Towards pragmatic solutions for two-level hierarchical scheduling. Part I: a basic approach for independent applications

Towards pragmatic solutions for two-level hierarchical scheduling Part I: A basic approach for independent applications

Reinder J. Bril

Technische Universiteit Eindhoven (TU/e),
Den Dolech 2, 5612 AZ Eindhoven,
The Netherlands
r.j.bril@tue.nl

Abstract

Resource reservation has been proposed in the literature as a means to prevent temporal interference between applications. With applications consisting of one or more real-time tasks, resource reservation inherently involves multiple levels of scheduling, i.e. a scheduling hierarchy.

This document considers two-level hierarchical scheduling for independent applications in cost-constrained systems, using fixed-priority pre-emptive scheduling (FPPS) for tasks. We aim at pragmatic solutions for such systems, which allow efficient implementations of budgets and simple analysis for tasks. Given the complexity and pessimism of the analysis for tasks for existing approaches and based on the observation that budgets are design artifacts, we consider a basic approach, where budgets are time-triggered and all budgets have the same period. Using jitter analysis as an example, it is shown that existing analytic results for real-time tasks under FPPS can be easily converted to two-level hierarchical scheduling for this approach by viewing the unavailability of a budget as an artificial highest priority task. We show that the conversion equally well applies to the approaches described in [12, 17, 18], amongst others. We briefly consider an advanced approach where budgets have harmonic periods, and show that the conversion is not straightforward for such an approach.

1 Introduction

1.1 Context and motivation

Today, fixed-priority pre-emptive scheduling (FPPS) is a de-facto standard in industry for scheduling systems with real-time constraints. A major shortcoming of FPPS, however, is that temporary or permanent faults occurring in one application can hamper the execution of other applications. To resolve this shortcoming, the notion of *resource reservation* [14] has been proposed. Resource reservation provides *isolation* between applications, effectively protecting an application against other, malfunctioning applications. With applications consisting of one or more real-time tasks, resource reservation inherently involves multiple levels of scheduling, i.e. a scheduling hierarchy [17]. Analytic methods for hierarchical scheduling are a topic of current research [1, 6, 9, 12, 18]. In these papers, each application consists of a set of independent tasks and a separate budget is allocated to each application. In this document, we consider two-level hierarchical scheduling, with FPPS for tasks.

We observe that the literature can be subdivided in two groups depending on whether or not the principle of *locality of scheduling analysis* [17] is adhered to, i.e. whether or not no assumptions are made about the scheduling policy for budgets. This principle facilitates independent design, analysis and validation of systems, but comes at the cost of pessimism of the analysis. This principle is adhered to by [12, 18], and their results therefore also apply for FPPS for budgets. Conversely, [1, 6, 9, 17] assume FPPS for budgets. Moreover, [1, 6, 9] assume arbitrary phasing of budgets, and assume that the other characteristics of budgets are known, as well as their priorities. Although these assumptions allow for a reduction of the pessimism in the analysis, this reduction comes at the cost of an increase in the complexity of the analysis, whilst the analysis

remains pessimistic [6]. Note that for a specific phasing of budgets, i.e. when the phasing is known, analysis based on arbitrary phasing of budgets typically gives rise to additional pessimism.

With FPPS for budgets, several server models have been considered for budgets, e.g. [1, 17, 9] consider deferrable servers, [9, 17] sporadic servers, and [6, 9] periodic servers. These server models differ in the rules for the preservation and replenishment of server capacity, and therefore in the complexity and overhead of their implementation. From these models, the periodic server is the easiest to implement and has the lowest overhead.

We aim at pragmatic solutions for reservation-based resource management for cost-constrained systems, allowing for an efficient implementation of budgets and for simple analysis for tasks. To this end, we observe that budgets are *design artifacts*, e.g. they serve as a means to solve the interference problem between applications. We therefore conclude that the selection of the scheduling policy and server model for budgets are primarily design decisions, which are constrained by system requirements and have to be balanced with other design considerations. Similarly, we can also constrain the selection of budget characteristics during design. In this document, we assume FPPS and a periodic server model for budgets, and consider a basic approach, where budgets are time-triggered rather than event-triggered and have equal periods. We briefly consider an advanced approach, where budgets have harmonic periods.

1.2 Contributions

We briefly review existing approaches in the area of two-level hierarchical FPPS (H-FPPS), in which we compare the *worst-case* (i.e. *minimum*) *available capacity* $WC^{\beta}(t)$ of a budget β in an interval of length t. The term worst-case available capacity originates from [3] and is also used in [6]. Other terms used in the literature for this notion are *least supply function* $S^*(t)$ [15], *characteristic function* $Z_S(t)$ of a periodic server S [12], (*resource*) *supply bound function* $\mathbf{sbf}_\Gamma(t)$ for a periodic resource Γ [18], and *availability function* $A\_S(t)$ of a server S [1].

We consider a basic approach for two-level H-FPPS, that can be applied to systems that allow for a selection of a common period and a specific phasing for budgets. We show that the worst-case available capacity for this approach improves on the results of existing approaches. Next, we present analysis for real-time tasks for the approach. In particular, we show that we can deal with a budget during the analysis of tasks by viewing the *unavailability* of the budget as an artificial highest priority task, similar to the worst-case response time analysis as presented in [17]. This view significantly simplifies the analysis, and allows for a straightforward conversion of analytic results for tasks under FPPS to tasks of independent applications under two-level H-FPPS, which we illustrate by converting jitter analysis of independent tasks. We subsequently show how to convert jitter analysis for the approaches described in [12, 17, 18], sketch how this view applies to [1], and compare our analysis with the analysis in [9]. Finally, we show by means of an example that the conversion of analytic results is not straightforward for an advanced approach where all budget have harmonic periods.

1.3 Structure

The remainder of this document is organized as follows. We start by giving real-time scheduling models for FPPS and two-level H-FPPS in Section 2. Next, we recapitulate analysis of tasks under FPPS in Section 3. We review existing approaches for two-level H-FPPS in Section 4, and consider the worst-case available capacity for each approach. A basic approach for budget scheduling is the topic of Section 5, and the analysis of tasks for this approach is presented in Section 6. In Section 7, we revisit and extend the analysis of tasks as presented in [12, 17, 18], describe how to convert analytic results for FPPS to two-level H-FPPS for [1], and compare our analysis with the analysis in [9]. We consider an advanced approach for budget scheduling in Section 8. Finally, we conclude the document in Section 9.

2 Real-time scheduling models

2.1 A basic model for FPPS

We assume a single processor and a set $\mathcal{T}$ of n periodically released, independent tasks $\tau_1, \tau_2, \ldots, \tau_n$ with unique, fixed priorities. At any moment in time, the processor executes the highest priority task that has work pending.

Each task τ_i is characterized by a (*release* or *activation*) *period* $T_i \in \mathbb{R}^+$, a *worst-case computation time* $WC_i \in \mathbb{R}^+$, a *best-case computation time* $BC_i \in \mathbb{R}^+$, a (*relative*) *deadline* $D_i \in \mathbb{R}^+$, where $D_i \leq T_i$, and a *phasing* $\varphi_i \in \mathbb{R}$. The set of phasings φ_i is termed the phasing φ of the task set $\mathcal{T}$. For ease of presentation the worst-case and best-case computation times of tasks in the examples are identical, i.e. $WC_i = BC_i$, and we therefore simply use C_i in the examples.

An *activation time* is a time at which a task τ_i becomes ready for execution. An activation of a task is also termed a *job*. The first job of task τ_i is activated at time φ_i and is referred to as job zero. The activation of job k of τ_i therefore takes place at time $a_{ik} = \varphi_i + kT_i$, $k \in \mathbb{Z}$. The (*absolute*) *deadline* of job k of τ_i takes place at $d_{ik} = a_{ik} + D_i$. The *finalization time* f_{ik} of job k of τ_i is the time at which τ_i ends the execution of that job. The *response time* R_{ik} of job k of τ_i is defined as the length of the time span between the activation time of that job and its finalization time, i.e. $R_{ik} = f_{ik} - a_{ik}$.

We assume that we do not have control over the phasing φ, for instance since the tasks are released by external events, so we assume that any arbitrary phasing may occur. This assumption is common in real-time scheduling literature [10, 11, 13]. We also assume other standard basic assumptions [13], i.e. tasks are ready to run at the start of each period and do not suspend themselves, tasks will be preempted instantaneously when a higher priority task becomes ready to run, a job of task τ_i does not start before its previous job is completed, and the overhead of context switching and task scheduling is ignored. Finally, we assume that the deadlines are hard, i.e. each job of a task must be completed at or before its deadline. For notational convenience, we assume that the tasks are given in order of decreasing priority, i.e. task τ_1 has highest priority and task τ_n has lowest priority.

Given these definitions and assumptions, we define the following derived notions. The *worst-case response time* WR_i and the *best-case response time* BR_i of a task τ_i are the largest and the smallest response time of any of its jobs under arbitrary phasing, respectively, i.e.

$$WR_i = \sup_{\varphi,k} R_{ik}(\varphi) \quad \text{and} \quad BR_i = \inf_{\varphi,k} R_{ik}(\varphi). \tag{1}$$

Note that the response time R_{ik} has been parameterized in these equations to denote its dependency on the phasing φ. A *critical instant* [13] and an *optimal* (or *favourable*) *instant* [5, 16] of a task are defined to be (hypothetical) instants that lead to the worst-case and the best-case response time for that task, respectively. The (*processor*) *utilization factor* U of a task set $\mathcal{T}$ is the fraction of the processor time spent on the execution of that task set [13]. The fraction of processor time spent on executing a periodic task τ_i with a fixed computation time C_i is C_i/T_i, and is termed the *utilization factor* U_i^τ of task τ_i, i.e.

$$U_i^\tau = \frac{C_i}{T_i}. \tag{2}$$

The *cumulative utilization factor* U_i for periodic tasks τ_1 till τ_i with fixed computation times is the fraction of processor time spent on executing these tasks, and is given by

$$U_i = \sum_{j \le i} U_j^\tau = \sum_{j \le i} \frac{C_j}{T_j}. \tag{3}$$

Therefore, U is equal to the cumulative utilization factor U_n for n periodic tasks.

Because we distinguish best-case and worst-case computation times in this document, we get best-case and worst-case versions of the various notions of utilization, i.e.

$$WU = WU_n = \sum_{j \le n} WU_j^\tau = \sum_{j \le n} \frac{WC_j}{T_j} \quad \text{and} \quad BU = BU_n = \sum_{j \le n} BU_j^\tau = \sum_{j \le n} \frac{BC_j}{T_j}. \tag{4}$$

Based on the notion worst-case response time and the assumption that deadlines are hard, we conclude that a set $\mathcal{T}$ of n periodic tasks can be scheduled if and only if

$$\bigvee_{i=1,\ldots,n} WR_i \le D_i. \tag{5}$$

Equation (5) represents an *exact* schedulability condition for $\mathcal{T}$. A *necessary* schedulability condition for $\mathcal{T}$ is given by

$$WU \le 1. \tag{6}$$

2.2 A periodic server model for budgets

In essence, a periodic budget may simply be viewed as an artificial periodic task with a fixed computation time (i.e $WC = BC$), a worst-case (relative) deadline equal to the period (i.e. $D = T$), and a *specific* phasing, i.e. we assume in this document that we have control over the phasing of budgets. Because we also assume FPPS for budgets, all notions and assumptions of our scheduling model for tasks can be reused for budgets. The fact that budgets are allocated and provided to applications is not important for our model for budgets.

Hence, we assume a single processor and a set $\mathcal{B}$ of n periodically released, independent budgets $\beta_1, \beta_2, \ldots, \beta_n$ with fixed, unique priorities. Note that we assume that the *capacity* of a budget (i.e. the equivalent of a computation time of a task) is fixed, hence a budget is always entirely consumed, and never discarded or suspended. Because we only consider hard real-time tasks in this document, we assume that any remaining capacity is idled away [9].

We assume a one-to-one relationship between budgets and applications, i.e. budget β_α is associated with application $\mathcal{A}_\alpha$ with $1 \leq \alpha \leq n$. An application $\mathcal{A}_\alpha$ is assumed to consist of n_α tasks. A *necessary* schedulability condition for $\mathcal{A}_\alpha$ with an associated budget β_α [4] is now given by

$$WU_\alpha^{\mathcal{A}} \leq U_\alpha^{\beta}, \tag{7}$$

where $WU_\alpha^{\mathcal{A}}$ and U_α^{β} denote the (worst-case) cumulative utilization factor of $\mathcal{A}_\alpha$ and β_α, respectively.

2.3 Concluding remarks

In the remainder of this document, we assume that task sets and budget sets satisfy the necessary worst-case schedulability condition as expressed by Equation (6). Moreover, we assume that applications satisfy the necessary schedulability condition as expressed by Equation (7). In this document, we will use a subscript to denote the application to which a task belongs and with which a budget is associated, e.g. $\tau_{\alpha,i}$ denotes task i of application $\mathcal{A}_\alpha$ and β_α denotes the budget associated with $\mathcal{A}_\alpha$. Moreover, we will use superscripts τ and β to denote characteristics of tasks and budgets, respectively, e.g. $C_{\alpha,i}^{\tau}$ denotes the computation time of task $\tau_{\alpha,i}$ and C_α^{β} denotes the capacity of budget β_α.

3 Recapitulation of analysis for FPPS

In this section we recapitulate analysis of tasks under FPPS, based on [2, 5, 10, 13, 16], amongst others. We start with worst-case and best-case response time analysis, using an example task set $\mathcal{T}_1$ with characteristics as given in Table 1 for illustration purposes. The response time analysis is exact under arbitrary phasing, but typically pessimistic under a specific phasing. We subsequently recapitulate finalization jitter and activation jitter.

task	$T = D$	C	WR	BR
τ_1	3	1	1	1
τ_2	4	1	2	1
τ_3	10	3	8	5

Table 1. Task characteristics of $\mathcal{T}_1$ and worst-case and best-case response times of tasks.

3.1 Worst-case response time analysis

A *critical instant* of a task τ_i is assumed when τ_i is simultaneously released with all tasks with a higher priority than τ_i [13]. Figure 1(a) shows a timeline of $\mathcal{T}_1$ with critical instants for all tasks. Because the worst-case response times of all tasks are smaller than their deadlines, we conclude from this timeline that $\mathcal{T}_1$ is schedulable. From this notion of critical instant, it

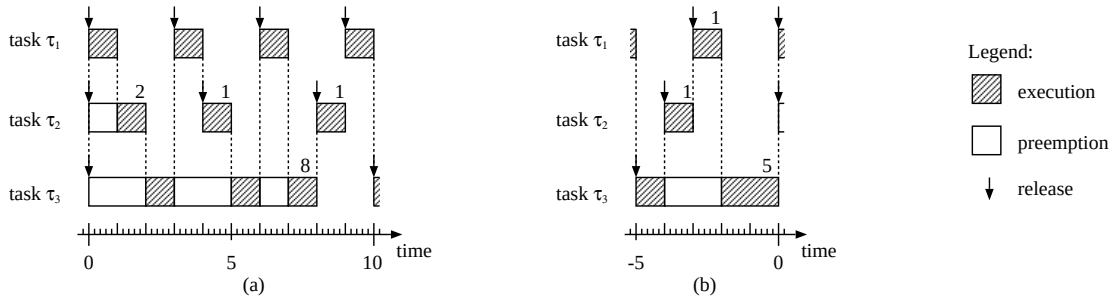

Figure 1. Timelines of $\mathcal{T}_1$ with (a) critical instants for all tasks and (b) an optimal instant for τ_3. The numbers to the top right corner of the boxes denote the response time of the respective releases.

has been derived in [10] that the *worst-case response time* WR_i of task τ_i is given by the smallest value $x \in \mathbb{R}^+$ satisfying

$$x = WC_i + \sum_{1 \le j < i} \left\lceil \frac{x}{T_j} \right\rceil WC_j. \tag{8}$$

Such a smallest value exists for task τ_i if and only if $WU_{i-1} < 1$; see, for example, [3]. Because we assume $WU \le 1$ and $WC_i > 0$ for all tasks, $WU_{i-1} < 1$ holds for all tasks. To calculate WR_i, we can use an iterative procedure based on recurrence relationships [2].

$$\begin{aligned} wr_i^{(0)} &= WC_i \\ wr_i^{(l+1)} &= WC_i + \sum_{1 \le j < i} \left\lceil \frac{wr_i^{(l)}}{T_j} \right\rceil WC_j, \quad l = 0, 1, \ldots \end{aligned}$$

The procedure is stopped when the same value is found for two successive iterations or when the deadline is exceeded. In the latter case, task τ_i is not schedulable.

3.2 Best-case response time analysis

An *optimal instant* of a task τ_i is assumed when the completion of τ_i coincides with the simultaneous release of all tasks with a higher priority than τ_i [5, 16]. Figure 1(b) shows a timeline of $\mathcal{T}_1$ with an optimal instant for task τ_3.

The *best-case response time* BR_i of task τ_i is given by the largest value $x \in \mathbb{R}^+$ satisfying

$$x = BC_i + \sum_{1 \le j < i} \left(\left\lceil \frac{x}{T_j} \right\rceil - 1 \right) BC_j. \tag{9}$$

Such a smallest value exists for task τ_i if and only if $BU_{i-1} < 1$; see [3]. Because $BU \le WU$ by definition, and we assume $WU \le 1$ and $BC_i > 0$ for all tasks, the relation $BU_{i-1} < 1$ trivially holds for all tasks. The best-case response time BR_i of task τ_i can be found by the following iterative procedure, which stops when the same value is found for two successive iterations.

$$\begin{aligned} br_i^{(0)} &= WR_i \\ br_i^{(l+1)} &= BC_i + \sum_{1 \le j < i} \left(\left\lceil \frac{br_i^{(l)}}{T_j} \right\rceil - 1 \right) BC_j, \quad l = 0, 1, \ldots \end{aligned}$$

3.3 Jitter analysis

The *worst-case* (*absolute*) *finalization jitter* FJ_i of a periodically released task τ_i is defined as

$$FJ_i = \sup_{\varphi,k,l} (f_{ik}(\varphi) - f_{il}(\varphi) - (k-l)T_i). \tag{10}$$

For a strictly periodically released task τ_i, this can be rewritten to

$$\begin{aligned} FJ_i &= \sup_{\varphi,k,l} ((f_{ik}(\varphi) - (\varphi_i + kT_i) - (f_{il}(\varphi) - (\varphi_i + lT_i))) \\ &= \sup_{\varphi,k,l} ((f_{ik}(\varphi) - a_{ik}(\varphi_i)) - (f_{il}(\varphi) - a_{il}(\varphi_i))) \\ &= \sup_{\varphi,k,l} (R_{ik}(\varphi) - R_{il}(\varphi)). \end{aligned} \tag{11}$$

Because the largest and smallest response times are not necessarily taken on for the same phasing, an upper bound on FJ_i is given by

$$FJ_i \le \sup_{\varphi,k} R_{ik}(\varphi) - \inf_{\varphi,l} R_{il}(\varphi) = \{(1)\}\ WR_i - BR_i. \tag{12}$$

For task τ_3 of $\mathcal{T}_1$, we find $FJ_3 \le WR_3 - BR_3 = 8 - 5 = 3$.

We note that Equation (11) would also be a sensible definition for *worst-case* (*absolute*) *response jitter* RJ_i of a periodic task τ_i. We observe that such a definition of the notion of worst-case (absolute) response jitter RJ_i differs from the notions of *absolute response time jitter* RTJ_i^{abs} in [8] and *absolute finishing jitter* AFJ_i in [7]. In [8], absolute response time jitter has been defined as

$$\mathrm{RTJ}_i^{abs} = \max_k R_{ik} - \min_k R_{ik}. \tag{13}$$

The definition of absolute finishing jitter in [7] is similar. The main difference between these two notions and our notion is that [7, 8] assume a specific phasing whereas we assume arbitrary phasing.

Next to finalization jitter, there can also be *activation* (or *release*) *jitter*. In this case, the releases of a task τ_i do not take place strictly periodically, with period T_i, but we assume they take place somewhere in an interval of length AJ_i that is repeated with period T_i. More specifically, the activation times satisfy

$$\sup_{k,l}(a_{ik}(\varphi_i) - a_{il}(\varphi_i) - (k-l)T_i) \leq AJ_i, \tag{14}$$

where φ_i now denotes the start of the interval of length AJ_i in which job zero of task τ_i is activated, rather than the time at which job zero is activated, i.e. $\varphi_i + kT_i \leq a_{ik} \leq \varphi_i + AJ_i + kT_i$. We now assume $D_i \leq T_i - AJ_i$, since otherwise there may be too little time between two successive releases to complete the task. In case of activation jitter, the analysis to derive worst-case and best-case response times, as well as finalization jitter, is slightly altered. For worst-case and best-case response times, we extend the analysis as described in [2] and [5, 16], respectively.

A *critical instant* of a task τ_i is assumed when τ_i is simultaneously released with all tasks with a higher priority than τ_i, all those task with a higher priority experience a maximum release delay at that simultaneous release and a minimum release delay at subsequent releases. The *worst-case response time* WR_i of task τ_i with activation jitter AJ_i of $\mathcal{T}$ is given by the smallest value $x \in \mathbb{R}^+$ satisfying

$$x = WC_i + \sum_{1 \leq j < i} \left\lceil \frac{x + AJ_j}{T_j} \right\rceil WC_j. \tag{15}$$

Similar to the case without activation jitter, there exists a smallest value if and only if $WU_{i-1} < 1$ and the recursive equation can than be solved by means of an iterative procedure, starting with a lower bound.

An *optimal instant* of a task τ_i is assumed when the completion of τ_i coincides with the simultaneous release of all tasks in $\mathcal{T}$ with a higher priority than τ_i, all those tasks with a higher priority experience a maximal release delay at that simultaneous release and a minimal release delay at previous releases. The *best-case response time* BR_i of task τ_i is given by the largest value $x \in \mathbb{R}^+$ satisfying

$$x = BC_i + \sum_{1 \leq j < i} \left(\left\lceil \frac{x - AJ_j}{T_j} \right\rceil - 1 \right)^+ BC_j, \tag{16}$$

where the notation w^+ stands for $\max(w, 0)$. Similar to the case without activation jitter, there exists a smallest value if $WU \leq 1$ and the recursive equation can than be solved by means of an iterative procedure, starting with an upper bound.

With activation jitter, we now derive from Equation (10)

$$\begin{aligned}
FJ_i &= \sup_{\varphi,k,l}(f_{ik}(\varphi) - f_{il}(\varphi) - (k-l)T_i) \\
&= \sup_{\varphi,k,l}((a_{ik}(\varphi_i) + R_{ik}(\varphi)) - (a_{il}(\varphi_i) + R_{il}(\varphi)) - (k-l)T_i) \\
&= \sup_{\varphi,k,l}(a_{ik}(\varphi_i) - a_{il}(\varphi_i) - (k-l)T_i + R_{ik}(\varphi) - R_{il}(\varphi)) \\
&\leq \sup_{\varphi_i,k,l}(a_{ik}(\varphi_i) - a_{il}(\varphi_i) - (k-l)T_i) + \sup_{\varphi,k} R_{ik}(\varphi) - \inf_{\varphi,l} R_{il}(\varphi).
\end{aligned}$$

Given the worst-case and best-case response times of task τ_i including the effect of release jitter of higher priority tasks, and given the release jitter of τ_i itself, its worst-case (absolute) finalization jitter is therefore bounded by

$$FJ_i \leq AJ_i + WR_i - BR_i. \tag{17}$$

We note that for periodically released tasks with activation jitter, finalization jitter differs from response time jitter. We merely mention that [7, 8] do not consider activation jitter.

As an example of finalization jitter with activation jitter, consider a task set $\mathcal{T}_1'$ with the same characteristics as $\mathcal{T}_1$, except that task τ_2 now has an activation jitter $AJ_2 = 1$ and a deadline $D_2 = T_2 - AJ_2 = 4 - 1 = 3$. Because the worst-case response times of τ_1 and τ_2 are independent of the activation jitter AJ_2 of τ_2 and $WR_2 \leq D_2$, tasks τ_1 and τ_2 remain schedulable. Figure 2 illustrates timelines of $\mathcal{T}_1'$ with a critical instant and an optimal instant for τ_3. From this figure, we conclude that the finalization jitter FJ_3 of task τ_3 is bounded by $FJ_3 \leq AJ_3 + WR_3 - BR_3 = 0 + 9 - 4 = 5$.

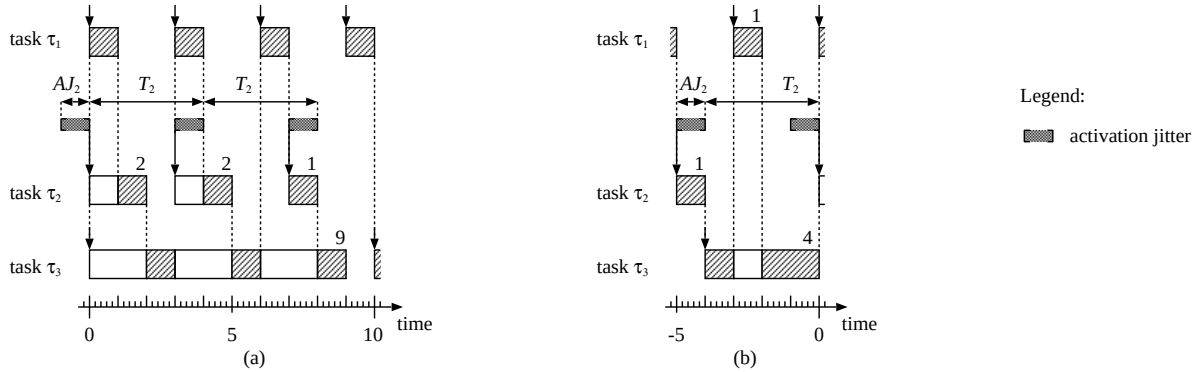

Figure 2. Timelines of $\mathcal{T}_1'$ with (a) a critical instant and (b) an optimal instant for task τ_3.

4 Recapitulation of analysis for two-level H-FPPS

In this section, we first briefly describe the worst-case response time of a task of an application with an associated budget using the notion of worst-case (i.e. minimum) available capacity, and subsequently compare the worst-case available capacity of existing approaches in the literature.

4.1 Worst-case response time analysis of tasks

Equation (8) for the worst-case response time of a task τ_i of a set $\mathcal{T}$ assumes that the entire processor is available to $\mathcal{T}$. When only the capacity of a budget β_α is available to the tasks of an application $\mathcal{A}_\alpha$, we can simply replace the x at the left hand side of Equation (8) by the worst-case (i.e. minimum) available capacity $WC_\alpha^\beta(x)$, as explained in [1]. Hence, the worst-case response time $WR_{\alpha,i}^\tau$ of a task $\tau_{\alpha,i}$ of application $\mathcal{A}_\alpha$ with associated budget β_α is given by the smallest $x \in \mathbb{R}^+$ satisfying

$$WC_\alpha^\beta(x) = WC_{\alpha,i}^\tau + \sum_{i \leq j < i} \left\lceil \frac{x}{T_{\alpha,j}^\tau} \right\rceil WC_{\alpha,j}^\tau. \tag{18}$$

For the approaches considered in the section, such a smallest value exists for $\tau_{\alpha,i}$ when $WU_{\alpha,i-1} < U_\alpha^\beta$. Because we assume $WU_\alpha^\mathcal{A} \leq U_\alpha^\beta$, $WU_{\alpha,i-1} < U_\alpha^\beta$ holds for all tasks of $\mathcal{A}_\alpha$.

4.2 Worst-case available capacity analysis

We now determine the *worst-case available capacity* $WC^\beta(t)$ that becomes available in an interval of length t from a budget β. We first give a general characterization of a situation in which a minimum amount of capacity becomes available. Next, we describe four classes of worst-case assumptions that are made in the literature for this situation, and illustrate the assumptions by means of an example set $\mathcal{B}_1$ of three budgets. For ease of presentation, these budget characteristics are chosen to be identical to the characteristics of our example set $\mathcal{T}_1$ of tasks as shown in Table 1. Finally, we determine the worst-case available capacity for each of these classes, and characterize a critical instant for a task of the application associated with β.

For $t > 0$, we may assume without loss of generality that the interval has an overlap with at least two periods of the budget β [6]. A minimum amount of capacity becomes available in an interval of length $t > 0$ when the capacity becomes available *as early as possible* in a first period of β overlapping with the interval and *as late as possible* in the last overlapping period.

Without any knowledge about scheduling at the budget level [12, 18], the capacity of β is assumed to become entirely available at the beginning in the first period and at the end in the last period, as illustrated in Figure 3(a) for two consecutive

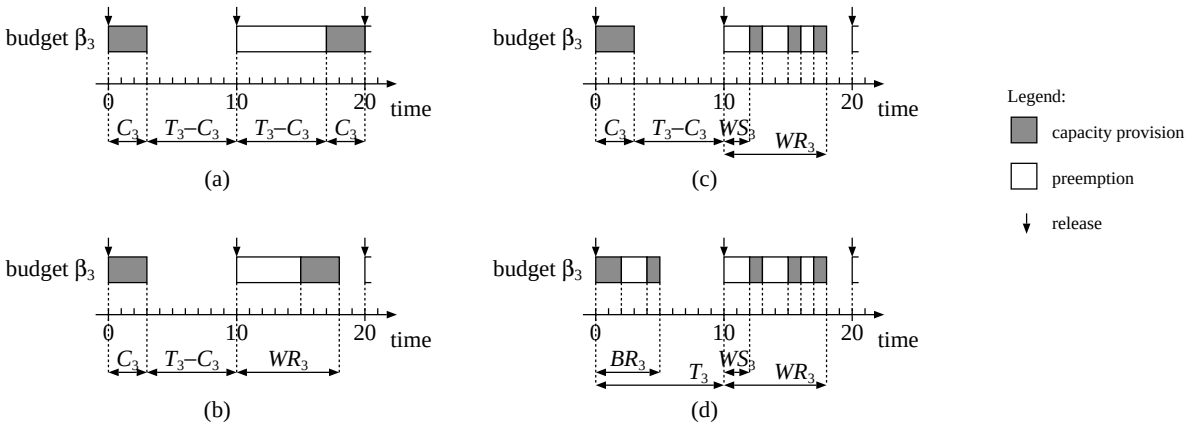

Figure 3. Worst-case assumptions for available capacity analysis for budget β_3 of our example $\mathcal{B}_1$ for two consecutive periods of β_3 according to (a) [12, 17, 18], (b) [1], (c) [9], and (d) [6].

periods of β_3 of our example $\mathcal{B}_1$. In [17], the same assumptions are made for the availability of the capacity of a budget based on a deferrable server model. For FPPS at the budget level, the capacity in the last period will become available no later than its worst-case response time see Figure 3(b). The analysis in [1] corresponds to these assumptions by choosing a so-called 'initial latency' equal to $WR_3^\beta - C_3^\beta$. The availability in the last period can be improved by using worst-case analysis techniques for that period [9]; see Figure 3(c). The term WS_3 is this figure denotes the *worst-case start time* of budget β_3 [3]. Finally, the availability in the first period can be improved by using best-case analysis techniques for that period [6]; see Figure 3(d).

We now turn to the worst-case available capacity for each of these four classes of assumptions. Let the *worst-case exact upper bound* T^{WUB} be defined as the largest value of t for which $WC^\beta(t) = 0$, i.e. $T^{\text{WUB}} = \sup\{t \mid WC^\beta(t) = 0\}$. For the four classes of assumptions, we derive from Figure 3 that $T_3^{\text{WUB}} = 2(T_3 - C_3) = 14$ for (a), $T_3^{\text{WUB}} = T_3 - C_3 + WR_3 - C_3 = 12$ for (b), $T_3^{\text{WUB}} = T_3 - C_3 + WS_3 = 9$ for (c), and $T_3^{\text{WUB}} = T_3 - BR_3 + WS_3 = 7$ for (d). For $t \geq T^{\text{WUB}}$, worst-case situations for t and $t + kT$ only differ in the number of overlapping periods of budget β, being k. Hence, the shape of $WC^\beta(t)$ is periodic with period T for $t \geq T^{\text{WUB}}$, i.e.

$$WC^\beta(t + kT) = WC^\beta(t) + kC.$$

Given the periodicity of $WC^\beta(t)$, we still need to determine $WC^\beta(t)$ for $t \in [T^{\text{WUB}}, T^{\text{WUB}} + T)$. For classes (a, b, c), we can simply derive $WC^\beta(t)$ by considering a fixed starting point for an interval of length $t \in [T^{\text{WUB}}, T^{\text{WUB}} + T)$, being the assumed completion of the budget in the first interval, and by subsequently determining the amount of capacity provided in the overlap with the last interval. For classes (a, b, c), a *critical instant* of a task $\tau_{\alpha,i}$ is assumed when $\tau_{\alpha,i}$ is simultaneously released with all tasks in $\mathcal{A}_\alpha$ with a higher priority than $\tau_{\alpha,i}$, and that simultaneous release coincides with the start of an interval with a minimum amount of available capacity of β_α, i.e. with the completion of budget β_α in the first period overlapping with that interval.

Unlike classes (a, b, c), it is not straightforward to determine a starting point for the interval for class (d) or a critical instant of a task $\tau_{\alpha,i}$. As examples, consider Figure 3(d) and two intervals of length 7 and 13, respectively. For a start of the intervals at time $t_s = 2$, we find available capacities equal to 1 and 2, respectively, whereas for a start at time $t_s = BR_3 = 5$, we find 0 and 3. As described in [6], we can now determine $WC^\beta(t)$ by a so-called *cognac-glass algorithm*. For our example, it suffices to consider two starting points of the interval, being $t_s = 2$ and $t_s = BR_3 = 5$, and determine the minimum of the amount of capacity provided in the overlapping intervals for both cases.

An overview of the worst-case available capacity $WC^\beta(t)$ for the four classes is shown in Figure 4. From these graphs for $WC^\beta(t)$, we conclude that the results gradually improve from class (a) till class (d). Unfortunately, the complexity of the worst-case response time analysis of tasks also increases from class (a) till class (d). Moreover, the worst-case available capacity analysis of class (d) need not yield a tight bound, and the worst-case response time analysis of tasks is therefore typically pessimistic [6].

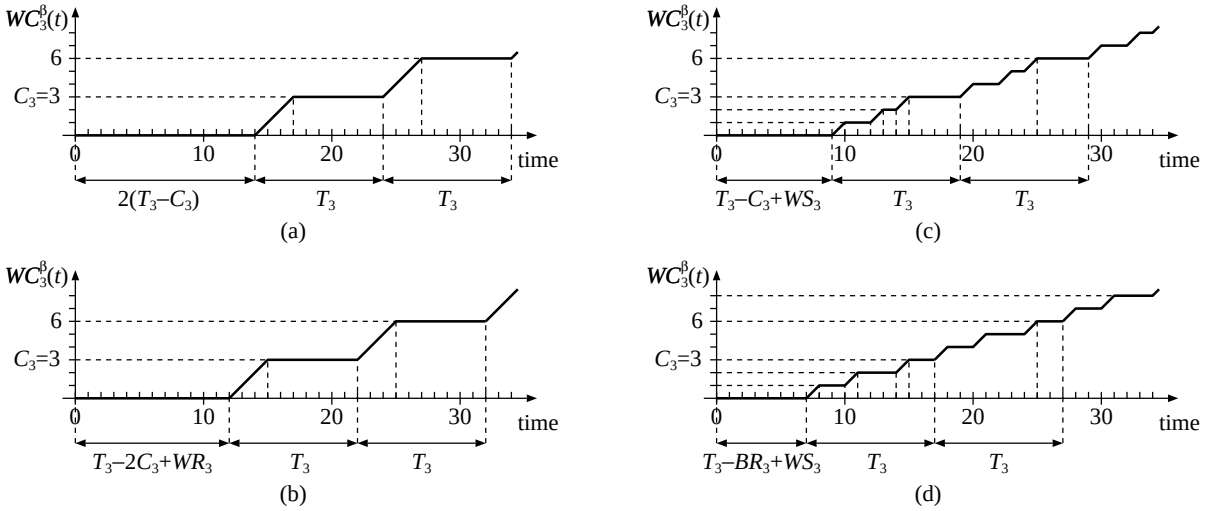

Figure 4. Worst-case available capacity $WC_3^\beta(t)$ in an interval of length t from a budget β_3 of our example $\mathcal{B}_1$ according to (a) [12, 17, 18], (b) [1], (c) [9], and (d) [6].

5 A basic approach for budget scheduling

Given the complexity of the response time analysis of tasks for classes (c, d) and the inherent pessimism of that analysis, combined with the fact that budgets are design artifacts, we consider a class (e) of assumptions and present the available capacity analysis for this class.

5.1 A class (e) of assumptions

For class (e), we assume FPPS at the budget level with arbitrary but fixed priority assignments. Moreover, we assume that all budgets have the same period and the same phasing, and therefore only differ in their capacity.

These assumptions have five major advantages. Firstly, because the periods of all the budgets are the same, the schedulability condition for budgets becomes trivial, i.e. a simple test whether or not the cumulative utilization $U = \sum_{1 \le \alpha \le n} C_\alpha^\beta / T_\alpha^\beta$ for budgets is at most 1. Secondly, because the capacities are fixed, the capacities of the budgets become available strictly periodically. Thirdly, because the phasing of all budgets are equal, the capacities become available non-pre-emptively, minimizing the cost of budget pre-emptions. Hence, fourthly, the available capacity analysis of budgets becomes trivial, in particular because that analysis for a budget is completely independent of the other budgets. Note that the latter two advantages also hold for any phasing satisfying $0 \le \varphi_i^\beta \le \sum_{j<i} C_j^\beta$, amongst others. Finally, the schedulability analysis of tasks using worst-case response time analysis becomes much simpler, because we only need to consider the budget associated with the application to which the tasks belong. We observe that this basic approach shares the last two advantages with the approaches described in [12, 17, 18].

5.2 Worst-case available capacity analysis

We will illustrate the available capacity analysis for class (e) by assuming that the periods of all budgets are equal to $T_3 = 10$. To facilitate comparison with the other classes, we assume that the capacities of β_1' and β_2' satisfy $C_1' + C_2' = WR_3 - C_3$. As a result, the worst-case available capacity $WC^\beta(t)$ of classes (a, b) remains unchanged, and $WC^\beta(t)$ for classes (c, d) become equal to $WC^\beta(t)$ of class (b).

Figure 5 illustrates that the capacity of β_3 becomes available periodically and non-pre-emptively. The worst-case start time WS_3' of budget β_3 is equal to the sum of the capacities of the higher priority budgets, i.e. $WS_3' = C_1' + C_2'$. The worst-case available capacity $WC^\beta(t)$ is illustrated in Figure 6. Note that the worst-case upper bound $T_3^{\mathrm{WUB}} = T_3 - C_3 = 7$. Compared to Figures 4(a) and 4(b), the graph in Figure 6 is shifted to the left with an amount $T_3 - C_3$ and $WR_3 - C_3$, respectively. Hence, we conclude that the result of class (e) improves on the existing results. Similar to classes (a, b, c), a *critical instant* of a task

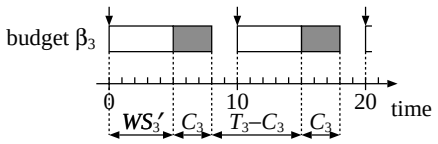

Figure 5. Assumptions for available capacity of budget β_3 for the basic approach.

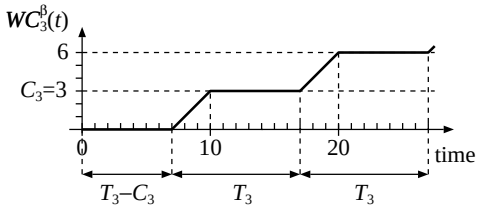

Figure 6. Worst-case available capacity $WC_3^{\beta}(t)$ in an interval of length t for budget β_3 when all periods are equal to T_3.

$\tau_{\alpha,i}$ for class (e) is assumed when $\tau_{\alpha,i}$ is simultaneously released with all task in $\mathcal{A}_\alpha$ with a higher priority than $\tau_{\alpha,i}$, and that simultaneous release coincides with the completion of budget β_α.

6 Analysis for basic approach

In this section, we will show for the basic approach that we can deal with a budget β_α during the analysis of tasks of an application $\mathcal{A}_\alpha$ by viewing the *unavailability* of the capacity of β_α as an artificial highest priority task $\tau_{\alpha,0}$, similar to the worst-case response time analysis in [17]. For this basic approach, the response time analysis of tasks is exact under arbitrary phasing. For illustration purposes, we use an application $\mathcal{A}_\alpha$ with two tasks $\tau_{\alpha,1}$ and $\tau_{\alpha,2}$ and an associated budget β_α with characteristics as given in Table 2.

	$T=D$	C
β_α	3	2
$\tau_{\alpha,1}$	4	1
$\tau_{\alpha,2}$	10	3

Table 2. Budget and task characteristics of budget β_α and tasks of application $\mathcal{A}_\alpha$.

6.1 Worst-case response time analysis

Figure 7(a) illustrates a timeline for β_α and $\mathcal{A}_\alpha$ with a critical instant for $\tau_{\alpha,2}$. From this figure, we draw the conclusion that the provision of the capacity C_α^β of budget β_α can be dealt with by viewing the *unavailability* of C_α^β as an artificial highest priority task $\tau_{\alpha,0}$ with a period $T_{\alpha,0}^\tau = T_\alpha^\beta$ and a computation time $C_{\alpha,0}^\tau = T_\alpha^\beta - WR_\alpha^\beta + WS_\alpha^\beta = T_\alpha^\beta - C_\alpha^\beta$, as illustrated in Figure 7(b). Note that the computation time $C_{\alpha,0}^\tau$ includes both the worst-case start time WS_α^β of budget β_α and the length $T_\alpha^\beta - WR_\alpha^\beta$ of the interval during which β_α is depleted. The *worst-case response time* $WR_{\alpha,i}^\tau$ of task $\tau_{\alpha,i}$ of application $\mathcal{A}_\alpha$ is now given by the smallest value $x \in \mathbb{R}^+$ satisfying

$$x = WC_{\alpha,i}^\tau + \sum_{0 \le j < i} \left\lceil \frac{x}{T_{\alpha,j}^\tau} \right\rceil WC_{\alpha,j}^\tau. \tag{19}$$

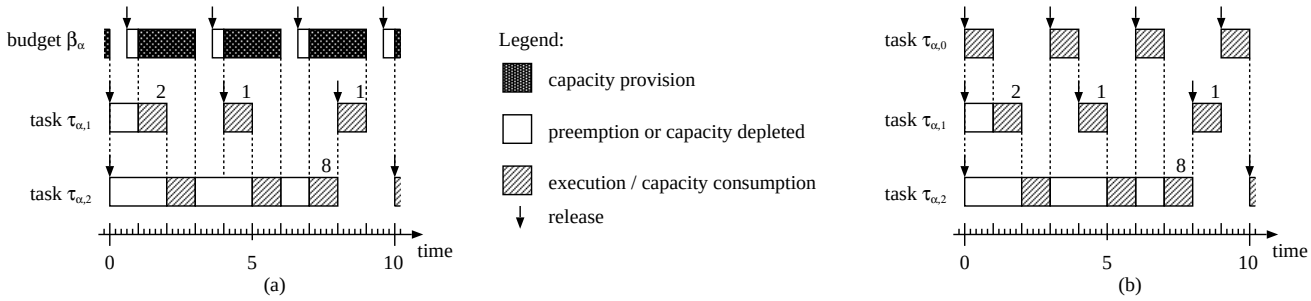

Figure 7. Timelines with a critical instant of task $\tau_{\alpha,2}$ for (a) budget β_α and application $\mathcal{A}_\alpha$ and for (b) an extension $\mathcal{A}'_\alpha$ of $\mathcal{A}_\alpha$ with an artificial task $\tau_{\alpha,0}$ that models the unavailability of capacity of β_α.

Alternatively, we can introduce a dedicated term $WI^\beta_\alpha(x)$ which denotes the 'worst-case interference' of task $\tau_{\alpha,0}$ in $\mathcal{A}'_\alpha$ or the 'worst-case unavailability' of the capacity C^β_α of budget β_α, i.e.

$$x = WC^\tau_{\alpha,i} + WI^\beta_\alpha(x) + \sum_{1 \le j < i} \left\lceil \frac{x}{T^\tau_{\alpha,j}} \right\rceil WC^\tau_{\alpha,j}, \tag{20}$$

where $WI^\beta_\alpha(x)$ is given by

$$WI^\beta_\alpha(x) = \left\lceil \frac{x}{T^\beta_\alpha} \right\rceil (T^\beta_\alpha - C^\beta_\alpha). \tag{21}$$

As usual, the recursive equation can be solved by means of an iterative procedure, starting with a lower bound.

In this document, we prefer the latter two equations (20) and (21) with a dedicated term $WI^\beta_\alpha(x)$ above Equation (19) for three reasons. Firstly, the variable j only ranges over the tasks of application $\mathcal{A}_\alpha$ in Equation (20) whereas it ranges over the tasks of $\mathcal{A}'_\alpha$ including the artificial task in Equation (19). Secondly, because the worst-case available capacity WC^β_α can be conveniently expressed in terms of WI^β_α as

$$WC^\beta_\alpha(x) = x - WI^\beta_\alpha(x), \tag{22}$$

the relation between Equations (18) and (20) is obvious. Finally, Equation (20) can be reused for classes (a) and (b), as we will see in a later section.

We finally observe that we can also *derive* a critical instant for a task of $\mathcal{A}_\alpha$ from the fact that an artificial task $\tau_{\alpha,0}$ models the unavailability of capacity of budget β_α. Note that although a budget is assumed to have a specific phasing and task $\tau_{\alpha,0}$ therefore has a specific phasing as well, we can still apply standard worst-case response time analysis under arbitrary phasing for the tasks of $\mathcal{A}_\alpha$, because all tasks of $\mathcal{A}_\alpha$ are assumed to have an arbitrary phasing.

6.2 Best-case response time analysis

For best-case response time analysis, we first consider the case with the artificial task $\tau_{\alpha,0}$, and next convert back to the case with a budget β_α. Hence, we *derive* an optimal instant for a task of an application $\mathcal{A}_\alpha$ with an associated budget β_α.

Figure 8(b) illustrates a timeline with an optimal instant for task $\tau_{\alpha,2}$ for the extension $\mathcal{A}'_\alpha$ of $\mathcal{A}_\alpha$ with the artificial task $\tau_{\alpha,0}$ that models the unavailability of the capacity C^β_α of the budget β_α. The equivalent timeline with an optimal instant for task $\tau_{\alpha,2}$ for budget β_α and application $\mathcal{A}_\alpha$ is shown in Figure 8(a). Hence, an *optimal instant* of a task $\tau_{\alpha,i}$ is assumed when the completion of $\tau_{\alpha,i}$ coincides with the simultaneous release of all tasks in $\mathcal{A}_\alpha$ with a higher priority than $\tau_{\alpha,i}$, and that simultaneous release coincides with the completion of budget β_α. Moreover, the *best-case response time* $BR^\tau_{\alpha,i}$ of task $\tau_{\alpha,i}$ of application $\mathcal{A}_\alpha$ is therefore given by the largest value $x \in \mathbb{R}^+$ satisfying

$$x = BC^\tau_{\alpha,i} + \sum_{0 \le j < i} \left(\left\lceil \frac{x}{T^\tau_{\alpha,j}} \right\rceil - 1 \right) BC^\tau_{\alpha,j}. \tag{23}$$

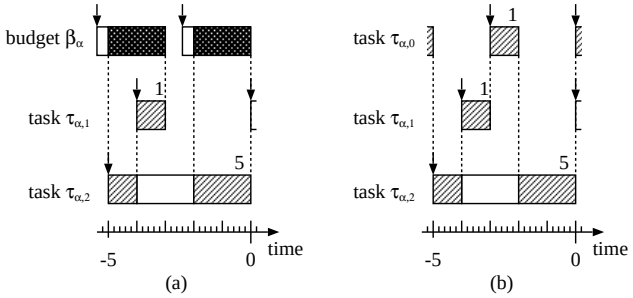

Figure 8. Timelines with an optimal instant of task $\tau_{\alpha,2}$ for (a) budget β_α and application $\mathcal{A}_\alpha$ and for (b) an extension $\mathcal{A}'_\alpha$ of $\mathcal{A}_\alpha$ with an artificial task $\tau_{\alpha,0}$ that models the unavailability of capacity of β_α.

Alternatively, we can introduce a dedicated term term $BI^\beta_\alpha(x)$ which denotes the 'best-case interference' of task $\tau_{\alpha,0}$ in $\mathcal{A}'_\alpha$ or the 'best-case unavailability' of the capacity C^β_α of budget β_α, i.e.

$$x = BC^\tau_{\alpha,i} + BI^\beta_\alpha(x) + \sum_{1\le j<i} \left(\left\lceil \frac{x}{T^\tau_{\alpha,j}} \right\rceil - 1 \right) BC^\tau_{\alpha,j}, \tag{24}$$

where $BI^\beta_\alpha(x)$ is given by

$$BI^\beta_\alpha(x) = \left(\left\lceil \frac{x}{T^\beta_\alpha} \right\rceil - 1 \right) (T^\beta_\alpha - C^\beta_\alpha). \tag{25}$$

As usual, the recursive equation can be solved by means of an iterative procedure, starting with an upper bound.

Similarly to the worst-case response time analysis, we prefer the latter two equations (24) and (25) above Equation (23).

6.3 Jitter analysis

As illustrated in the previous sections, the equivalence of budget β_α and an artificial task $\tau_{\alpha,0}$ for the response time analysis of tasks of $\mathcal{A}_\alpha$ enabled an easy conversion of response time analysis results of a set of tasks under FPPS to tasks belonging to an application with a dedicated budget under two-level H-FPPS. Given the worst-case and best-case response time analysis, the conversion of jitter analysis of tasks is easy as well.

For the worst-case (absolute) finalization jitter $FJ^\tau_{\alpha,i}$ of a strictly periodically released task $\tau_{\alpha,i}$ of an application $\mathcal{A}_\alpha$, we can convert Equations (10) and (12) by simply adding subscripts and superscripts. Similarly, we can convert Equation (14) for the activation jitter $AJ^\tau_{\alpha,i}$. For the response time analysis of tasks with activation jitter, we now assume $D^\tau_{\alpha,i} \le T^\tau_{\alpha,i} - AJ^\tau_{\alpha,i}$, since otherwise there may be too little time between two successive releases to complete the task. The descriptions of *critical instant* and *optimal instant* can also be easily transliterated. As an example, a *critical instant* of a task $\tau_{\alpha,i}$ is assumed when $\tau_{\alpha,i}$ is simultaneously released with all tasks in $\mathcal{A}_\alpha$ with a higher priority than $\tau_{\alpha,i}$, all those task with a higher priority experience a maximum release delay at that simultaneous release and a minimum release delay at subsequent releases, and that simultaneous release coincides with the completion of budget β_α. Moreover, we can easily convert Equations (15) and (16) by adding the terms $WI^\beta_\alpha(x)$ and $BI^\beta_\alpha(x)$, next to adding subscripts and superscripts, as illustrated below.

The *worst-case response time* $WR^\tau_{\alpha,i}$ of task $\tau_{\alpha,i}$ with activation jitter $AJ^\tau_{\alpha,i}$ of application $\mathcal{A}_\alpha$ is given by the smallest value $x \in \mathbb{R}^+$ satisfying

$$x = WC^\tau_{\alpha,i} + WI^\beta_\alpha(x) + \sum_{1\le j<i} \left\lceil \frac{x + AJ^\tau_{\alpha,j}}{T^\tau_{\alpha,j}} \right\rceil WC^\tau_{\alpha,j}, \tag{26}$$

where $WI^\beta_\alpha(x)$ is given by Equation (21). Similarly, the *best-case response time* $BR^\tau_{\alpha,i}$ of task $\tau_{\alpha,i}$ of application $\mathcal{A}_\alpha$ is given by the largest value $x \in \mathbb{R}^+$ satisfying

$$x = BC^\tau_{\alpha,i} + BI^\beta_\alpha(x) + \sum_{1\le j<i} \left(\left\lceil \frac{x - AJ^\tau_{\alpha,j}}{T^\tau_{\alpha,j}} \right\rceil - 1 \right)^+ BC^\tau_{\alpha,j}, \tag{27}$$

where $BI^{\beta}_{\alpha}(x)$ is given by Equation (25). Finally, the worst-case (absolute) finalization jitter is now bounded by

$$FJ^{\tau}_{\alpha,i} \leq AJ^{\tau}_{\alpha,i} + WR^{\tau}_{\alpha,i} - BR^{\tau}_{\alpha,i}. \tag{28}$$

Note that we constructed our examples such that the characteristics of task $\tau_{\alpha,i}$ of application $\mathcal{A}'_{\alpha}$ and of task τ_{i+1} of task set $\mathcal{T}_1$ are identical. The timelines for task set $\mathcal{T}_1$ in Figure 1 are therefore the same as the timelines for application $\mathcal{A}'_{\alpha}$ in Figures 7(b) and 8(b). Similarly, the timelines for task set $\mathcal{T}'_1$ in Figure 2 apply for an application $\mathcal{A}''_{\alpha}$ with the same characteristics as $\mathcal{A}'_{\alpha}$, except that task $\tau_{\alpha,1}$ now has an activation jitter $AJ^{\tau}_{\alpha,1} = 1$. The finalization jitter of task $\tau_{\alpha,2}$ in $\mathcal{A}''_{\alpha}$ is therefore also bounded by $FJ^{\tau}_{\alpha,2} \leq 0 + 9 - 4 = 5$.

7 Existing analysis revisited and extended

In this section, we first revisit and extend the analysis of tasks as presented in [12, 17, 18]. Next, we describe how to convert analytic results for FPPS to two-level H-FPPS for [1]. Finally, we compare our analysis with the analysis in [9].

7.1 Analysis for class (a) revisited and extended

We now revisit worst-case response time analysis for class (a), as described in [12, 17, 18], and subsequently present best-case analysis and jitter analysis for that class.

7.1.1 Worst-case response time analysis

Figure 9(a) illustrates a timeline for β_{α} and $\mathcal{A}_{\alpha}$ with a critical instant for $\tau_{\alpha,2}$. From this figure, we draw the conclusion that the provision of the capacity C^{β}_{α} of budget β_{α} can be dealt with by viewing the unavailability of budget β_{α} as an artificial task $\tau_{\alpha,0}$, where task $\tau_{\alpha,0}$ has a period $T^{\tau}_{\alpha,0} = T^{\beta}_{\alpha}$, a computation time $C^{\tau}_{\alpha,0} = T^{\beta}_{\alpha} - C^{\beta}_{\alpha}$, and an activation jitter $AJ^{\tau}_{\alpha,0} = C^{\beta}_{\alpha}$, as illustrated in Figure 9(b).

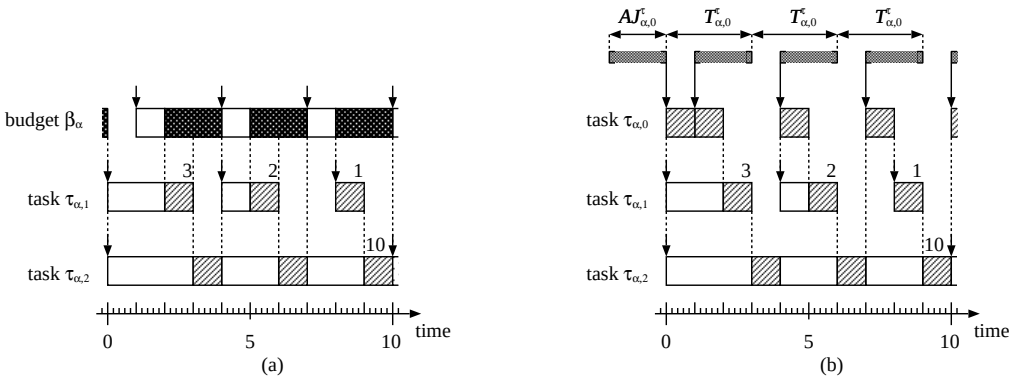

Figure 9. Timelines with a critical instant of task $\tau_{\alpha,2}$ for (a) budget β_{α} and application $\mathcal{A}_{\alpha}$ according to [12, 17, 18] and for (b) an extension $\mathcal{A}'_{\alpha}$ of $\mathcal{A}_{\alpha}$ with an artificial task $\tau_{\alpha,0}$.

The worst-case response time $WR^{\tau}_{\alpha,i}$ of $\tau_{\alpha,i}$ is therefore given by the smallest $x \in \mathbb{R}^{+}$ satisfying Equation (20), where $WI^{\beta}_{\alpha}(x)$ denotes the 'worst-case interference' of $\tau_{\alpha,0}$ or the 'worst-case unavailability' of C^{β}_{α}, i.e.

$$WI^{\beta}_{\alpha}(x) = \left\lceil \frac{x + C^{\beta}_{\alpha}}{T^{\beta}_{\alpha}} \right\rceil (T^{\beta}_{\alpha} - C^{\beta}_{\alpha}). \tag{29}$$

7.1.2 Best-case response time analysis

Similarly to the best-case response time analysis presented in Section 6.2, we *derive* an optimal instant for a task of an application $\mathcal{A}_{\alpha}$ with an associated budget β_{α}.

Figure 10(b) illustrates a timeline with an optimal instant for task $\tau_{\alpha,2}$ for the extension $\mathcal{A}'_\alpha$ of $\mathcal{A}_\alpha$ with the artificial task $\tau_{\alpha,0}$. The equivalent timeline with an optimal instant for task $\tau_{\alpha,2}$ for budget β_α and application $\mathcal{A}_\alpha$ is shown in Figure 8(a). The best-case response time $BR^\tau_{\alpha,i}$ of $\tau_{\alpha,i}$ is therefore given by the smallest $x \in \mathbb{R}^+$ satisfying Equation (24), where $BI^\beta_\alpha(x)$

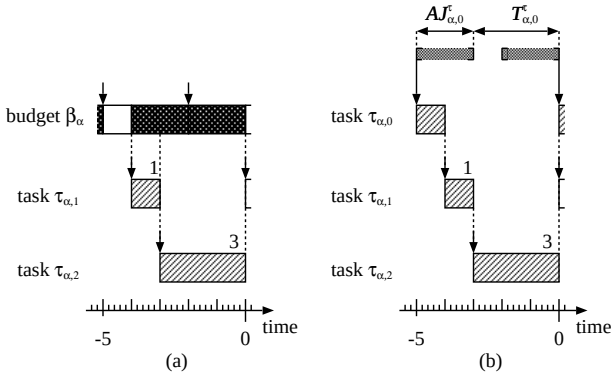

Figure 10. Timelines with an optimal instant of task $\tau_{\alpha,2}$ for (a) budget β_α and application $\mathcal{A}_\alpha$ according to [12, 17, 18] and for (b) an extension $\mathcal{A}'_\alpha$ of $\mathcal{A}_\alpha$ with an artificial task $\tau_{\alpha,0}$.

denotes the 'best-case interference' of task $\tau_{\alpha,0}$ or the 'best-case unavailability' of C^β_α, i.e.

$$BI^\beta_\alpha(x) = \left(\left\lceil \frac{x - C^\beta_\alpha}{T^\beta_\alpha} \right\rceil - 1 \right)^+ (T^\beta_\alpha - C^\beta_\alpha). \tag{30}$$

7.1.3 Jitter

Given the worst-case and best-case response time analysis for tasks for class (a), we can now simply reuse the jitter analysis described in Section 6.3 by applying Equations (29) and (30) for the terms $WI^\beta_\alpha(x)$ and $BI^\beta_\alpha(x)$ in the recursive equations (26) and (27) for the worst-case and best-case response time of task $\tau_{\alpha,i}$ in $\mathcal{A}_\alpha$.

7.2 Worst-case response time analysis for class (b) revisited

The unavailability of the capacity of budget β_α in [1] can be modeled by means of two artificial tasks $\tau_{\alpha,-1}$ and $\tau_{\alpha,0}$. If the so-called 'initial latency' in [1] is chosen to be equal to $WR^\beta_\alpha - C^\beta_\alpha$, then task $\tau_{\alpha,-1}$ is characterized by a period $T^\tau_{\alpha,-1} = T^\beta_\alpha$, a computation time $C^\tau_{\alpha,-1} = T^\beta_\alpha - WR^\beta_\alpha$, and a relative phasing $\varphi^\tau_{\alpha,-1} = WR^\beta_\alpha - C^\beta_\alpha$, and task $\tau_{\alpha,0}$ is characterized by $T^\tau_{\alpha,0} = T^\beta_\alpha$, $C^\tau_{\alpha,0} = WR^\beta_\alpha - C^\beta_\alpha$, and an activation jitter $AJ^\tau_{\alpha,0} = C^\beta_\alpha$. This is illustrated in Figure 11 for the worst-case assumption for available capacity analysis for budget β_3 of $\mathcal{B}_1$.

As described in Section 4, time zero in Figure 11(a) is a critical instant for a task $\tau_{3,i}$ of application $\mathcal{A}_3$ when $\tau_{3,i}$ is simultaneously released with all tasks in $\mathcal{A}_3$ with a higher priority than $\tau_{3,i}$. Under the same conditions, time zero in Figure 11(b) is a critical instant for task $\tau_{3,i}$. The worst-case response time $WR^\tau_{\alpha,i}$ of a task τ_α, i is therefore given by the smallest $x \in \mathbb{R}^+$ satisfying Equation (20), where $WI^\beta_\alpha(x)$ denotes the 'worst-case interference' of tasks $\tau_{\alpha,-1}$ and $\tau_{\alpha,0}$, i.e.

$$WI^\beta_\alpha(x) = \left(\left\lceil \frac{x - \varphi^\tau_{\alpha,-1}}{T^\tau_{\alpha,-1}} \right\rceil \right)^+ C^\tau_{\alpha,-1} + \left\lceil \frac{x + AJ^\tau_{\alpha,0}}{T^\tau_{\alpha,0}} \right\rceil C^\tau_{\alpha,0}. \tag{31}$$

In terms of the characteristics of the budget β_α, the 'worst-case unavailability' $WI^\beta_\alpha(x)$ of the capacity C^β_α can be written as

$$WI^\beta_\alpha(x) = \left(\left\lceil \frac{x - (WR^\beta_\alpha - C^\beta_\alpha)}{T^\beta_\alpha} \right\rceil \right)^+ (T^\beta_\alpha - WR^\beta_\alpha) + \left\lceil \frac{x + C^\beta_\alpha}{T^\beta_\alpha} \right\rceil (WR^\beta_\alpha - C^\beta_\alpha). \tag{32}$$

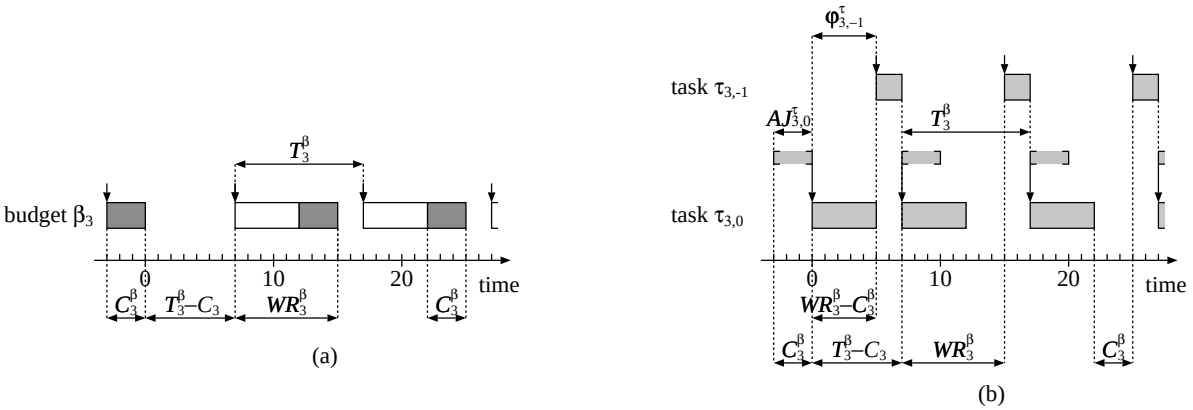

Figure 11. Timelines for (a) budget β_3 of $\mathcal{B}_1$ and (b) artificial tasks $\tau_{3,-1}$ and $\tau_{3,0}$ that model the unavailability of β_3.

task	$T^\tau = D^\tau$	C^τ	AJ^τ	φ^τ
$\tau_{\alpha,-1}$	T_α^β	$T_\alpha^\beta - \lambda - C_\alpha^\beta$	0	λ
$\tau_{\alpha,0}$	T_α^β	λ	C_α^β	0

Table 3. Characteristics of tasks $\tau_{\alpha,-1}$ and $\tau_{\alpha,0}$ assuming an initial latency λ.

In this section, we assumed an 'initial latency' λ equal to $WR_\alpha^\beta - C_\alpha^\beta$. In general, however, $\lambda \in [0, T_\alpha^\beta - C_\alpha^\beta]$. We can therefore also express the characteristics of the artificial tasks $\tau_{\alpha,-1}$ and $\tau_{\alpha,0}$ using λ, as illustrated in Table 3. This generalization of the analysis for class (b) unifies the analysis for classes (a) and (e), i.e. the analysis specializes to the analysis for class (a) for $\lambda = T_\alpha^\beta - C_\alpha^\beta$, and it specializes to (e) for $\lambda = 0$.

7.3 A comparison with the worst-case response time analysis for class (c)

The worst-case response time analysis for our basic approach is basically the same as the analysis of (unbound) tasks of an application associated with the highest priority budget in [9]. Using our notation, the worst-case response time $WR_{1,i}^\tau$ of a task $\tau_{1,i}$ in [9] is given by $WR_{1,i}^\tau = x + (T_1^\beta - C_1^\beta)$, where x is the smallest non-negative value satisfying

$$x = L_1(x) + \left\lceil \frac{L_1(x)}{C_1^\beta} \right\rceil (T_1^\beta - C_1^\beta),$$

and $L_1(x)$ is given by

$$L_1(x) = C_{1,i}^\tau + \sum_{1 \le j < i} \left\lceil \frac{x + (T_1^\beta - C_1^\beta)}{T_{1,j}^\tau} \right\rceil C_{1,j}^\tau.$$

Comparing these equations with Equations (20) and (21) for our basic approach, we conclude that it is not obvious that both analyses actually yield the same result. Moreover, our analysis cannot simply be viewed as a specialization of the analysis in [9] either.

8 Towards advanced approaches for budget scheduling

In this section, we consider the impact of selecting harmonic periods for budgets. For illustration purposes, we use an example budget set $\mathcal{B}_2$, with characteristics as given in Table 4. Note that $\mathcal{B}_2$ has a utilization equal to 1. Similarly to the basic approach, the schedulability condition for such an advanced approach is trivial, i.e. a simple test if the cumulative utilization is at most equal to 1. The capacities of the budgets also become available strictly periodically. Unfortunately, the other advantages of the basic approach do not necessarily hold for the advanced approach.

budget	$T = D$	C
β_1	7	2
β_2	14	3
β_3	14	3
β_4	14	4

Table 4. Budget characteristics of $\mathcal{B}_2$.

Figure 12 shows a timeline with the provision of the capacity of budget β_3, assuming a simultaneous release of all budgets of $\mathcal{B}_2$ at time zero. As illustrated in the figure, the provision of the capacity of β_3 is pre-empted by budget β_1. Moreover, it is no longer straightforward to determine the worst-case available capacity, and we therefore have to apply a cognac-glass algorithm. Figure 13 shows that a worst-case available capacity is assumed for an interval of length 11 when that interval starts at time $t_s = 7$. However, for an interval of length 18, a start at time $t_s = 10$ is required. For this example, it also suffices to consider only these two starting points.

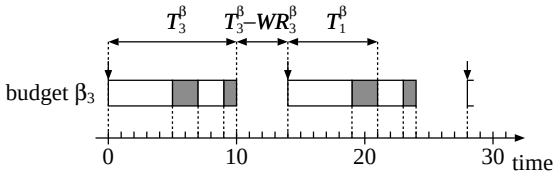

Figure 12. Assumptions for available capacity of budget β_3 of $\mathcal{B}_2$.

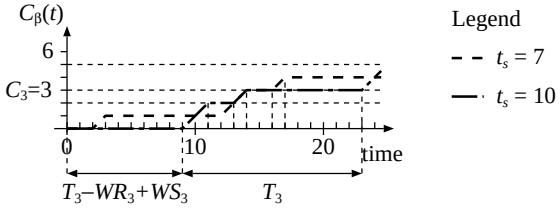

Figure 13. Construction of worst-case available capacity of budget β_3 of $\mathcal{B}_2$.

Hence, to determine the worst-case response times of tasks for this advanced approach for budget scheduling, we have to extend the analysis for the basic approach in Section 6 in two ways. Firstly, we have to add an additional artificial task for every additional pre-emption of the provision of the capacity. Secondly, we have to apply a cognac-glass algorithm.

We will also illustrate these extensions by means of our example. Similarly to the basic approach, we have an artificial task $\tau_{3,0}$ with a period $T^\tau_{3,0} = T^\beta_3 = 14$ and a computation time $C^\tau_{3,0} = T^\beta_3 - WR^\beta_3 + WS^\beta_3 = 14 - 10 + 5 = 9$. Because the provision of the capacity has one additional pre-emption, we need one additional artificial task $\tau_{3,-1}$ with a period $T^\tau_{3,-1} = T^\beta_3 = 14$ and a computation time equal to the duration of the pre-emption of β_3 by budget β_1, i.e. $C^\tau_{3,-1} = C^\beta_1 = 2$. This is illustrated in Figure 14. Note that task $\tau_{3,-1}$ has a phasing $\varphi^\tau_{3,-1}$ equal to $T^\beta_3 - WR^\beta_3 + T^\beta_1 = 14 - 10 + 7 = 11$ relative to task $\tau_{3,0}$. Conversely, we can state that $\tau_{3,0}$ has a phasing $\varphi^\tau_{3,0}$ equal to $WR^\beta_3 - T^\beta_1$ relative to task $\tau_{3,-1}$. To determine the worst-case response time of a task $\tau_{3,i}$ of the application $\mathcal{A}_3$ associated with β_3, we need to consider two potential critical instants corresponding with the two starting points $t_s = 7$ and $t_s = 10$ mentioned above. To determine the response time for these two instances, we can use Equation (20), where $\alpha = 3$ and $WI^\beta_3(x)$ is given by

$$WI^\beta_3(x) = \left\lceil \frac{x - \varphi^\tau_{3,-1}}{T^\tau_3} \right\rceil C^\tau_{3,-1} + \left\lceil \frac{x - \varphi^\tau_{3,0}}{T^\tau_3} \right\rceil C^\tau_{3,0}.$$

For the instant with starting point $t_s = 7$, we use $\varphi^\tau_{3,-1} = 0$ and $\varphi^\tau_{3,0} = WR^\beta_3 - T^\beta_1 = 3$. Similarly, we use $\varphi^\tau_{3,-1} = T^\beta_3 - WR^\beta_3 + T^\beta_1 = 11$ and $\varphi^\tau_{3,0} = 0$ for $t_s = 10$. The worst-case response time of task $\tau_{3,i}$ is given by the minimum of the response times of task $\tau_{3,i}$ for these two instances.

We observe that although the analysis becomes more complicated due to these extensions, it remains exact.

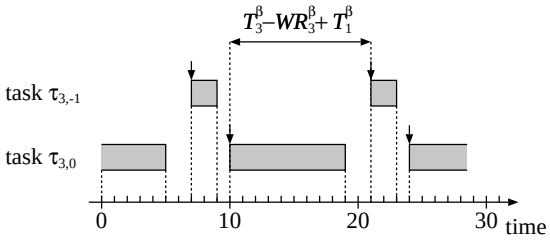

Figure 14. Timeline with artificial tasks $\tau_{3,0}$ and $\tau_{3,-1}$ that model the unavailability of β_3.

9 Conclusions

In this document, we considered two-level scheduling of independent applications in cost-constrained systems, using FPPS for tasks. We focused on pragmatic solutions for such systems, allowing efficient implementations of budgets and simple analysis of tasks. We first briefly reviewed existing approaches of two-level H-FPPS, in which we compared their worst-case available capacity. We subsequently considered a basic approach, where all budgets have the same period and each budget has a specific phasing, and showed that the worst-case available capacity for this approach improves on results of existing approaches. Next, we presented analysis for tasks, and showed that existing analytic results for real-time tasks under FPPS can be easily converted to H-FPPS by viewing the *unavailability* of a budget as an artificial highest priority task. As a result, both the worst-case and best-case response time analysis of independent tasks with arbitrary phasing is *exact* for this basic approach. The conversion was illustrated by means of jitter analysis. We conclude that this view therefore significantly simplifies the analysis of tasks for two-level hierarchical scheduling.

We showed that the conversion equally well applies to the approaches described in [12, 17, 18], and sketched how it applies to [1]. We briefly discussed how our analytical results compare to the results in [9], and concluded that although the analysis for the highest priority budget is basically the same, our analysis cannot be simply viewed as a specialization of that analysis. Finally, we considered an advanced approach for budget scheduling, with harmonic periods for budgets, and showed by means of an example that we may have to apply a cognac-glass algorithm during analysis.

In this document, we assumed independent tasks. Assuming independent applications, our results can easily be extended to cover dependent tasks. Given the analytic legacy for FPPS, the identified possibility to easily convert existing analytic results for real-time tasks under FPPS to two-level hierarchical scheduling is considered the main contribution of this document. Further work is required for dependent applications and advanced approaches for budget scheduling.

Acknowledgements

We thank Pieter J.L. Cuijpers and Mike Holenderski for discussions and their comment on an earlier version of this document.

References

[1] L. Almeida and P. Peidreiras. Scheduling with temporal partitions: response-time analysis and server design. In *Proc. of the 4th ACM International Conference on Embedded Software (EMSOFT)*, pp. 95 – 103, September 2004.

[2] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[3] R. Bril. *Real-time scheduling for media processing using conditionally guaranteed budgets*. PhD thesis, Technische Universiteit Eindhoven (TU/e), The Netherlands, July 2004. http://alexandria.tue.nl/extra2/200412419.pdf.

[4] R. Bril and P. Cuijpers. Analysis of hierarchical fixed-priority pre-emptive scheduling revisited. Technical Report CS Report 06-36, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, December 2006.

[5] R. Bril, E. Steffens, and W. Verhaegh. Best-case response times and jitter analysis of real-time tasks. *Journal of Scheduling*, 7(2):133–147, March 2004.

[6] R. Bril, W. Verhaegh, and C. Wüst. A cognac-glass algorithm for conditionally guaranteed budgets. In *Proc. 27th IEEE Real-Time Systems Symposium (RTSS)*, pp. 388–397, December 2006.

[7] G. Buttazzo. *Hard real-time computing systems - predictable scheduling algorithms and applications (2nd edition)*. Springer, 2005.

[8] G. Buttazzo and A. Cervin. Comparative assessment and evaluation of jitter control methods. In *Proc. 15th International Conference on Real-Time and Network Systems (RTNS)*, pp. 163–172, March 2007.

[9] R. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proc.* 26^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pp. 389–398, December 2005.
[10] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.
[11] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.
[12] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proc.* 15^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 151–158, July 2003.
[13] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.
[14] C. Mercer, R. Rajkumar, and J. Zelenka. Temporal protection in real-time operating systems. In *Proc.* 11^{th} *IEEE Workshop on Real-Time Operating Systems and Software (RTOSS)*, pp. 79–83, May 1994.
[15] A. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. In *Proc.* 7^{th} *Real-Time Technology and Applications Symposium (RTAS)*, pp. 75–84, May 2001.
[16] O. Redell and M. Sanfridson. Exact best-case response time analysis of fixed priority scheduled tasks. In *Proc.* 14^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 165–172, June 2002.
[17] S. Saewong, R. Rajkumar, J. Lehoczky, and M. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proc.* 14^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 152–160, June 2002.
[18] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proc.* 24^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pp. 2–13, December 2003.