$ and $U_{\mathcal{T}_{7.1}} = 1$.

Figure 7.1: A timeline for $\mathcal{T}_{7.1}$ depicting the minimal preempting instant for the last job $i_{4,-1}$ of task $\tau_4$. Preempting tasks $\tau_1$ and $\tau_2$ are simultaneously activated at time $f_{4,-1} = 0$. Furthermore, delaying task $\tau_3$ is activated a sufficiently small amount of time after $s_{4,-1}$.

Note that the shortest response time of a job experiencing a minimal preempting instant is an upper bound for the best-case response time, as illustrated in Figure 7.1. Furthermore, a job experiencing a minimal preempting instant assumes the shortest possible hold time. Hence, this hold time is a lower bound for the best-case response time. Therefore, we introduce the following corollary that bounds the best-case response time of a task.

**Corollary 4.** Let $h_{i_{mp}}$ be the hold time of a job $k_{mp}$ of $\tau_i$ that experiences a minimal preempting instant. Furthermore, let $r_{i_{mp}}$ be the shortest response time of such a job, i.e. $r_{i_{mp}} = SR_i(h_{i_{mp}}, \{ \})$. The best-case response time of $\tau_i$ is bounded from above by $r_{i_{mp}}$ and from below by $h_{i_{mp}}$, i.e.

$$h_{i_{mp}} \leq BR_i \leq r_{i_{mp}}. \quad (7.1)$$

It is worth noting that a lower bound $BR_{i_{lb}}$ for the best-case response time of a task $\tau_i$ is also within the bounds described by Corollary 4, i.e. $h_{i_{mp}} \leq BR_{i_{lb}} \leq BR_i \leq r_{i_{mp}}$. In the remaining of this chapter, we formalize a novel lower bound for the best-case response time. To this end, we first introduce the notion of artificial hold time.

### 7.2 Artificial hold time of a job

Recall from Corollary 1 that, given a job $k_{br}$ of task $\tau_i$ that assumes the best-case response time, the activation of delaying tasks relative to the simultaneous activations of minimal preempting tasks depends on the hold time $H_{i,k_{br}}$. Clearly, this hold time is also within the bounds given by Corollary 4. In order to simulate the activation of delaying tasks when the hold time of a job of $\tau_i$ is $H_{i,k_{br}}$ without the need of introducing extra preempting tasks, we introduce the notion of artificial hold time.

**Definition 7.** An artificial hold time $h_{i_{ar}}$ for a job $k_{mp}$ of task $\tau_i$, experiencing a minimal preempting instant is an artifact that forces a simultaneous activation of delaying tasks at time $t = f_{i,k_{mp}} - h_{i_{ar}}$. 

Best-case response time of real-time tasks under FPTS
**Furthermore, all the executions of delaying jobs activated at or after time** \( t = f_{i,k_{mp}} - h_{ar} \) **are postponed after** \( f_{i,k_{mp}} \) **and the activation time of** \( i_{k_{mp}} \) **is constraint to** \( a_{i,k_{mp}} \leq f_{i,k_{mp}} - h_{ar} \).**

Based on the previous definition, the only purpose of this artificial hold time of a job \( i_{k_{mp}} \) **is to force the activation of delaying tasks at certain time before the minimal preempting instant without modifying the schedule for the job** \( i_{k_{mp}} \). Figure 7.2 shows an example of this situation. Because we do not allow the job of the delaying task to execute during the artificial hold time \( H_{ar} \), the activation of the delaying task \( \tau_3 \) can be moved to an earlier moment in time without affecting the schedule of \( \tau_{4} \). For this case, note that the response time of \( \tau_{4} \) remains the same as in Figure 7.1.

Figure 7.2: Timeline for \( T_{7.1} \) where an artificial hold time \( H_{ar} = 24 \) is assumed by the last job \( \tau_{4} \) activated at time \( t = -36 \).

From Figure 7.2, note that the job \( \tau_{4} \) of \( \tau_{4} \) activated at time \( t = -176 \) is the job that determines the shortest response time of the job experiencing the minimal preempting instant. We call this job “tight” because it can immediately start upon activation and, in addition, restricts the response time of the job \( \tau_{4} \) experiencing the minimal preempting instant. In order to reduce the response time of \( \tau_{4} \), we can increase its artificial hold time till job \( \tau_{4} \) does no longer determine its response time. Figure 7.3 shows that this is achieved when the artificial hold time for \( \tau_{4} \) is \( H_{ar} = 26 \), allowing activations of the delaying task \( \tau_3 \) at times \( t = -26 \) and \( t = -176 \). Note that the activation of \( \tau_3 \) at time \( t = -176 \) “releases” job \( \tau_{4} \) from being tight, i.e. this job does not longer delay the start time of \( \tau_{4} \). Therefore, the response time of job \( \tau_{4} \) can be reduced. Furthermore, note that the activation time of \( \tau_{4} \) is now determined by such artificial hold time; hence, the new artificial response time of \( \tau_{4} \) is \( R_{ar} = 26 \).

Figure 7.3: Timeline for \( T_{7.1} \) where a sufficiently large artificial hold time is assumed by the last job of \( \tau_{4} \) to reduce its response time till the start of the artificial hold time.

Based on the notion of artificial hold time, we propose to investigate the shortest response time of a job experiencing a minimal preempting instant when assuming artificial hold times \( h_{ar} \) in the range given by Corollary 4, i.e. \( h_{mp} \leq h_{ar} \leq r_{mp} \). The minimum of such shortest response times is the lower bound for the best-case response time that we propose. For the previous example, the best case response time lower bound for task \( \tau_4 \) is \( BR_{lb} = 26 \). In the following sections, we prove that this is a proper lower bound for the best-case response time of a task scheduled using FPTS.

We finalize this section summarizing the properties of an artificial hold time. In general, a job \( h_{mp} \) of task \( \tau_i \) assuming an artificial hold time \( h_{ar} \in \mathbb{R}^+ \) and experiencing a minimal preempting
Theorem 5. The best-case artificial interval assumed by $\iota$ ends with a job for the exact analysis. The main difference is that, for $\text{BI}_{\iota}$ of the shortest interval $\iota$, Let $\text{Definition 8.}$

Property 2. The artificial hold time is not considered in the schedule of $\tau_i$. It just serves as a means to allow delaying tasks to be activated earlier without affecting the schedule of $\iota_{\text{BC}}$. As an example see Figure 7.3.

Property 3. The activation time of the job $\iota_{\text{BC}}$ is considered to be not later than the start of the interval of the artificial hold time. Hence, it holds that $R_{\iota_{\text{BC}}} \geq h^{\text{ar}}$ for the artificial response time of $\iota_{\text{BC}}$. As an example see Figure 7.3.

Property 4. The job $\iota_{\text{BC}}$ under consideration can execute its best-case computation time $BC_i$ in the interval of the artificial hold time, i.e. the artificial hold time is large enough to accommodate the execution of the job $\iota_{\text{BC}}$.

7.3 Best-case interval for the execution of a task experiencing an artificial hold time

Similar to the exact best-case analysis, a job assuming an artificial hold time can still experience interference by its previous jobs. This is the case in Figure 7.2 where the last job of $\tau_4$ cannot start upon activation. Therefore, we have to look to previous jobs of task $\tau_4$ to determine its shortest response time. In order to do so, we have to determine intervals of minimal length with enough processing time to execute complete jobs of $\tau_i$. We subsequently prove that this best-case interval is at most equal to the corresponding generalized best-case interval.

Definition 8. Let $\iota_{\text{BC}}$ be a job of a task $\tau_i$ that experiences a minimal preempting instant. Furthermore, let $\iota_{\text{BC}}$ assume an artificial hold time $h$, i.e. all jobs of delaying tasks activated at or after $f_i,h - h$ are postponed after $f_i,h$. The best-case artificial interval $BI_{\text{ar}}(y,h)$ is defined as the length of the shortest interval $[t_s,t_e]$ before the completion of $\iota_{\text{BC}}$, i.e. $t_e = f_i,h$, in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of the job $\iota_{\text{BC}}$.

Note that $BI_{\text{ar}}(y,h)$ is similar to the notion of generalized best-case interval $GI_{\text{BC}}(y,h,\overline{x})$ needed for the exact analysis. The main difference is that, for $BI_{\text{ar}}(y,h)$, it is considered that the interval ends with a job $\iota_{\text{BC}}$ with an artificial hold time $h$ which is not necessarily a hold time that can be assumed by $\iota_{\text{BC}}$. We now propose the following theorem.

Theorem 5. The best-case artificial interval $BI_{\text{ar}}(y,h)$ is given by the largest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{p: \tau_p \in \mathcal{T}_i} \left( \left\lfloor \frac{x}{T_p} \right\rfloor - 1 \right)^+ BC_p + \sum_{d: \tau_d \in \mathcal{D}_i} \left( \left\lfloor \frac{x - h}{T_d} \right\rfloor - 1 \right)^+ BC_d,$$

(7.2)

where $\mathcal{T}_i$ and $\mathcal{D}_i$ are the set of preempting tasks and delaying tasks respectively.

Proof. The general best-case interval $BI_{\text{ar}}(y,h)$ consists of three parts: the amount of time $y \in \mathbb{R}^+$ that corresponds to the first term in the RHS of Equation (7.2), and the interference of preempting and delaying tasks in the interval $[t_s,t_e]$ of length $BI_{\text{ar}}(y,h)$. Since we assumed in Definition 8 that the last job in the interval experiences a minimal preempting instant, the influence of preempting tasks in the interval $[t_s,t_e]$ must be minimal. Furthermore, it is obtained when there is a simultaneous activation of all preempting tasks at time $t_s$. Therefore, the shortest amount of time reserved for preempting tasks in the interval is given by

$$\sum_{p: \tau_p \in \mathcal{T}_i} \left( \left\lfloor \frac{t_e - t_s}{T_p} \right\rfloor - 1 \right)^+ BC_p = \sum_{p: \tau_p \in \mathcal{T}_i} \left( \left\lfloor \frac{BI_{\text{ar}}(y,h)}{T_p} \right\rfloor - 1 \right)^+ BC_p.$$
where $y$ amount of time from the proof of (7.2) recall that the best-case interval notation considering the interference in the interval $g$ and delaying tasks as follows.

In order to prove the lemma, we first introduce a short hand notation for the equations of Proof.

**Lemma 7.** Given a job $i,j,k$ that assumes a hold time $h \in \mathcal{H}(E_i)$ and a set of extra preempting tasks $E_i$ that leads to such a hold time. The generalized best-case interval $GI_i(y,h,E_i)$ for a given $y \in \mathbb{R}^+$ is bounded from below by the best-case interval $BI_i^P(y,h)$, i.e. it holds that

$$GI_i(y,h,E_i) \geq BI_i^P(y,h).$$  

(7.3)

**Proof.** In order to prove the lemma, we first introduce a short hand notation for the equations of the best-case intervals for the sake of clarity. Let $g = GI_i(y,h,E_i)$ and $b = BI_i^P(y,h)$. Furthermore, from the proof of (7.2) recall that the best-case interval $BI_i^P(y,h)$ consists of three parts. That is the amount of time $y$, a minimal interference $I_p^m(b)$ of preempting tasks in the best-case interval, and the interference $I_d(b,h)$ of delaying tasks in the best-case interval. The interference by preempting and delaying tasks is given by

$$I_p^m(b) = \sum_{p:p \in P_i} \left( \left\lfloor \frac{b}{T_p} \right\rfloor - 1 \right)^+ BC_p.$$

$$I_d(b,h) = \sum_{d:d \in D_i} \left( \left\lfloor \frac{b-h}{T_d} \right\rfloor - 1 \right)^+ BC_d.$$

Using this shorthand notation and Equation (7.2), the best-case interval $b$ is given by

$$b = y + I_p^m(b) + I_d(b,h).$$  

(7.4)

Similarly, the generalized best-case interval in Equation (6.5) can be expressed in a short hand notation considering the interference in the interval $g$ by minimal preempting, extra preempting and delaying tasks as follows.

$$g = y + I_m^m(g) + I_e(g,h) + I_d'(g,h),$$

(7.5)

where

$$I_m^m(g) = \sum_{m:m \in M_i} \left( \left\lfloor \frac{g}{T_m} \right\rfloor - 1 \right)^+ BC_m$$

$$I_e(g,h) = \sum_{e:e \in E_i} \left[ \left\lfloor \frac{g-h}{T_e} \right\rfloor - 1 \right]^+ BC_e + \beta_e(h)$$

$$I_d'(g,h) = \sum_{d:d \in D_i} \left[ \left\lfloor \frac{g-h}{T_d} \right\rfloor - 1 \right]^+ BC_d.$$
We now prove that \( g \geq b \) holds. The proof is based on a contradiction argument. Assume that \( g < b \) holds. Recall that the generalized best-case interval can be found by solving Equation (6.5) using an iterative procedure starting with an upper bound. Furthermore, in every iteration of such a procedure, the result decreases till we find the largest positive solution that satisfies the equation. In this case, this solution is \( g \). Since the result of (6.5) decreases in every iteration, and since we assumed that \( b \) is an upper bound for \( g \), the following holds

\[
y + I_m^\min(b) + I_e(b, h) + I_d'(b, h) < b
\]

Using (7.4), we can replace the RHS of the inequality as follows.

\[
y + I_m^\min(b) + I_e(b, h) + I_d'(b, h) < y + I_m^\min(b) + I_d(b, h),
\]

which can be reduced to

\[
I_m^\min(b) + I_e(b, h) + I_d'(b, h) < I_m^\min(b) + I_d(b, h).
\] (7.6)

Note that \( I_p^\min(b) \) considers that all preemptions in the interval \( b \) of a job of task \( \tau \) by extra preempting tasks, it cannot be larger than any other interference by extra preemptions in the same interval. Hence, we arrive at a contradiction. We conclude that our initial assumption was wrong and, therefore, the lemma is proven.

\[
I_e(b, h) < I_e^\min(b)
\] (7.8)

Hence, we replace \( I_p^\min(b) \) in the RHS of (7.6) as follows

\[
I_m^\min(b) + I_e(b, h) + I_d'(b, h) < I_m^\min(b) + I_e^\min(b) + I_d(b, h),
\]

which can be reduced to

\[
I_e(b, h) + I_d'(b, h) < I_e^\min(b) + I_d(b, h).
\] (7.7)

We now derive a relation between \( I_d(b, h) \) and \( I_d'(b, h) \) as follows.

\[
I_d(b, h) = \sum_{d: \tau_d \in D_h} \left( \left\lceil \frac{b - h}{T_d} \right\rceil - 1 \right) + BC_d
\]

\[
\leq \sum_{d: \tau_d \in D_h} \left\lceil \frac{b - h}{T_d} \right\rceil + BC_d
\]

\[
= I_d'(b, h)
\]

Hence, since it holds that \( I_d(b, h) \leq I_d'(b, h) \), (7.7) is simplified as follows.

\[
I_e(b, h) < I_e^\min(b)
\]

Since \( I_e^\min(b) \) is the minimal interference in the best-case interval \( b \) by extra preempting tasks, it cannot be larger than any other interference by extra preemptions in the same interval. Hence, we arrive at a contradiction. We conclude that our initial assumption was wrong and, therefore, the lemma is proven.

### 7.4 Shortest response time of a job experiencing an artificial hold time

Given the notion of best-case interval \( B_{I_e^\min}(y, h) \), we can now determine the shortest response time of a job of task \( \tau_1 \) that assumes an artificial hold time. We subsequently prove that this artificial shortest response time is at most equal to the corresponding shortest response time presented in the previous chapter.
**Theorem 6.** Let all tasks of a set $\mathcal{T}$ be strictly periodic and let $w\ell_i$ exist. Furthermore, let job $i,j$ of $\mathcal{T}$ experience minimal preemptions and assume an artificial hold time $h$. The shortest artificial response time $SR_i^{\text{ar}}$ of $i,j$ is given by

$$SR_i^{\text{ar}}(h) = \max\{h, \max_{1 \leq k \leq w\ell_i} (BI_i^{\text{ar}}(k \cdot BC_i, h) - (k - 1)T_i)\}$$  \hfill (7.9)

**Proof.** The proof follows directly from the proof of Theorem 4. Furthermore, the outer max guarantees that the Property 3 of an artificial hold time presented in Section 7.2 is met, i.e. the response time of a job that assumes an artificial hold time $h^{\text{ar}}$ is at least equal to its artificial hold time.

**Figure 7.2** and **Figure 7.3** show examples of a job of $r_4 \in \mathcal{T}_{i,1}$ experiencing shortest artificial response times when assuming artificial hold times. We now introduce a lemma related with the notion of shortest response time.

**Lemma 8.** Given a job $i,j$ that assumes a hold time $h \in \mathcal{H}(i,\varepsilon)$ and a set of extra preempting tasks $\varepsilon_i$ that leads to such a hold time. The shortest response time of such a job is bounded from below by the shortest artificial response time of a job experiencing a minimal preempting instant and that assumes an artificial hold time $h$, i.e.

$$SR_i(h,\varepsilon_i) \geq SR_i^{\text{ar}}(h).$$  \hfill (7.10)

**Proof.** We prove the lemma by contradiction. Assume that $SR_i(h,\varepsilon_i) < SR_i^{\text{ar}}(h)$ holds. Using Theorem 6, we can rewrite our assumption as follows

$$SR_i(h,\varepsilon_i) < \max_{1 \leq k \leq w\ell_i} (BI_i^{\text{ar}}(k \cdot BC_i, h) - (k - 1)T_i).$$  

Furthermore, by definition, $SR_i(h,\varepsilon_i)$ is the shortest response time of a job that assumes a hold time $h \in \mathcal{H}(i,\varepsilon_i)$. It therefore holds that $SR_i(h,\varepsilon_i) \geq h$. Based on this and using Theorem 4, we can simplify the inequality to

$$\max_{1 \leq k \leq w\ell_i} (GI_i(k \cdot BC_i, h, \varepsilon_i) - (k - 1)T_i) < \max_{1 \leq k \leq w\ell_i} (BI_i^{\text{ar}}(k \cdot BC_i, h) - (k - 1)T_i).$$  \hfill (7.11)

Assume that we find a $k^{\text{max}}$ with $1 \leq k^{\text{max}} \leq w\ell_i$ that leads to the maximum value in the RHS of (7.11). Hence, we can derive the following.

$$GI_i(k^{\text{max}} \cdot BC_i, h, \varepsilon_i) - (k^{\text{max}} - 1)T_i < BI_i^{\text{ar}}(k^{\text{max}} \cdot BC_i, h) - (k^{\text{max}} - 1)T_i,$$  \hfill (7.12)

which simplifies to

$$GI_i(k^{\text{max}} \cdot BC_i, h, \varepsilon_i) < BI_i^{\text{ar}}(k^{\text{max}} \cdot BC_i, h).$$  \hfill (7.13)

From Lemma 7, we know that the generalized best-case interval is always larger than or equal to the best-case interval of a job with an artificial hold time. Therefore, we arrive at a contradiction and the lemma is therefore proven.

**7.5 Best-case response time lower bound**

Given the notion of shortest artificial response time $SR_i^{\text{ar}}(h)$ of a job that assumes an artificial hold time, we now introduce our novel lower bound for the best-case response time. This is found in the minimum shortest artificial response time when considering artificial hold times in the range $[h^{\text{mp}}_i, r^{\text{mp}}_i]$, where $h^{\text{mp}}_i$ is the hold time of a job of $r_i$ experiencing minimal preemptions and $r^{\text{mp}}_i$ is the shortest response time of such a job, i.e. $r^{\text{mp}}_i = SR_i(h^{\text{mp}}_i,\{}\})$. Since $h^{\text{mp}}_i$ is the hold time of a job of $r_i$ experiencing minimal preemption, it can be calculated using the best-case hold interval $BH_i(y, \varepsilon_i)$ given by Equation (6.2) for an interval $y = BC_i$. Recall that $BH_i(y, \varepsilon_i)$ gives the best-case hold time of a job of $r_i$ with computation time $y$.

We now formalize the lower bound for the best-case response time of a task scheduled using FPTS as follows.
Theorem 7. Let all tasks of a task-set $T$ be strictly periodic and let $wl_i$ exist. A lower bound $BR_i^{lb}$ for the best-case response time of a task $\tau_i$ scheduled using FPTS is given by

$$BR_i^{lb} = \min_{h \in [h_{i}^{mp}, r_i^{mp}]} (SR_i^{ar}(h)), \quad (7.14)$$

where $h_{i}^{mp}$ and $r_i^{mp}$ are the hold time and the shortest response time respectively of a job $i_{i,k}$ of $\tau_i$ that experiences a minimal preemtping instant. The former can be calculated as $h_{i}^{mp} = BR_i(BC_i, \tau_i)$ whereas $r_i^{mp}$ is given by $r_i^{mp} = SR_i(h_{i}^{mp}, \{}$).

Proof. In order to prove the theorem, let $h^{br} \in \alpha_i(\tau_i^{br})$ be the hold time of the job that assumes the best-case response time, and $\alpha_i^{br}$ the set of extra preemtping tasks that leads to such a hold time. Hence, it holds that $BR_i = SR_i(h^{br}, \tau_i)$. Furthermore, from Lemma 8, it also holds that

$$BR_i = SR_i(h^{br}, \tau_i) \geq SR_i^{ar}(h^{br}). \quad (7.15)$$

From Corollary 4, the best-case response time $BR_i$ is bounded from above by the shortest response time of a job of $\tau_i$ experiencing a minimal preemtping instant, and from below by the shortest hold time, i.e. $h_{i}^{mp} \leq BR_i \leq r_i^{mp}$. Moreover, $h^{br}$ is also constraint by such bounds; hence, $h_{i}^{mp} \leq h^{br} \leq BR_i \leq r_i^{mp}$. Based on the previous observation, we know that $h^{br} \in [h_{i}^{mp}, r_i^{mp}]$ and, therefore, it holds that

$$SR_i^{ar}(h^{br}) \geq \min_{h \in [h_{i}^{mp}, r_i^{mp}]} (SR_i^{ar}(h)) = BR_i^{lb}. \quad (7.16)$$

From inequalities (7.15) and (7.16), we conclude that $BR_i \geq BR_i^{lb}$ and, therefore, $BR_i^{lb}$ is a lower bound for the best-case response time of a task $\tau_i$ scheduled using FPTS. \qed

7.6 An algorithm for computing the lower bound for the best-case response time

Equation (7.14) gives a lower bound for the best-case response time of a task scheduled using FPTS. However, since the artificial hold time $h \in \mathbb{R}^+$ can take any value in the range $[h_{i}^{mp}, r_i^{mp}]$, the set of values where we have to find the minimum of the response times may be infinite. In order to make this computable, we present in this section an algorithm to derive the lower bound $BR_i^{lb}$ of the best-case response time presented in Equation (7.14) for a task $\tau_i$. In particular, this algorithm starts determining the shortest artificial response time $SR_i^{ar}(h)$ when $h = h_{i}^{mp}$ and only increases $h$ in discrete steps that can decrease the result of $SR_i^{ar}(h)$. If the result cannot longer be reduced, the algorithm stops and the minimum is found.

The algorithm is based on the fact that the response time of a job $i_{i,k}$ experiencing a minimal preemtping instant can be delayed by a previous job. In the introductory example, we denoted such a previous job as “tight” because it can immediately start upon activation. In Figure 7.2, this is the job of $\tau_4$ activated at time $t = -176$. Furthermore, the response time of $i_{i,k}$ can only be reduced when the artificial hold time is sufficiently large to allow an activation of a delaying task at the same time as the activation of the job that is tight (see Figure 7.3). Therefore, the algorithm determines how large the artificial hold time should be to allow such an activation of a delaying tasks and subsequently reduce the response time of $i_{i,k}$.

The procedure that we propose to derive the lower bound for the best-case response time is shown in Algorithm 1. Table 7.2 shows an overview of the variables used in the algorithm. This algorithm first initializes the artificial hold time $h$ with the hold time of a job of $\tau_i$ when experiencing a minimal preemtping instant in Line 2, defined as $h_{i}^{mp}$ in Theorem 7. Furthermore, in Line 3, the variable $r_i^{min}$ is initialized with the shortest artificial response time of a job when assuming such an
artificial hold time \( h \). \( r^\text{min} \) is the variable where the current minimum response time is stored. In case that \( r^\text{min} \) is higher than \( h \), it means that there is a previous job that is tight and, therefore, it is possible to increase \( h \) hopefully leading to a reduction in the response time. Hence, Lines 5 to 8 increase the value of the artificial hold time \( h \) using a discrete increment \( \Delta h \). More precisely, the job \( k^\text{max} \) of task \( r_i \) that is tight and induces some delay in the job experiencing the minimal preemption instant is determined in Line 5. For example, in Figure 7.2, this is the job of \( r_4 \) activated at time \( t = -176 \). Moreover, Lines 6 and 7 determine the necessary increment \( \Delta h \) for the artificial hold time \( h \) in order to reduce the response time. This time increment \( \Delta h \) is simply the time between the activation of \( k^\text{max} \) and the first activation of a delaying task after \( k^\text{max} \). For instance, consider again Figure 7.2. In such an example, \( k^\text{max} \) is activated at time \( t = -176 \) and the first activation of a delaying task occurs at time \( t = -174 \). Hence, the time increment is simply \( \Delta h = 2 \) for this example.

In Line 8, the artificial hold time \( h \) is incremented with such a discrete value. Finally, the response time of a job of \( r_i \) assuming the new artificial hold time \( h \) is determined in Line 9 and it is assigned to \( r^\text{min} \) only if it is smaller than its current value. If the new value of \( h \) is still smaller than the new value of \( r^\text{min} \), the procedure is repeated. Otherwise, the algorithm terminates and the minimum response time is found.

It is worth noting that, in Line 7 of the algorithm, the increment \( \Delta h \) is determined based on the minimum given a set of delaying tasks \( D_i \). Since the set of tasks \( D_i \) may be empty, we assume that the minimum of an empty set is infinite.

### Table 7.2: Terminology.

<table>
<thead>
<tr>
<th>Name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r^\text{min} )</td>
<td>Minimum response time of the job experiencing the minimal preemption instant for a specific artificial hold time ( h ).</td>
</tr>
<tr>
<td>( k^\text{max} )</td>
<td>The job of ( r_i ) with ( 1 \leq k^\text{max} \leq w_i ) that induces delay on the job experiencing the minimal preemption instant. This is the tight job that leads to the maximum value in the inner max of Equation (7.9).</td>
</tr>
<tr>
<td>( A_{i,k^\text{max}} )</td>
<td>The length of the interval between the activation time of the job ( r_i ) that is &quot;tight&quot; and leads to ( r^\text{min} ) and the start of the artificial hold time.</td>
</tr>
<tr>
<td>( \Delta h )</td>
<td>Hold time increment that can potentially reduce the minimum response time.</td>
</tr>
</tbody>
</table>

### Algorithm 1

*Algorithm to derive a lower bound for the best-case response time of task \( r_i \).*

1: procedure \( \text{bcrtLowerBound}(T_i) \)
2: \( h \leftarrow BH_i(BC_i, D_i) \);
3: \( r^\text{min} \leftarrow SR^\text{ar}_i(h) \);
4: while \( h < r^\text{min} \) do
5: \( k^\text{max} \leftarrow \) the job \( k \) with \( 1 \leq k \leq w_i \) that is "tight" and leads to \( r^\text{min} \);
6: \( A_{i,k^\text{max}} \leftarrow r^\text{min} + (k^\text{max} - 1)T_i - h \);
7: \( \Delta h \leftarrow \min_{d: \tau_d \in D_i} \left( A_{i,k^\text{max}} \mod T_d \right) \);
8: \( h \leftarrow h + \Delta h \);
9: \( r^\text{min} \leftarrow \min \{ r^\text{min}, SR^\text{ar}_i(h) \} \);
10: end while
11: return \( r^\text{min} \);

### 7.7 An example of the algorithm to derive \( BR^\text{lb}_i \)

Consider the set of tasks \( T_{7,3} \) with characteristics as described in Table 7.3. Note that tasks \( r_1 \) and \( r_2 \) are preempting tasks of \( r_4 \), whereas \( r_3 \) is a delaying task. We will now apply Algorithm 1 to derive a lower bound \( BR^\text{lb}_4 \) for the best-case response time of task \( r_4 \). Figure 7.4 shows an example of how each variable of the algorithm would be represented in a timeline for \( T_{7,3} \) at the beginning of
the procedure. As can be seen, \( h^{(0)} = 22 \) represents the initial value of the artificial hold time \( h \), and it is initially equal to the hold time of the last job of \( \tau_4 \) experiencing a minimal preemption instant. Furthermore, the job of \( \tau_4 \) activated at time \( t = -176 \) denoted as \( k^{\text{max}} \) induces some delay on the last job of \( \tau_4 \). Therefore, resulting in an initial response time \( r^{\text{min}}(0) = SR_{\text{ar}}(22) = 36 \).

Table 7.3: Characteristics of task-set \( T_{7.3} \).

<table>
<thead>
<tr>
<th>task</th>
<th>( T_i )</th>
<th>( C_i )</th>
<th>( \pi_i )</th>
<th>( \theta_i )</th>
<th>( w_i )</th>
<th>( WR_i )</th>
<th>( BR_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>35</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>35</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>50</td>
<td>20</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>62</td>
<td>20</td>
</tr>
<tr>
<td>( \tau_4 )</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>66</td>
<td>27</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P^{T_{7.3}} = 350 \) and \( U^{T_{7.3}} = 1 \).

Figure 7.4: Timeline for \( T_{7.3} \) during the first iteration of \( \text{bcrtLowerBound}(T_{7.3}, 4) \).

The algorithm now increments \( h \) till an activation of \( \tau_3 \) coincides with the activation of job \( \tau_3^{\text{max}} \) of \( \tau_4 \). By doing so, the response time of its last job can be reduced. This time increment can be calculated as \( \Delta h = (A_4, k^{\text{max}} \mod T_3) = 4 \), and it leads to \( h^{(1)} = h^{(0)} + \Delta h = 26 \). Next, the algorithm calculates the new value for the response time \( r^{\text{min}} \) with the new artificial hold time, i.e. \( r^{\text{min}}(1) = \min(r^{\text{min}}(0), SR_{\text{ar}}(26)) = 26 \). Figure 7.5 shows the timeline when the last job of \( \tau_4 \) has an artificial hold time \( h^{(1)} = 26 \). As can be seen, the activations of the delaying task \( \tau_3 \) was pushed to an earlier moment in time, allowing to decrease the response time of the last job of \( \tau_4 \). At this point, the artificial hold time \( h^{(1)} \) and the minimum response time \( r^{\text{min}}(1) \) are equal. Therefore, the algorithm terminates and returns \( r^{\text{min}}(1) = 26 \) as the lower bound for the best-case response time of \( \tau_4 \), i.e. \( BR_{lb}^{\tau_4} = 26 \). Note that the actual best-case response time is \( BR^{\tau_4} = 27 \).

Figure 7.5: Timeline for \( T_{7.3} \) showing the lower-bound for the best-case response time of task \( \tau_4 \) returned by \( \text{bcrtLowerBound}(T_{7.3}, 4) \).

Table 7.4 summarizes the intermediate results of Algorithm 1 when determining the lower
bound for the response time of task $r_4$ in task-set $T_{7.3}$. It is worth noting that, for this example, the lower bound was found after the first iteration. For examples in which the lower bound is found after executing multiple iterations see Appendix D. In particular, for an example in which the lower bound is tight with respect to the actual best-case response time see Appendix D.3.

Table 7.4: Intermediate results for $bcrlowerbound(T_{7.3}, 4)$.

<table>
<thead>
<tr>
<th>iteration</th>
<th>0*</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{\text{max}}$</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>$A_{i,k_{\text{max}}}$</td>
<td>-</td>
<td>154</td>
</tr>
<tr>
<td>$\Delta h$</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>$h$</td>
<td>22</td>
<td>26</td>
</tr>
<tr>
<td>$r_{\text{min}}$</td>
<td>36</td>
<td>26</td>
</tr>
</tbody>
</table>

* Iteration 0 denotes the initialization of the variables, i.e. Lines 2 and 3 in Algorithm 1.
In this chapter, we use the novel best-case response time analysis for FPTS presented in Section 6.4 to evaluate three lower bounds for the response time of a task \( \tau_i \in T \). Such lower bounds are the best-case computation time \( BC_i \), the best-case response time \( BR_i^{\mathcal{T} \setminus \mathcal{D}_i} \) when ignoring delaying tasks, i.e., the best-case response time of \( \tau_i \) in task-set \( T \setminus \mathcal{D}_i \), and the novel lower bound \( BR_i^{lb} \) presented in Section 7.5. Note that \( BR_i^{\mathcal{T} \setminus \mathcal{D}_i} \) can be calculated using the best-case response time analysis for FPPS presented in Section 3.3 because all tasks with a priority higher than \( \tau_i \) in \( T \setminus \mathcal{D}_i \) can preempt \( \tau_i \).

8.1 Set-up for the evaluation

For the evaluation, we generate systems containing \( n = 10 \) tasks. The periods for the tasks are randomly generated in the range \([100, 10000]\). In addition, the utilization \( U_i \) for each task in a system is selected using the UUnifast algorithm [2] and its computation time is simply determined by \( WC_i = BC_i = U_i \times T_i \). Furthermore, we use deadline monotonic priority assignment and we assign thresholds as high as possible while keeping the task-set schedulable. To this end, we use the Optimal Threshold Assignment algorithm in [3]. We set best-case deadlines equal to zero; hence, tasks always meet their best-case deadlines. For each experiment and for each parameter configuration, we generate a new set of 5000 schedulable task-sets. Finally, we only consider systems in which the lowest priority task has at least one delaying task.

8.2 Constraint deadlines

For the first experiment, we assign worst-case deadlines equal to periods for all the tasks, i.e. \( \forall 1 \leq i \leq n \), \( WD_i = T_i \). For each system, we determine whether the lower bounds \( BC_i \), \( BR_i^{\mathcal{T} \setminus \mathcal{D}_i} \) and \( BR_i^{lb} \) for the response time of the lowest priority task \( \tau_i \) are tight, i.e. if they are equal to its best-case response time \( BR_i \).

Figure 8.1 shows the fraction of the times the lower bounds are equal to the best-case response time for different utilizations. As can be seen, the number of times the best-case computation time \( BC_i \) is a tight lower bound decreases considerably when increasing the utilization above 0.85. On the other hand, observe that the lower bounds \( BR_i^{\mathcal{T} \setminus \mathcal{D}_i} \) and \( BR_i^{lb} \) seem to be excellent lower bounds when deadlines are equal to periods because they match exactly the best-case response time for all the evaluated cases, even for high utilizations.

We can conclude that, for deadlines equal to periods, using the the best-case response time analysis for FPPS to calculate the response time of a task \( \tau_i \) when ignoring delaying tasks is sufficient to determine a tight lower bound for most of the evaluated cases.
Figure 8.1: Fraction of the times the best-case response time of the lowest priority task is equal to a lower bound versus the utilization of the task-set considering $n = 10$ tasks and $WD_i = T_i$.

8.3 Arbitrary deadlines

For our second experiment, we only generate systems with a fixed utilization of $U^T = 0.98$. In addition, in order to test the tightness of the lower bounds for arbitrary deadlines, the worst-case deadline for each task is uniformly drawn from the range $[T_i - a \cdot (T_i - BC_i), T_i + a \cdot T_i]$, where the parameter $a \in [0, 1)$ is used to vary the range of possible deadlines. Figure 8.2 shows the results of this experiment for different values of $a$. As can be seen, $BC_i$ is only a tight lower bound in less than 24% of the cases for all values of $a$. Most likely due to the high utilization. On the other hand, $BR_i$ only differs from the actual best-case response time in at most 5% of the cases when increasing $a$, whereas the novel lower bound $BR_{i, lb}$ only yields deviating results in approximately 1.5% of the cases. This experiment therefore shows that, even for arbitrary deadlines and high utilizations, the best-case response time $BR_i$ when ignoring delaying tasks and using the analysis for FPPS is still a tight lower bound for the majority of the evaluated cases. The same holds for the novel lower bound $BR_{i, lb}$ achieving slightly better results.

We now investigate how close the lower bounds are to the actual best-case response time for the cases in which the best-case response time when ignoring delaying tasks $BR_i$ is not tight, i.e. for the 5% of the cases in which $BR_i$ differs from the actual best-case response time. For this experiment, we fixed the parameter to vary the worst-case deadlines to $a = 0.7$ while varying the utilization of the generated systems in the range $U^T \in (0.9, 1)$. Furthermore, we estimate how "close" a lower bound is to its actual best-case response in a given system simply by taking the division of the two, i.e.

$$\text{closeness}(R_{lb}) = \frac{R_{lb}}{BR_i},$$

(8.1)

for a given response time lower bound $R_{lb}$. Hence, a closeness equal to one means that the lower bound is tight.
Figure 8.2: Fraction of the times the best-case response time of the lowest priority task is equal to a lower bound for different values of $\alpha$ considering $n = 10$ tasks and task-set’s utilizations of $U^T = 0.98$.

Figure 8.3 shows the result of this experiment by means of box plots (a.k.a. box and whisker diagrams). Each diagram represents the data using five values. The bottom and the top horizontal lines in the diagram represent the minimum and maximum values respectively. The bottom edge in a square is the first quartile, whereas the top edge in a square is the third quartile of the values. Finally, the line in the middle of a square is the median value.

As can be seen from Figure 8.3, the closeness of the lower bounds reduces when increasing the utilization. Surprisingly, the best-case computation time $BC_i$ is relatively close to the actual best-case response time when considering $U^T = 0.9$, being at 87% of the actual value in the median value. For the same utilization, the median value of $BR_i^{\cup \phi_i}$ is at 97% of the actual best-case response time, and decreases to 90% for an utilization $U^T = 0.98$. Finally, the median value of the novel lower bound $BR_{lb_i}$ as well as the first quartile are equal to the best-case response time for all the evaluated utilizations. It is also worth noting that for $U^T = 0.90$, the minimum value in closeness of the novel lower bound $BR_{lb_i}$ still stays relatively close to $BR_i$, being at 81.5% of the actual value. However, the minimum closeness found for this utilization in $BR_i^{\cup \phi_i}$ and $BC_i$ are 70% and 66% respectively. For the high utilizations $U^T = 0.98$, the minimum closeness found for each lower bound drops considerably. In particular, we found a system in which $BR_i^{\cup \phi_i}$ is a little less than one third of the $BR_i$.
Figure 8.3: Closeness of the lower bounds to the actual best-case response time versus task-set's utilization. Deadlines vary according to $\alpha = 0.7$, and $n = 10$ tasks are considered per system.
Chapter 9

Conclusions

In this thesis, we studied the best-case response time for tasks scheduled using FPTS. To this end, we first proved, as an intermediate result towards the exact analysis, that the best-case response time for non-preemptive tasks is equal to its best-case computation time; hence, addressing our first research question (RQ1). This contribution was accepted as a work-in-progress (WiP) session paper for a conference [18]. Next, we proved exact best-case response time for independent and strictly periodic real-time tasks with arbitrary deadlines scheduled using FPTS. This addressed our second research question (RQ2) regarding the exact best-case analysis for FPTS. In order to formalize the analysis, we presented some facts derived from experimentation regarding the best-case behaviour of a task scheduled using FPTS. In particular, we identified that, unlike FPPS, the job of a task that assumes the best-case response time does not necessarily have the shortest hold time. Hence, a job assuming the best-case response time may experience additional preemptions by some preempting tasks. Based on this, we derived a property for an optimal instant expressed in terms of its higher priority tasks. In particular, this property for an optimal instant partitions the set of preempting tasks into the tasks causing additional preemptions in the job that assumes the best-case response time, and the remaining preempting tasks. Since the best-case response time can potentially be found in any partition of preempting tasks, our analysis explores all of them by determining the shortest response time that a job can assume for each partition. The best-case response time is then simply the minimum of such shortest response times. Due to the necessity of exploring all possible partitions of preempting tasks, the presented exact analysis has an exponential time complexity.

In order to provide a schedulability test that is computational tractable, we also presented and proved a novel lower bound for the best-case response time of a task. This lower bound, unlike the exact analysis, eliminates the need of partitioning the set of preempting tasks. Therefore, reducing the complexity of the analysis considerably.

In addition to our novel best-case analysis for FPTS, we used such analysis to evaluate the tightness of three lower bounds for the best-case response time of a task. Such lower bounds are the best-case computation time of a task, our novel lower bound, and the best-case response time when ignoring the tasks that cannot preempt another task. The latter can be calculated using the best-case analysis for FPPS. This evaluation was performed to address our third research question (RQ3) regarding the appropriate lower bounds for the response time of a task. The results of the evaluation showed that our novel lower bound and the best-case response analysis for FPPS are tight lower bounds for the majority of the considered cases. These lower bounds are particularly effective for tasks with deadlines equal to periods yielding the same value as the actual best-case response time for all the cases considered in the evaluation. In addition, for arbitrary deadlines and high utilizations their efficiency drops by only 5% for the best-case response analysis for FPPS and 1.5% for our novel lower bound. The novel best-case response time analysis for FPTS along with the evaluation of tightness of the best-case computation time and the best-case analysis for FPPS was submitted as a conference paper [17].

As future work, we aim to extend the best-case response time analysis for FPTS with the notion of activation jitter of a task.
Bibliography


Appendix A

Auxiliary Lemmas

Lemma 9. When \( \lim_{x \to X} f(x) \) is defined, and \( f(x) \) is strictly increasing in an interval \( (X, X + \gamma) \) for sufficiently small \( \gamma \in \mathbb{R}^+ \), then the following equation holds:

\[
\lim_{x \to X} \lceil f(x) \rceil = \left\lfloor \lim_{x \to X} f(x) \right\rfloor + 1.
\]  
(A.1)

Proof. The proof of this lemma can be found in [21].
Appendix B

Additional Proofs

The following proofs are adaptations of the proofs given in Section 6.3 of [20].

B.1 Proof for Lemma 4

**Lemma 4.** Let all tasks of a set $\mathcal{T}$ be strictly periodic and let $w \ell$ exist. Furthermore, let job $i_{\ell - 1}$ of $\tau_i$ have a hold time equal to $h \in 3^i(\mathcal{E}_i)$. For every $\ell \in \mathbb{N}$ a lower bound for the shortest response time $SR_i$ of $i_{\ell - 1}$ is given by

$$SR_i(h, \mathcal{E}_i) \geq \max_{1 \leq k \leq \ell} (GI_i(k \cdot BC_i, h, \mathcal{E}_i) - (k - 1)T_i) \quad (B.1)$$

**Proof.** Let $S_k$ be a sequence of $k$ jobs of $\tau_i$ with $1 \leq k \leq \ell$, e.g. $i_{\ell - k}, ..., i_{\ell - 1}$. Now let job $i_{\ell - 1}$ be the job in the sequence that experiences the property of optimal instant with extra preemptions by the tasks in $\mathcal{E}_i$ and hold time $H_{i_{\ell - 1}} = h$.

For the sequence $S_k$ of $k$ jobs of $\tau_i$, there should be at least $k \cdot BC_i$ time available for the execution of $\tau_i$ in the interval $[a_{i_{\ell - 1}}, f_{i_{\ell - 1}}]$. Otherwise, there will not be enough space to execute all jobs of the sequence $S_k$ before or at $f_{i_{\ell - 1}}$. From Definition 5, the shortest interval in which exactly $k \cdot BC_i$ time is available for $\tau_i$ is given by $GI_i(k \cdot BC_i, h, \mathcal{E}_i)$. Therefore, the latest time for the activation of $i_{\ell - k}$ is given by $f_{i_{\ell - 1}} - GI_i(k \cdot BC_i, h, \mathcal{E}_i)$, i.e.

$$a_{i_{\ell - k}} \leq f_{i_{\ell - 1}} - GI_i(k \cdot BC_i, h, \mathcal{E}_i),$$

Recall that $\tau_i$ is strictly periodic; hence, it holds that $a_{i_{\ell - 1}} = a_{i_{\ell - k}} + (k - 1) \cdot T_i$. The latest activation of job $i_{\ell - 1}$ is therefore

$$a_{i_{\ell - 1}} \leq f_{i_{\ell - 1}} - (GI_i(k \cdot BC_i, h, \mathcal{E}_i) - (k + 1) \cdot T_i),$$

which leads to

$$f_{i_{\ell - 1}} - a_{i_{\ell - 1}} \geq GI_i(k \cdot BC_i, h, \mathcal{E}_i) - (k + 1) \cdot T_i.$$ 

This latest activation gives rise to a lower bound $R_{i_{\ell - 1}}^{(k)}$ on the response time of $i_{\ell - 1}$, i.e.

$$R_{i_{\ell - 1}}^{(k)} = GI_i(k \cdot BC_i, h, \mathcal{E}_i) - (k + 1) \cdot T_i.$$ 

Each value of $R_{i_{\ell - 1}}^{(k)}$ represents a constraint for the response time of $i_{\ell - 1}$, i.e. a lower bound on the shortest response time $SR_{i_{\ell - 1}}$ of such a job. Every constraint must be satisfied to allow $i_{\ell - 1}$ to finalize at or before $f_{i_{\ell - 1}}$; hence, $\max_{1 \leq k \leq \ell} R_{i_{\ell - 1}}^{(k)}$ is a lower bound for $SR_{i_{\ell - 1}}$ for every $\ell$. $\square$
B.2 Proof for Lemma 5

In order to prove Lemma 5, we first derive a property from Lemma 4.

Let \( k^{\text{max}} \) denote a value for which a maximum is assumed by \( \max_{1 \leq k \leq w \ell_i} (G_1(k \cdot BC_i, h, \mathcal{E}_i) - (k - 1) \cdot T_i) \). Without loss of generality, we may assume \( f_{i-1} = 0 \). The corresponding latest activation time \( a_{i-\ell}(k^{\text{max}}) \) of job \( i_{i-\ell} \) with \( 1 \leq k \leq w \ell_i \) is now given by

\[
a_{i-\ell}(k^{\text{max}}) = -G_1(k^{\text{max}} \cdot BC_i, h, \mathcal{E}_i) - (k^{\text{max}} - 1) \cdot T_i - (k - 1) \cdot T_i
\]

From the proof of Lemma 4 we conclude the following.

**Corollary 5.** For \( 1 \leq k \leq w \ell_i \), at least an amount \( k \cdot BC_i \) is available for \( \tau_i \) in the interval \( (a_{i-\ell}(k^{\text{max}}), 0) \), i.e. \( \tau_i \) may execute \( k \) jobs in \( (a_{i-\ell}(k^{\text{max}}), 0) \).

**Lemma 5.** To determine the highest lower bound using (6.6), it is sufficient to consider at most \( w \ell_i \) jobs.

**Proof.** The proof is based on a contradiction argument. Assume there exists an \( \ell > w \ell_i \) such that

\[
G_1(\ell \cdot BC_i, h, \mathcal{E}_i) - (\ell - 1) \cdot T_i > \max_{1 \leq k \leq w \ell_i} (G_1(k \cdot BC_i, h, \mathcal{E}_i) - (k - 1) \cdot T_i),
\]

i.e.

\[
G_1(\ell \cdot BC_i, h, \mathcal{E}_i) > G_1(k^{\text{max}} \cdot BC_i, h, \mathcal{E}_i) - (k^{\text{max}} - \ell) \cdot T_i.
\]

Let \( i_{i-\ell} \) be activated relative to \( a_{i-\ell}(k^{\text{max}}) = -G_1(k^{\text{max}} \cdot BC_i, h, \mathcal{E}_i) - (k^{\text{max}} - \ell) \cdot T_i \), i.e.

\[
a_{i-\ell}(k^{\text{max}}) = -G_1(k^{\text{max}} \cdot BC_i, h, \mathcal{E}_i) - (k^{\text{max}} - \ell) \cdot T_i.
\]

We now partition \( (a_{i-\ell}(k^{\text{max}}), 0) \) in two intervals, i.e. \( (a_{i-\ell}(k^{\text{max}}), a_{i-\ell}(k^{\text{max}})) \) and \( (a_{i-\ell}(k^{\text{max}}), 0) \).

On the one hand, because \( G_1(\ell \cdot BC_i, h, \mathcal{E}_i) > a_{i-\ell}(k^{\text{max}}) \), there is not enough time available in \( (a_{i-\ell}(k^{\text{max}}), 0) \) to execute \( k \) jobs of \( \tau_i \), with \( 1 \leq k \leq w \ell_i \). Hence, we know that there is not enough time available in \( (a_{i-\ell}(k^{\text{max}}), 0) \) to execute \( \ell - k \) jobs of \( \tau_i \). The activation of \( i_{i-\ell} \) at \( a_{i-\ell}(k^{\text{max}}) \) will therefore give rise to a pending load larger than zero at \( a_{i-\ell}(k^{\text{max}}) \) for \( 1 \leq k \leq w \ell_i \). Stated differently, an activation of \( i_{i-\ell} \) at time \( a_{i-\ell}(k^{\text{max}}) \) gives rise to a level-\( i \) active period of a length larger than \( w \ell_i \). This contradicts the definition of \( w \ell_i \), our assumptions was therefore wrong, hence we only have to consider sequences of at most \( w \ell_i \) jobs.

B.3 Proof for Theorem 4

**Theorem 4.** Let job \( i_{i-1} \) of \( \tau_i \) have a hold time equal to \( h \in \mathcal{H}_i(\mathcal{E}_i) \) and let \( w \ell_i \) exists. The shortest response time \( SR_i(h, \mathcal{E}_i) \) of job \( i_{i-1} \) scheduled under FPTS is given by

\[
SR_i(h, \mathcal{E}_i) = \max_{1 \leq k \leq w \ell_i} (G_1(k \cdot BC_i, h, \mathcal{E}_i) - (k - 1) \cdot T_i).
\]

**Proof.** Given Lemmas 4 and 5, we have to prove that \( BR_i \) cannot be larger than \( \max_{1 \leq k \leq w \ell_i} (G_1(k \cdot BC_i, h, \mathcal{E}_i) - (k - 1) \cdot T_i) \). We first observe that \( G_1(k \cdot BC_i, h, \mathcal{E}_i) \) is the shortest interval before the completion of a job of \( \tau_i \) with hold time \( h \in \mathcal{H}_i(\mathcal{E}_i) \) in which an amount of time \( k \cdot BC_i \) is available for the execution of \( \tau_i \). Hence, the right-hand-side (RHS) of (B.2) determines the latest activation time for \( i_{i-1} \) allowing all sequences \( S_k \) of \( k \) jobs of \( \tau_i \) with \( 1 \leq k \leq w \ell_i \) to have enough time to execute in \( (a_{i-\ell}, 0) \). Moreover, this latest start time for \( i_{i-1} \) also allows every sequence \( S_\ell \) of \( \ell \) jobs of \( \tau_i \) with \( \ell > w \ell_i \) to have enough time to execute in \( (a_{i-\ell}, 0) \). Finally, because the analysis holds for every level-\( i \) active period, the RHS of (B.2) yields the highest lower bound for \( BR_i \), which proves the theorem.
Appendix C

Tools

In this chapter, we present an overview of the tools used during the Master’s project.

C.1 Grasp tool

The Grasp tool [9] was developed in the System Architecture and Networking (SAN) group in the department of Computer Science and Mathematics of TU/e. Grasp is a tool for visualizing and measuring the behavior of real-time systems. The Grasp Player receives a trace file containing a Tcl script with a recorded trace, and it displays the task execution. All the timelines shown in this document were generated using Grasp.

C.2 FPTS scheduler tool

Motivation

During the initial stage of the Master’s project, there was a need to visualize the execution of tasks scheduled using FPTS to understand their best-case behaviour. However, to manually schedule tasks is time consuming and prompt to errors. Therefore, we developed a scheduler for FPTS to facilitate this task.

Goal

The goal of this tool is to facilitate the creation of trace files with the execution of tasks scheduled using FPTS that can be visualized using the Grasp Player.

Description

The scheduler tool receives as input a file containing the time frame to schedule and the characteristics of a task-set, i.e. the period, computation times, priorities, thresholds and phasings for all the tasks. The scheduler then uses FPTS scheduling to determine the order in which the tasks will be executed during the predefined time frame. The tool then generates the corresponding trace file containing a Tcl script with the execution of the tasks in the specified time frame and, in addition, it generates the response time of each of the scheduled jobs. The generated trace file can subsequently be visualized using the Grasp Player. This new tool is developed in the C++ programming language.

We used the scheduler tool in combination with Grasp during the Master’s project to visualize the behavior of tasks under FPTS. Furthermore, these tools were particularly useful in the early stage of the project to find the facts presented in Section 5.1 regarding the best-case behaviour of tasks scheduled using FPTS.
C.3 Simulation tool

Motivation

In order to find interesting examples of task-sets where the best-case response time is not assumed by a trivial lower bound, we first had to be able to determine the best-case response time of a task. Without an exact analysis, however, the approach that we found more reliable is using brute force simulations.

Goal

This tool has the goal of providing the best-case response time of tasks scheduled using FPTS by means of brute force simulations.

Description

The simulation tool receives a file as an input containing the characteristics of a set of tasks, i.e. the periods, computation times, priorities and thresholds for all the tasks. Next, it simulates all possible integral phasings for each task except for the last task. For example, consider the set of tasks described in Table C.1. For this example, the tool simulates all integral phasings of the tasks $\tau_1$ and $\tau_2$ in the range $\varphi_1 \in [0, 35)$ and $\varphi_2 \in [0, 50)$, while fixing the phasing of task $\tau_3$ to $\varphi_3 = 0$. The shortest response time for each task for all the explored phasings $\varphi$ is then determined. This shortest response time is printed as an output as well as the phasing $\varphi$ in which the shortest response time was found. This tool is developed using the C++ programming language.

Table C.1: Characteristics of task-set $\mathcal{T}_{C1}$.

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>35</td>
<td>10</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>50</td>
<td>20</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{\mathcal{T}_{C1}} = 350$ and $U_{\mathcal{T}_{C1}} = 1$.

The simulation tool is built using the scheduler tool. For each simulation of a phasing $\varphi$ the schedule of the tasks is constructed for a time frame of four hyperperiods. Afterwards, the shortest response time for each task under a specific phasing is found using the forth hyperperiod. Since an hyperperiod is the length at which the schedule repeats itself after an initial start-up, it is sufficient to look for the shortest response time in one hyperperiod per phasing $\varphi$. However, in order to allow such an initial start-up, we use the forth hyperperiod.

C.4 Exploration tool

Motivation

In the early stage of the project, we identified the need of finding interesting examples of task-sets showing the differences between the best-case behaviour of FPPS and FPTS. For example, in FPTS the last job in a level-i active period is not necessarily the job with the shortest response time (Fact 4 in Section 5.1). Looking for such examples by hand, however, is very time consuming. Hence, we built an exploration tool to automate the search of examples.

Goal

The goal of this tool is to find interesting examples of task-sets that would potentially lead to some facts related with the best-case behaviour of tasks scheduled using FPTS.

Best-case response time of real-time tasks under FPTS 57
Description
This tool takes as inputs the number of tasks in the task-set, its utilization, the range of periods of the tasks, and the number of delaying tasks for the last job. Next, it randomly generates task-sets with such characteristics. For each generated task-set, the simulation tool was used to determine the shortest response time for each task. In addition, the best-case response time analysis for FPPS was implemented to determine the best-case response time of the last task when ignoring delaying tasks. The exploration tool then prints as an output all task-sets in which the shortest response time of the lowest priority task using simulations is greater than the best-case response time of such a job when ignoring delaying tasks. Most of the examples presented in this thesis were found using the exploration tool. This tool is developed using the C++ programming language.

C.5 Evaluation tool

Motivation
The motivation of this tool is to help in the process of answering our second research question (RQ2) related with the appropriate lower bounds for the response time of a task scheduled using FPTS. Hence, we are required to determine how tight the lower bounds are with respect to the exact best-case response time.

Goal
The goal of this tool is to evaluate the tightness of some lower bounds for the response time of a task scheduled using FPTS with respect to its actual best-case response time.

Description
The evaluation tool is used to perform the evaluation of the lower bounds for the response time of a task scheduled using FPTS (see Chapter 8). This tool is an extension of the RTA tool developed in the System Architecture and Networking (SAN) group in the department of Computer Science and Mathematics of TU/e. The evaluation tool has inherited implementations of the RTA tool, such as the worst-case response times analysis for FPPS and FPTS, and the Optimal Threshold Assignment (OTA) algorithm in [3] to determine maximum thresholds assignment. Furthermore, we extended this analysis with the exact best-case response time analysis for FPPS and FPTS, as well as the novel lower bound for FPTS.

The evaluation tool randomly generate systems containing $n = 10$ tasks. The periods for the tasks are uniformly drawn in the range $[100, 10000]$. In addition, the utilization $U_i$ for each task in a system is selected using the UUnifast algorithm [2] and its computation time is simply determined by $WC_i = BC_i = U_i \times T_i$. Afterwards, the priority of each task is selected using a deadline monotonic priority assignment and the thresholds using the OTA algorithm. For each experiment and for each parameter configuration, a new set of 5000 schedulable task-sets is generated in which the lowest priority task has at least one delaying task. Furthermore, for each experiment the lower bounds are evaluated, i.e. it is determined the number of times the lower bounds are tight and the closeness to the actual best-case response times. A graph is then generated with the results.
Appendix D

Additional Examples

In this chapter, we apply the exact best-case response time analysis as well as the novel lower bound to some additional examples. We start with an example that shows that the best-case response time can also be found in a job assuming its worst-case hold time, i.e. when all its preempting tasks are extra preempting tasks. Furthermore, the second example shows that the largest hold time in a set of hold times $H_i(E_i)$ can also lead to the best-case response time. For all the examples, the lower bound is found after multiple iterations of Algorithm 1. Furthermore, the lower bound is not tight for the first two examples. In the last example, however, we show a case where the novel lower bound is tight.

D.1 Example 1

Consider the task-set $T_{D.1}$ given in Table D.1. We now apply our novel best-case response time analysis for FPTS to determine the best-case response time of task $\tau_4$. Note that $\tau_1$ and $\tau_2$ are preempting tasks of $\tau_4$, whereas $\tau_3$ is a delaying task. Table D.2 shows the hold times for all possible partitions of preempting tasks into the sets of extra and minimal preempting tasks. These hold times were found using the analysis in Section 6.1.

Table D.1: Characteristics of task-set $T_{D.1}$.

<table>
<thead>
<tr>
<th>task</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$\pi_i$</th>
<th>$\theta_i$</th>
<th>$w\ell_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>100</td>
<td>18</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>100</td>
<td>22</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>20</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>170</td>
<td>60</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

The hyperperiod is $P_{T_{D.1}} = 1700$ and $U_{T_{D.1}} \approx 0.95$.

Table D.2: Hold times that a job of $\tau_4$ can assume.

<table>
<thead>
<tr>
<th>$E_j$</th>
<th>${}$</th>
<th>${\tau_1}$</th>
<th>${\tau_2}$</th>
<th>${\tau_1, \tau_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_4(E_j)$</td>
<td>(60)</td>
<td>(78)</td>
<td>(82)</td>
<td>(100)</td>
</tr>
</tbody>
</table>

Based on a hold time of a job of $\tau_4$, we can derive its shortest response time $SR_4(h, E_j)$ using Theorem 4. From Table D.1, note that the worst-case number of jobs of $\tau_4$ in a level-4 active period is $w\ell_4 = 2$. The intermediate results for $SR_4(h, E_j)$ are depicted in Table D.3. Furthermore, the maximum of such intermediate results is the shortest response time for the given configuration. The best-case response time is then simply the minimum of such shortest response times according to Corollary 2, i.e. $BR_4 = \min\{108, 104, 104, 100\} = 100$. For this example, the best-case response
time of \( \tau_4 \) is assumed when experiencing its worst-case hold time, i.e. when experiencing extra preemptions by both, \( \tau_1 \) and \( \tau_2 \).

Table D.3: Intermediate results for \( SR_4(h, E_4) \).

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SR_4(60, { }) )</td>
<td>108</td>
<td>62</td>
</tr>
<tr>
<td>( SR_4(78, { \tau_1 }) )</td>
<td>104</td>
<td>58</td>
</tr>
<tr>
<td>( SR_4(82, { \tau_2 }) )</td>
<td>104</td>
<td>58</td>
</tr>
<tr>
<td>( SR_4(100, { \tau_1, \tau_2 }) )</td>
<td>100</td>
<td>54</td>
</tr>
</tbody>
</table>

The value selected as the shortest response time \( SR_4(h, E_4) \) is highlighted in bold. That is the maximum of all its intermediate results.

We now apply the algorithm for the novel lower bound given in Section 7.6. Table D.4 shows the value of the variables in each iteration when applying the algorithm to task \( \tau_4 \) in task-set \( \mathcal{T}_{D.1} \). The best-case response time lower bound for \( \tau_4 \) is therefore \( BR_{\text{lb}}^4 = 84 \). For this case, the lower bound is not tight.

Table D.4: Intermediate results for \( bcrtLowerBound(\mathcal{T}_{D.1}, 4) \).

<table>
<thead>
<tr>
<th>iteration</th>
<th>0*</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_{\text{max}} )</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( A_{i,k_{\text{max}}} )</td>
<td>-</td>
<td>48</td>
<td>36</td>
</tr>
<tr>
<td>( \Delta h )</td>
<td>-</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>( h )</td>
<td>60</td>
<td>68</td>
<td>84</td>
</tr>
<tr>
<td>( r_{\text{min}} )</td>
<td>108</td>
<td>104</td>
<td>\textbf{84}</td>
</tr>
</tbody>
</table>

* Iteration 0 denotes the initialization of the variables, i.e. Lines 2 and 3 in Algorithm 1.

### D.2 Example 2

For this example consider the task-set \( \mathcal{T}_{D.5} \) given in Table D.5. Table D.6 shows the hold times of task \( \tau_4 \) for all possible partitions of preemting tasks into the sets of extra and minimal preempting tasks. These hold times were found using the analysis in Section 6.1.

Table D.5: Characteristics of task-set \( \mathcal{T}_{D.5} \).

<table>
<thead>
<tr>
<th>task</th>
<th>( T_i )</th>
<th>( C_i )</th>
<th>( \pi_i )</th>
<th>( \theta_i )</th>
<th>( wL_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>12</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>21</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>( \tau_4 )</td>
<td>120</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P_{\mathcal{T}_{D.5}} = 840 \) and \( U_{\mathcal{T}_{D.5}} = 1 \).

Table D.6: Hold times that a job of \( \tau_4 \) can assume.

<table>
<thead>
<tr>
<th>( E_4 )</th>
<th>{ }</th>
<th>{ \tau_1 }</th>
<th>{ \tau_2 }</th>
<th>{ \tau_1, \tau_2 }</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{H}_4(E_4) )</td>
<td>{4}</td>
<td>{10,23}</td>
<td>{17}</td>
<td>{36}</td>
</tr>
</tbody>
</table>

Using Theorem 4, we now derive the shortest response times of \( \tau_4 \) for each hold time. The intermediate results for \( SR_4(h, E_4) \) are depicted in Table D.7. Furthermore, the maximum of such
intermediate results is the shortest response time for the given configuration. The best-case response time is then simply the minimum of such shortest response times according to Corollary 2, i.e. \( BR_4 = \min(50, 71, 23, 62, 36) = 23 \).

Table D.7: Intermediate results for \( SR_4(h, x_i) \).

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( SR_4(4, { } ) )</td>
<td>25</td>
<td>50</td>
<td>37</td>
<td>1</td>
<td>49</td>
<td>13</td>
<td>38</td>
</tr>
<tr>
<td>( SR_4(10, { r_1 }) )</td>
<td>46</td>
<td>71</td>
<td>58</td>
<td>22</td>
<td>70</td>
<td>34</td>
<td>59</td>
</tr>
<tr>
<td>( SR_4(23, { r_1 }) )</td>
<td>23</td>
<td>-13</td>
<td>-47</td>
<td>-1</td>
<td>-35</td>
<td>11</td>
<td>-25</td>
</tr>
<tr>
<td>( SR_4(17, { r_2 }) )</td>
<td>38</td>
<td>25</td>
<td>50</td>
<td>37</td>
<td>62</td>
<td>49</td>
<td>13</td>
</tr>
<tr>
<td>( SR_4(36, { r_1, r_2 }) )</td>
<td>36</td>
<td>0</td>
<td>-34</td>
<td>12</td>
<td>-22</td>
<td>24</td>
<td>-12</td>
</tr>
</tbody>
</table>

The value selected as the shortest response time \( SR_4(h, x_i) \) is highlighted in bold. That is the maximum of all its intermediate results.

Table D.8: Intermediate results for \( bcrtLowerBound(\mathcal{T}_{\mathcal{D}, 5}, 4) \).

<table>
<thead>
<tr>
<th>iteration</th>
<th>0*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_{\text{max}} )</td>
<td>-</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>( A_{1,k_{\text{max}}} )</td>
<td>-</td>
<td>166</td>
<td>753</td>
<td>498</td>
<td>243</td>
</tr>
<tr>
<td>( \Delta h )</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( h )</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>( r_{\text{min}} )</td>
<td>50</td>
<td>38</td>
<td>26</td>
<td>14</td>
<td>14</td>
</tr>
</tbody>
</table>

* Iteration 0 denotes the initialization of the variables, i.e. Lines 2 and 3 in Algorithm 1.

### D.3 Example 3

In this section, we revisit the example given in Section 6.1 and we apply the algorithm for the best-case response time lower bound to the last task in task-set \( \mathcal{T}_{6,1} \). Table D.9 shows the characteristics of this task-set.

Table D.9: Characteristics of task-set \( \mathcal{T}_{6,1} \).

<table>
<thead>
<tr>
<th>task</th>
<th>( T_i )</th>
<th>( C_i )</th>
<th>( \pi_i )</th>
<th>( \theta_i )</th>
<th>( w_{\ell_i} )</th>
<th>( BR_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>35</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>49</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>56</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>( \tau_4 )</td>
<td>70</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>36</td>
</tr>
</tbody>
</table>

The hyperperiod is \( P_{\mathcal{T}_{6,1}} = 1960 \) and \( U_{\mathcal{T}_{6,1}} = 0.98 \).

After applying the analysis for the best-case lower bound given in Section 7.6, we find that \( BR_{\text{lb}}_4 = 36 \) is a lower bound for the response time of task \( \tau_4 \in \mathcal{T}_{6,1} \). For this example, this lower bound is tight, i.e. it is equal to the actual best-case response time of task \( \tau_4 \).
Table D.10: Intermediate results for \textit{bertLowerBound}(\tau_{6,1,4}).

<table>
<thead>
<tr>
<th>iteration</th>
<th>0*</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{\text{max}}$</td>
<td>-</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$A_{1,k_{\text{max}}}$</td>
<td>-</td>
<td>228</td>
<td>80</td>
</tr>
<tr>
<td>$\Delta h$</td>
<td>-</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>$h$</td>
<td>22</td>
<td>26</td>
<td>50</td>
</tr>
<tr>
<td>$\varphi_{\text{min}}$</td>
<td>40</td>
<td>36</td>
<td>36</td>
</tr>
</tbody>
</table>

* Iteration 0 denotes the initialization of the variables, i.e. Lines 2 and 3 in Algorithm 1.