Exact analysis for basic cyclic executives

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Exact analysis for basic cyclic executives

Reinder J. Bril
Eindhoven University of Technology, Netherlands

20/02

ISSN 0926-4515

editor: prof.dr.ir. J.J. van Wijk

Reports are available at:
https://research.tue.nl/en/publications/?search=Computer+science+reports+eindhoven&originalSearch=Computer+science+reports+eindhoven&pageSize=50&ordering=publicationYearThenTitle&descending=true&showAdvanced=false&allConcepts=true&inferConcepts=true&searchBy=RelatedConcepts

Computer Science Reports 20-02
Eindhoven, November 2020

Exact analysis for basic cyclic executives

Reinder J. Bril
Technische Universiteit Eindhoven (TU/e), Eindhoven,
The Netherlands
Email: r.j.bril@tue.nl

Version 1.0 of November 2, 2020

Abstract

The academic interest in cyclic executives for multitasking in hard real-time systems decreased significantly during the past decades. Moreover, cyclic executives are hardly addressed in contemporary text books, if at all. Cyclic executives are still in use, however, and there is therefore a need for analysis techniques for these executives.

In this document, we present exact analysis for basic cyclic executives scheduling a given sequence of independent hard real-time polling tasks in single-processor systems. Unlike existing approaches, which typically take periodic tasks as a starting point for cyclic executives and focus on schedulability of *tasks*, we take the schedulability of the *system* as a starting point, i.e. whether or not the system meets its deadlines. In particular, we do not assume periods and deadlines for tasks.

Contents

1 Introduction

1.1 Background and motivation

As described in [1], cyclic executives are generally used in defense military systems and traffic control systems to handle periodic tasks. Cyclic executives effectively execute hard real-time tasks in a continuous loop, in which every task appears at least once. Since the paper by C. Douglass Locke [2], in which it is concluded that the fixed-priority approach generally dominates the cyclic executive approach for hard real-time systems, the academic interest in cyclic executives has gradually decreased[1]. Accordingly, cyclic executives are mentioned in contemporary text books, such as [4, 5, 1, 6], but hardly addressed in detail. Despite the decreasing academic interest, cyclic executives are still in use, however. In a recent survey of 120 industry practitioners in the field of real-time embedded systems [7], it was observed that from the 97 respondents to the question *which task scheduling policy/policies are used in the considered systems*, 54.17% answered "static cycle/table driven/time-triggered". Availability of exact analysis for cyclic executives therefore remains desirable. This document is a treatise on exact analysis for basic cyclic executives scheduling independent, hard real-time polling tasks in single-processor systems.

1.2 Problem description

Hard real-time systems require timely responses to events. As illustrated by an air bag, a system response (the inflation of an airbag) shall be provided in a well-defined interval relative to the occurrence of an event (a collision), i.e. neither *too early* nor *too late*. A system specification may therefore constrain the time-interval from an event to a response by both a lower bound, a so-called *best-case system deadline*, as well as an upper bound, a so-called *worst-case system deadline*. For hard real-time computing systems, where control is realized in software, these system deadlines give rise to timing constraints on the real-time tasks that have to detect events (through regularly polling sensors) and provide responses (using actuators).

In this document, we assume a hard real-time system with a set of n (input) events, a set of n (output) responses, and a 1-1 relationship between events and responses. Events are sporadic, i.e. are characterized by a minimal inter-arrival time. Each event-response tuple is characterized by a best-case system deadline and a worst-case system deadline, where the worst-case system deadline is at most equal to the minimal inter-arrival time of the event. Each event-response tuple is handled by a dedicated real-time polling task. A task detects an event through *observation*, e.g. by reading sensory data, and provides a response through a *command*, e.g. using an actuator; see Figure 1.

We assume a single-processor system and a basic cyclic executive to schedule the set of tasks. In particular, we assume a statically known sequence of tasks executed in a continuous loop. Depending on the type of cyclic executive, a task may occur exactly once or at least once in the sequence. Each task is characterized by a lower bound and an upper bound on its computation time, i.e. by a *best-case computation time* and a *worst-case computation time*. Although

[1] In a paper from 1994 by Alan Burns [3] it is explicitly stated that "*Most existing hard real-time systems are implemented using a static table driven schedule (often called a* cyclic executive*)*".

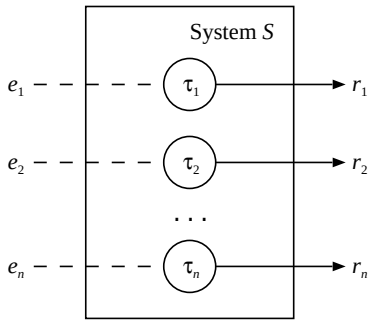

Figure 1: A system $\mathcal{S}$ with n (input) events $e_1, e_2, \ldots, e_n$, n (output) responses $r_1, r_2, \ldots, r_n$, and n polling tasks $\tau_1, \tau_2, \ldots, \tau_n$ detecting events through observation (as indicated by the dashed lines) and providing responses through commands (as indicated by the arrows).

tasks scheduled by means of a cyclic executive are typically assumed to be periodic [8], i.e. are typically assumed to be activated with a fixed inter-arrival time (or *period*), we do not make that assumption in this document. The problem addressed in this document is how to determine the schedulability of the system, i.e. whether or not the best-case and worst-case deadlines of the system are met. Part of the problem concerns the derivation of timing constraints for the tasks from the deadlines of the system.

1.3 Goal

In this document, we present exact analysis for six basic cyclic executives. These six cyclic executives are classified based on two orthogonal dimensions. The first dimension determines the number of times a task may be executed in a cycle, e.g. either exactly once (single-rate) or at least once (multi-rate). The second dimension determines the start times of tasks, i.e. (*i*) tasks are executed immediately one after the other and as fast as possible (AFAP), (*ii*) the cycle is started by a timer interrupt and tasks are executed AFAP (time-driven AFAP), or (*iii*) every job of a task is started at a statically known time (periodic). The selection of these basic cyclic executives has been inspired by [9]. We assume that the sequence of jobs to be executed in a cycle is known.

1.4 Topics addressed

This document addresses the following four main topics. Firstly, it presents exact analysis for basic cyclic executives. Secondly, it compares the various cyclic executives presented with respect to schedulability of systems. Thirdly, it provides simple means to determine bounds on the cycle time of time-driven AFAP and periodic cyclic executives guaranteeing schedulability of a system with a given execution sequence of tasks in a cycle. Finally, it presents simple means to determine bounds on the fraction of time that can be gained for background processing by time-driven AFAP and periodic cyclic executive compared to a AFAP cyclic executive.

1.5 Organization

The remainder of this document is organized as follows. In Section 2, we present a basic real-time system model. The problem is formalized in Section 3. Exact best-case analysis, which turns out to be trivial for cyclic executives, is also addressed in this section. In Section 4, we provide a recap of basic single-rate and multi-rate cyclic executives. Exact worst-case analysis of these basic single-rate and multi-rate cyclic executives are the topic of Section 5 and Section 6, respectively. We briefly discuss various related topics in Section 7 and subsequently conclude the document in Section 8.

2 A basic real-time system model

We assume a system $\mathcal{S}$, a set $\mathcal{E}^{\mathcal{S}}$ of n sporadic (input) events $e_1, e_2, \ldots, e_n$, a set $\mathcal{R}^{\mathcal{S}}$ of n (output) responses $r_1, r_2, \ldots, r_n$ to these events, and a 1-1 relationship between events and responses. Each event e_i is characterized by a minimal inter-arrival time $WT_i^{\mathcal{S}}$. Each event-response tuple $< e_i, r_i >$ is characterized by a best-case system deadline $BD_i^{\mathcal{S}}$ and a worst-case system deadline $WD_i^{\mathcal{S}}$, where $0 \leq BD_i^{\mathcal{S}} \leq WD_i^{\mathcal{S}} \leq WT_i^{\mathcal{S}}$.

Moreover, we assume a set $\mathcal{T}^{\mathcal{S}}$ of n independent polling tasks $\tau_1, \tau_2, \ldots, \tau_n$, where task τ_i senses event e_i and provides response r_i. Each task τ_i is characterized by a best-case computation time BC_i and a worst-case computation time WC_i, where $0 < BC_i \leq WC_i$. The execution of a task is termed a *job*. A job of task τ_i is denoted by $\iota_{i,k}$, with $0 \leq k$, i.e. the first job of τ_i is denoted by $\iota_{i,0}$. The computation time $C_{i,k}$ of every job $\iota_{i,k}$ of task τ_i shall be between BC_i and WC_i, i.e.

$$\underset{1 \leq i \leq n \ \wedge \ 0 \leq k}{\forall} BC_i \leq C_{i,k} \leq WC_i.$$

We use $a_{i,k}$, $s_{i,k}$ and $f_{i,k}$ to denote the *activation time*, *start time* and the *finalization time* of job $\iota_{i,k}$, respectively. The response-time $R_{i,k}$ of job $\iota_{i,k}$ of task τ_i is now defined as

$$R_{i,k} \stackrel{\text{def}}{=} f_{i,k} - a_{i,k}. \tag{1}$$

The best-case response time BR_i and worst-case response time WR_i of τ_i are subsequently defined as

$$BR_i \stackrel{\text{def}}{=} \inf_{0 \leq k} R_{i,k}, \tag{2}$$

and

$$WR_i \stackrel{\text{def}}{=} \sup_{0 \leq k} R_{i,k}, \tag{3}$$

respectively. The *start jitter* SJ_i of a task τ_i is defined as the maximum deviation of the start-times of a task from a strictly periodic pattern, i.e.

$$SJ_i = \sup_{0 \leq k \ \wedge \ 0 \leq \ell} \left((s_{i,k} - k \times T_i) - (s_{i,\ell} - \ell \times T_i) \right), \tag{4}$$

where T_i denotes the (assumed) period of the task.

We assume that tasks are scheduled by cyclic executives, run non-preemptively and do not suspend themselves. The sequence of tasks executed in a cycle is assumed to be known statically. Moreover, we ignore scheduling overhead and system initialization. When a system $\mathcal{S}$ meets all its deadlines under a spe-

cific cyclic executive and a specific sequence of tasks, the system $\mathcal{S}$ is called *schedulable* under that cyclic executive and that sequence.

For single-rate cyclic executives we assume, for ease of presentation and without loss of generality, that the first jobs of tasks in a cycle are executed in the order of increasing index, i.e.

$$\underset{1 \le i < j \le n \ \wedge \ 0 \le k}{\forall} s_{i,k} < s_{j,k}. \tag{5}$$

where m_i denotes the number of times task τ_i is executed in a cycle.

For multi-rate cyclic executives, we use $m_i \geq 1$ to denote the number of times task τ_i is executed in a cycle. For ease of presentation, we assume a sequence `int task[]` of length $N = \sum_{1 \le i \le n} m_i$ representing the cycle for multi-rate cyclic executives, where the index ranges from 0 to $N-1$, and a function `int index( job )` yielding the index in the sequence of the job `job` past as an argument, where $\texttt{index}(\iota_{i,k}) = \texttt{index}(\iota_{i,(k \mod m_i)})$ for $1 \le i \le n$ and $0 \le k$. The relation between the sequence `int task[]` and function `int index( job )` is expressed by

$$\underset{1 \le i \le n}{\forall} \underset{0 \le k < m_i}{\forall} \texttt{task}[\texttt{index}(\iota_{i,k})] = i$$

and

$$\underset{0 \le j \le N-1}{\forall} \underset{0 \le k < m_{\texttt{task}[j]}}{\exists!} \texttt{index}(\iota_{\texttt{task}[j],k}) = j.$$

An overview of the notations used in this document can be found in Table 1.

3 Problem formalization

As described in Section 1.2, the response r_i of a hard real-time system $\mathcal{S}$ to the occurrence of an event e_i at time t_e shall be provided in a well-defined time-interval, bounded by a best-case system deadline $BD_i^{\mathcal{S}}$ and a worst-case system deadline $WD_i^{\mathcal{S}}$ relative to the occurrence of the event; see Figure 2. When the response r_i is given *before* time $t_e + BD_i^{\mathcal{S}}$, it is *too early*, and when it is given *after* time $t_e + WD_i^{\mathcal{S}}$, it is *too late*.

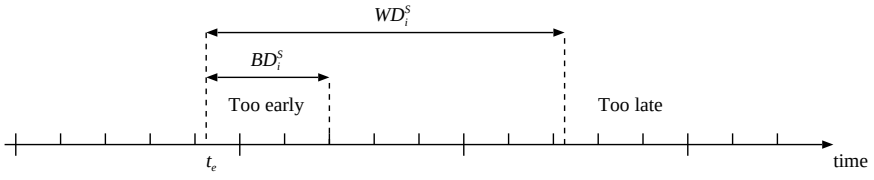

Figure 2: An event e_i for a system $\mathcal{S}$ occurs at time t_e. The system shall provide a response r_i within a well-defined time-interval $[t_s + BD_i^{\mathcal{S}}, t_s + WD_i^{\mathcal{S}}]$.

We assume a real-time task τ_i that regularly polls for the event e_i and provides the response r_i upon the occurrence of the event. In this section, we derive timing constraints for task τ_i for two cases. The first case, addressed in Section 3.1, recapitulates the existing approach for strictly periodic tasks, where a period T_i and a *best-case task deadline* BD_i and a *worst-case task deadline*

Table 1: An overview of the notations used.

System	
$\mathcal{S}$	system
$\mathcal{E}^{\mathcal{S}}$	set of n sporadic (input) events $e_1, e_2, \ldots, e_n$ of $\mathcal{S}$
$\mathcal{R}^{\mathcal{S}}$	set of n (output) responses $r_1, r_2, \ldots, r_n$ of $\mathcal{S}$
e_i	(input) event i of system $\mathcal{S}$
r_i	(output) response i of system $\mathcal{S}$
$WT_i^{\mathcal{S}}$	minimal inter-arrival time of the evente_i
$BD_i^{\mathcal{S}}$	best-case system deadline of the event-response tuple $< e_i, r_i >$
$WD_i^{\mathcal{S}}$	worst-case system deadline of the event-response tuple $< e_i, r_i >$
Tasks	
$\mathcal{T}^{\mathcal{S}}$	set of n independent polling tasks $\tau_1, \tau_2, \ldots, \tau_n$ of $\mathcal{S}$
τ_i	task identified by i
$\iota_{i,k}$	job k of task τ_i
BC_i	best-case computation time of task τ_i
WC_i	worst-case computation time of task τ_i
T_i	period of task τ_i
BR_i	worst-case response time of task τ_i
WR_i	worst-case response time of task τ_i
WD_i	worst-case task deadline of task τ_i
SJ_i	start jitter of task τ_i
$C_{i,k}$	computation time of job $\iota_{i,k}$
$R_{i,k}$	response time of job $\iota_{i,k}$
$a_{i,k}$	activation time of job $\iota_{i,k}$
$s_{i,k}$	start time of job $\iota_{i,k}$
$f_{i,k}$	finalization time of job $\iota_{i,k}$
Scheduling	
m_i	number of times task τ_i is executed in a cycle
Φ	start time of the first job of the first cycle
$T^{\mathcal{S}}$	cycle time of the cyclic executive of system $\mathcal{S}$
Auxiliaries	
`int task[]`	sequence of length $N = \sum_{1 \le i \le n} m_i$ representing a cycle
`int index( job )`	function yielding the index in the sequence of job `job`

WD_i are derived from the system deadlines $BD_i^{\mathcal{S}}$ and $WD_i^{\mathcal{S}}$. The second case, addressed in Section 3.2, introduces our alternative approach. We conclude this section with a problem formulation.

3.1 Strictly periodic polling tasks - a recap

In this section, we derive timing constraints for strictly periodic polling tasks from the system deadlines and formulate both worst-case and best-case schedulability conditions for tasks as well as the system.

3.1.1 A worst-case schedulability condition

For a strictly periodic polling task τ_i, a worst-case situation arises when (*i*) the event occurs immediately *after* a job $\iota_{i,k}$ of τ_i polls for the event, (*ii*) job $\iota_{i,k}$ immediately starts upon its activation, (*iii*) the next job $\iota_{i,k+1}$ experiences a worst-case interference of other tasks, and (*iv*) job $\iota_{i,k+1}$ requires its worst-case computation time for execution; see Figure 3. For such a worst-case situation, $\iota_{i,k+1}$ must finish before the worst-case system deadline $WD_i^{\mathcal{S}}$.

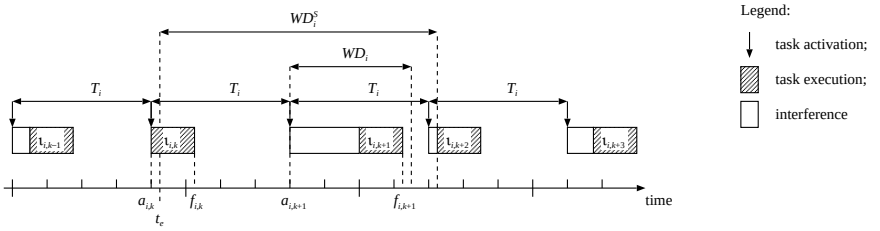

Figure 3: A worst-case situation for a strictly periodic polling task τ_i to handle an event e_i for a system $\mathcal{S}$ that occurs at time t_e, where job $\iota_{i,k+1}$ assumes the worst-case response time WR_i of τ_i.

Based on Figure 3, we therefore derive the following timing constraint for task τ_i based on activation times and finalization times

$$\underset{0\leq\ell}{\forall} f_{i,\ell+1} - a_{i,\ell} \leq WD_i^{\mathcal{S}}. \tag{6}$$

For a strictly periodic polling task with period T_i, this can be rewritten to

$$\begin{aligned} & \underset{0\leq\ell}{\forall} f_{i,\ell+1} - a_{i,\ell} \leq WD_i^{\mathcal{S}} \\ \Leftrightarrow \quad & \{(1)\} \underset{0\leq\ell}{\forall} f_{i,\ell+1} - a_{i,\ell+1} + T_i \leq WD_i^{\mathcal{S}} \\ \Leftrightarrow \quad & \underset{0\leq\ell}{\forall} R_{i,\ell+1} + T_i \leq WD_i^{\mathcal{S}} \end{aligned}$$

Hence, by using Equation (3), ignoring system initialization, and choosing the period T_i and worst-case task deadline WD_i of task τ_i such that (see also Figure 3)

$$T_i + WD_i \leq WD_i^{\mathcal{S}} \tag{7}$$

we arrive at the classical condition for worst-case schedulability of task τ_i

$$WR_i \leq WD_i,$$

irrespective of the scheduling algorithm. The worst-case deadlines of a system $\mathcal{S}$ are therefore met when

$$\underset{1\leq i\leq n}{\forall} T_i + WD_i \leq WD_i^{\mathcal{S}} \wedge WR_i \leq WD_i. \tag{8}$$

As long as the period T_i and deadline WD_i of the task τ_i satisfy this constraint, and τ_i is guaranteed to meet its deadline WD_i, the system deadline $WD^{\mathcal{S}}$ is met as well. Observe that no specific values for the period and deadline

are required, i.e. many tuples satisfy Equation (8). A specific selection of values is to decide upon a so-called *implicit* deadline, where the period and worst-case task deadline are equal, i.e. $T_i = WD_i$. In such a case, Equation (8) simplifies to

$$\underset{1 \leq i \leq n}{\forall} 2 \times T_i \leq WD_i^{\mathcal{S}} \wedge WR_i \leq T_i.$$

3.1.2 A best-case schedulability condition

For a strictly periodic polling task τ_i, a best-case situation arises when (*i*) the event occurs immediately *before* a job $\iota_{i,k}$ of τ_i polls for the event, (*ii*) job $\iota_{i,k}$ immediately starts upon its activation, (*iii*) job $\iota_{i,k}$ experiences a best-case interference of other tasks, and (*iv*) job $\iota_{i,k}$ requires its best-case computation time for execution; see Figure 4.

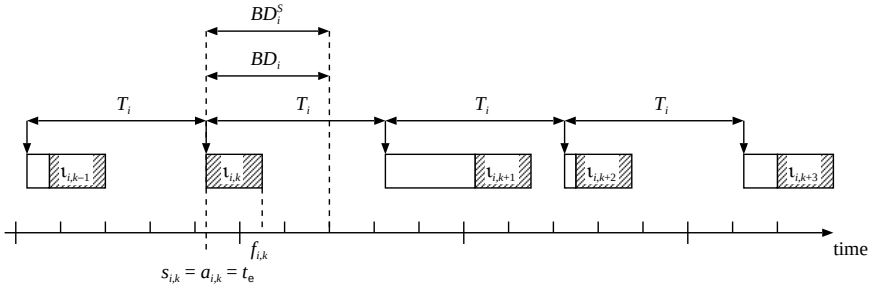

Figure 4: A best-case situation for a strictly periodic polling task τ_i to handle an event e_i for a system $\mathcal{S}$ that occurs at time t_e, where job $\iota_{i,k}$ assumes the best-case response time BR_i of τ_i.

Based on Figure 4, we therefore derive the following timing constraint for task τ_i based on activation times and finalization times

$$\underset{0 \leq \ell}{\forall} f_{i,\ell} - a_{i,\ell} \geq BD_i^{\mathcal{S}}.$$

Using Equations (1) and (2), this can be rewritten to

$$\begin{aligned} & \underset{0 \leq \ell}{\forall} R_{i,\ell} \geq BD_i^{\mathcal{S}} \\ \Leftrightarrow \quad & BR_i \geq BD_i^{\mathcal{S}}. \end{aligned}$$

By choosing the best-case task deadline BD_i of task τ_i such that (see also Figure 4)

$$BD_i \geq BD_i^{\mathcal{S}} \tag{9}$$

we arrive at the classical condition for best-case schedulability of task τ_i

$$BR_i \geq BD_i.$$

For cyclic executives, where tasks are executed non-preemptively, the best-case response time is the same as the best-case computation time, similar to fixed-priority non-preemptive scheduling with arbitrary phasing [10]. Hence, the best-

case schedulability condition for task τ_i now becomes

$$BC_i \geq BD_i.$$

The best-case deadlines of a system $\mathcal{S}$ are therefore met when

$$\underset{1 \leq i \leq n}{\forall} BC_i \geq BD_i \geq BD_i^{\mathcal{S}}. \tag{10}$$

This best-case schedulability condition is independent of a specific cyclic executive.

3.1.3 Concluding remarks

From the previous subsections, we conclude that a system $\mathcal{S}$, based on (*i*) strictly periodic polling tasks to handle events and (*ii*) cyclic executives executing tasks non-preemptively, meets its deadlines when the following schedulability condition holds

$$\underset{1 \leq i \leq n}{\forall} \left(\left(T_i + WD_i \leq WD_i^{\mathcal{S}} \wedge WR_i \leq WD_i\right) \wedge \left(BC_i \geq BD_i \geq BD_i^{\mathcal{S}}\right) \right). \tag{11}$$

How to check the condition falls outside the scope of this document.

We merely observe that the so-called *minor cycle* and *major cycle*, which are typically used to describe a cyclic executive [1], are generally *derived* from the periods of the tasks. In particular, the *minor cycle* and the *major cycle* are generally taken to be the Greatest Common Divider (GCD) and the Least Common Multiple (LCM) of the periods of the tasks, respectively.

3.2 A "regularly" polling task

Although it is common to assume periodic tasks for system scheduled by means of a cyclic executive, there is no inherent need to do so. In this section, we derive timing constraints for "regularly" polling tasks from the system deadlines and formulate both worst-case and best-case schedulability conditions for both the tasks and the system.

3.2.1 A worst-case schedulability condition

Similar to a strictly periodic polling task τ_i, a worst-case situation arises for a "regularly" poling task when (*i*) the event occurs immediately *after* a job $\iota_{i,k}$ of τ_i polls for the event, (*ii*) job $\iota_{i,k}$ immediately starts upon its activation, (*iii*) the next job $\iota_{i,k+1}$ experiences a worst-case interference of other tasks, and (*iv*) job $\iota_{i,k+1}$ requires its worst-case computation time for execution; see Figure 5. For such a worst-case situation, $\iota_{i,k+1}$ must finish before the worst-case system deadline $WD_i^{\mathcal{S}}$. Based on Figure 5, we therefore derive the following timing constraint for task τ_i based on start times and finalization times, similar to Equation (7)

$$\underset{0 \leq k}{\forall} f_{i,k+1} - s_{i,k} \leq WD_i^{\mathcal{S}}. \tag{12}$$

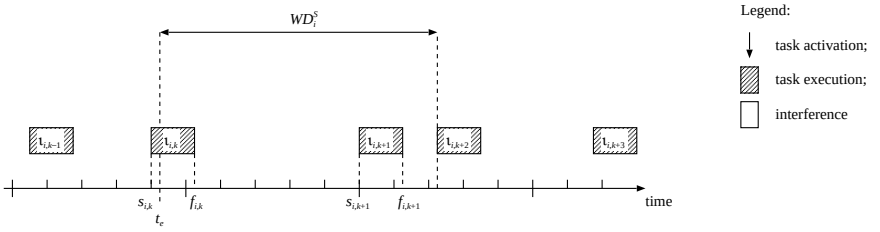

Figure 5: A worst-case situation for a regularly polling task τ_i to handle an event e_i for a system $\mathcal{S}$ that occurs at time t_e, where job $\iota_{i,k+1}$ executes for the worst-case computation time WC_i of τ_i.

The worst-case deadlines of a system $\mathcal{S}$ are therefore met when the following condition holds

$$\bigvee_{1 \leq i \leq n} \left(\bigvee_{0 \leq k} f_{i,k+1} - s_{i,k} \leq WD_i^{\mathcal{S}} \right). \tag{13}$$

3.2.2 A best-case schedulability condition

Similar to a strictly periodic polling task τ_i, a best-case situation arises for a "regularly" poling task when (*i*) the event occurs immediately *before* a job $\iota_{i,k}$ of τ_i polls for the event, (*ii*) job $\iota_{i,k}$ immediately starts upon its activation, (*iii*) job $\iota_{i,k}$ experiences a best-case interference of other tasks, and (*iv*) job $\iota_{i,k}$ requires its best-case computation time for execution; see Figure 6. Based on

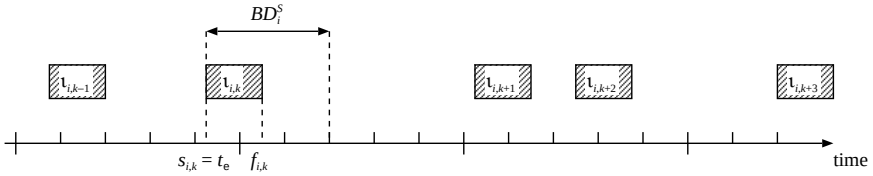

Figure 6: A best-case situation for a regularly polling task τ_i to handle an event e_i for a system $\mathcal{S}$ that occurs at time t_e, where job $\iota_{i,k+1}$ executes for the best-case computation time BC_i of τ_i.

Figure 6, we therefore derive the following timing constraint for task τ_i based on start times and finalization times

$$\bigvee_{0 \leq \ell} f_{i,\ell} - s_{i,\ell} \geq BD_i^{\mathcal{S}}.$$

For cyclic executives, where tasks are executed non-preemptively, this can be rewritten to

$$\bigvee_{1 \leq i \leq n} BC_i \geq BD_i^{\mathcal{S}}. \tag{14}$$

This best-case schedulability condition is independent of a specific cyclic executive.

3.2.3 Concluding remarks

From the previous subsections, we conclude that a system $\mathcal{S}$, based on (*i*) "regular" polling tasks to handle events and (*ii*) cyclic executives executing tasks non-preemptively, meets its deadlines when the following schedulability condition holds

$$\underset{1 \leq i \leq n}{\forall} \left(\left(\underset{0 \leq k}{\forall} f_{i,k+1} - s_{i,k} \leq WD_i^{\mathcal{S}} \right) \wedge BC_i \geq BD_i^{\mathcal{S}} \right). \tag{15}$$

The best-case schedulability condition is independent of a specific cyclic executive and checking the condition is considered trivial. In the remainder of this document, we therefore focus on checking Equation (13) for various basic single-rate and multi-cyclic executives, respectively, where the sequence of tasks to be executed in a cycle is assumed to be known.

3.3 Problem formulation

As illustrated in this section, exact best-case analysis of a system $\mathcal{S}$, where scheduling is based on a cyclic executive with non-preemptive execution of tasks, is trivial. We therefore focus on exact worst-case analysis in the remainder of this document.

For a (worst-case) deadline $WD_i^{\mathcal{S}}$ of the system $\mathcal{S}$, related to event $e_i \in \mathcal{E}^{\mathcal{S}}$, we have to guarantee that task $\tau_i \in \mathcal{T}^{\mathcal{S}}$ provides a timely response $r_i \in \mathcal{R}^{\mathcal{S}}$. We therefore have to make sure that the length of the interval from the *start time* $s_{i,k}$ of a job $\iota_{i,k}$ till the *finalization time* $f_{i,k+1}$ of the next job $\iota_{i,k+1}$ is at most equal to $WD_i^{\mathcal{S}}$; see Equation (12). To determine *worst-case schedulability of the system* $\mathcal{S}$, we therefore have to prove Equation (13)

In Sections 5 and 6, we provide exact worst-case analysis for basic cyclic executives, under the assumptions given in Section 2. In the next section, basic cyclic executives are recapitulated.

4 A recap of basic cyclic executives

Inspired by [9], we consider six basic cyclic executives based on two orthogonal dimensions:

1. single-rate, i.e. $\underset{1 \leq i \leq n}{\forall} m_i = 1$, versus multi-rate, i.e. $\underset{1 \leq i \leq n}{\forall} m_i \geq 1$, and
2. AFAP (as fast as possible), time-driven AFAP, and periodic.

These six basic cyclic executives are briefly recapitulated below, using the first dimension as leading for the structure of the section.

For each of these executives, tasks can simply be implemented as procedures and executed within a loop.

4.1 Single-rate cyclic executive

A single-rate cyclic executive essentially executes every task exactly once in a cycle. In the following subsections, three basic types of single-rate cyclic executives are presented, single-rate AFAP, single-rate time-driven AFAP, and single-rate periodic.

4.1.1 Single-rate AFAP

Figure 7 illustrates a single-rate AFAP cyclic executive.

```
while( 1 ){
   τ1;
   τ2;
   τ3;
   /* ... */
   τ12;
}
```

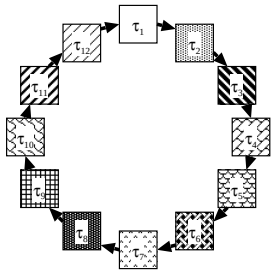

Figure 7: Graphical representation and code structure of a single-rate AFAP cyclic executive.

A single-rate AFAP cyclic executive has at least three general disadvantages. A first disadvantage is that tasks may experience *drift*[2], which is generally considered undesirable. We will illustrate this by considering the start times of jobs of a task τ_i. The start time $s_{i,k}$ of job $\iota_{i,k}$ can be bounded by

$$s_{1,0} + k \times \sum_{1 \leq j \leq n} BC_j + \sum_{1 \leq j < i} BC_j \leq s_{i,k} \leq s_{1,0} + k \times \sum_{1 \leq j \leq n} WC_j + \sum_{1 \leq j < i} WC_j.$$

When $\sum_{1 \leq j \leq n} BC_j < \sum_{1 \leq j \leq n} WC_j$, the length of the interval in which job $\iota_{i,k}$ may be started becomes arbitrary large for increasing k. Stated differently, task τ_i experiences *unbounded start jitter* or *drift*.

A second general disadvantage is that a single-rate AFAP cyclic executive is *energy inefficient*, because the processor is constantly active executing tasks. The single-rate time-driven AFAP cyclic executive, described in the next subsection, resolves both these disadvantages.

A third disadvantage is that whenever a system deadline is small compared to the sum of the worst-case computation times, to be more specific when $\exists_{1 \leq i \leq n} \sum_{1 \leq j \leq n} WC_j > WD_i^{\mathcal{S}} - WC_i$ as we will see in Section 5.3, the system is not schedulable. A multi-rate cyclic executive may solve this disadvantage; see Section 4.2.

4.1.2 Single-rate time-driven AFAP

For a single-rate time-driven AFAP, a cycle is started by a timer. The sum of the worst-case computation times of all tasks shall be less than the inter-arrival time $T^{\mathcal{S}}$ of the timer, otherwise the (worst-case) executions of the tasks do not 'fit' in a cycle, i.e.

$$\sum_{1 \leq i \leq n} WC_i \leq T^{\mathcal{S}}. \tag{16}$$

Figure 8 illustrates a single-rate time-driven AFAP cyclic executive with *cycle time* $T^{\mathcal{S}}$, where the cycle is started at time Φ, i.e. the start time of the first job of task τ_1 is given by $s_{1,0} = \Phi$.

[2]Drift led to a system failure of the Patriot Missile Defense at Dharan, Saudi Arabia [11].

```
int k = 0; /* cycle counter */

/* wait until time Φ, the start of the cycle */
sleep( Φ );
while( 1 ){
   τ1;
   τ2;
   τ3;
   /* ... */
   τ12;
   /* wait until the start of the next cycle */
   k = k + 1;
   sleep( Φ + k * T^S );
}
```

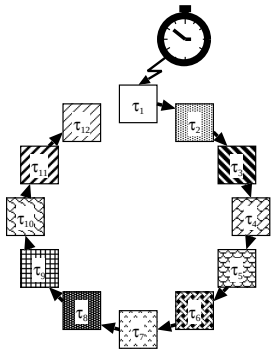

Figure 8: Graphical representation and code structure of a single-rate time-driven-AFAP cyclic executive with cycle time $T^{\mathcal{S}}$.

The start time $s_{i,k}$ of job $\iota_{i,k}$ is now bounded by

$$s_{1,0} + k \times T^{\mathcal{S}} + \sum_{1 \leq j < i} BC_j \leq s_{i,k} \leq s_{1,0} + k \times T^{\mathcal{S}} + \sum_{1 \leq j < i} WC_j.$$

The length of the interval, in which job $\iota_{i,k}$ may be started, is now bounded by a value that is independent of k. Stated differently, task τ_i experiences (*bounded*) *start jitter*. The start jitter SJ_i of task τ_i is given by

$$SJ_i = \sum_{1 \leq j < i} \left(WC_j - BC_j \right).$$

The single-rate time-driven AFAP cyclic executive therefore resolves the first disadvantage of the single-rate AFAP cyclic executive.

By starting cycles with an inter-arrival time of $T^{\mathcal{S}}$, the *spare time*, i.e. the combination of *slack*[3] and *gain time*[4], can be used to execute background tasks or run the processor at a lower, more energy efficient, clock frequency after the cycle is completed. The second disadvantage of the single-rate AFAP cyclic executive is therefore also resolved by this executive.

4.1.3 Single-rate periodic

For a single-rate periodic cyclic executive, we focus on the 'strict' case, where every task is started at a specific time. In particular, for the 'basic strict' case we assume that every task is started at the worst-case start-time relative to the start of the cycle assuming each task needs its worst-case execution time.

Figure 9 illustrates such a single-rate periodic cyclic executive with cycle time $T^{\mathcal{S}}$.

[3]Slack is the amount of time that has not been allocated to tasks, i.e. $T^{\mathcal{S}} - \sum_{1 \leq i \leq n} WC_i$ per cycle.

[4]Gain time is time that has been allocated to tasks, but is not used, i.e. at most $\sum_{1 \leq i \leq n} WC_i - BC_i$ per cycle.

```
int k = 0, /* cycle counter */
    i,     /* task counter  */
    time;

/* wait until time Φ, the start of the cycle */
time = Φ;
sleep( time );
while( 1 ){
   for( i = 1; i <= 12; ++i ){
      τi;
      /* wait until the start of the next task */
      time = time + WCi ;
      sleep( time );
   }
   /* wait until the start of the next cycle */
   k = k + 1;
   time = Φ + k * T^S;
   sleep( time );
}
```

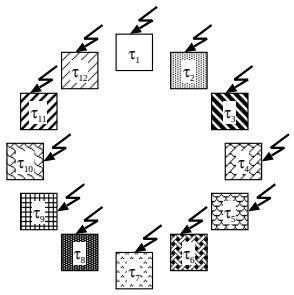

Figure 9: Graphical representation and code structure of a (basic strict) single-rate periodic cyclic executive with cycle time $T^{\mathcal{S}}$.

The start time $s_{i,k}$ of job $\iota_{i,k}$ is now exactly defined by

$$s_{i,k} = s_{1,0} + k \times T^{\mathcal{S}} + \sum_{1 \leq j < i} WC_j. \tag{17}$$

Stated differently, every task is started strictly periodically, i.e. $\underset{1 \leq i \leq n}{\forall} SJ_i = 0$, making scheduling highly deterministic.

In a non-strict case, not all tasks are necessarily started at a statically determined time, but some may be started immediately after the previous task completed. Hence, the single-rate time-driven AFAP cyclic executive is a special case of a non-strict single-rate periodic cyclic executive.

4.2 Multi-rate cyclic executive

A multi-rate cyclic executive executes every task at least once in a cycle. By allowing tasks to be executed multiple times in a cycle, the third disadvantage of a single-rate AFAP cyclic executive is addressed.

Let task τ_i be executed $m_i \geq 1$ times in the cycle. The multi-rate cyclic executive specializes to a single-rate cyclic executive when every task is executed exactly once in the cycle, i.e. $\underset{1 \leq i \leq n}{\forall} m_i = 1$.

4.2.1 Multi-rate AFAP

Figure 10 illustrates a multi-rate AFAP cyclic executive.

A multi-rate AFAP cyclic executive has the same two general disadvantages as a single-rate AFAP cyclic executive, being tasks may experience drift and the cyclic executive is energy inefficient.

4.2.2 Multi-rate time-driven AFAP

Similar to the single-rate time-driven AFAP, the sum of the worst-case computation times of all jobs in a cycle shall be less than the inter-arrival time $T^{\mathcal{S}}$ of the timer for a multi-rate time-driven AFAP, otherwise the (worst-case)

```
while( 1 ){
   τ1; τ2; τ3;
   τ1; τ4; τ5;
   τ1; τ2; τ6;
   τ1; τ7; τ8;
}
```

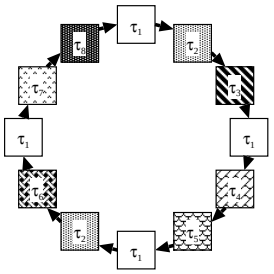

Figure 10: Graphical representation and code structure of a multi-rate AFAP cyclic executive. In this example, τ_1 is executed four times in the cycle, τ_2 is executed twice, and τ_3, τ_4, τ_5, τ_6, τ_7 and τ_8 are executed once.

executions of the tasks do not 'fit' in a cycle, i.e.

$$\sum_{1 \leq i \leq n} m_i \times WC_i \leq T^{\mathcal{S}}. \tag{18}$$

Figure 11 illustrates a multi-rate time-driven AFAP cyclic executive with cycle time $T^{\mathcal{S}}$, where the cycle is started at time Φ.

```
int k = 0; /* cycle counter */

/* wait till time Φ, the start of the cycle */
sleep( Φ );
while( 1 ){
   τ1; τ2; τ3;
   τ1; τ4; τ5;
   τ1; τ2; τ6;
   τ1; τ7; τ8;
   /* wait until the start of the next cycle*/
   k = k + 1;
   sleep( Φ + k * T^S );
}
```

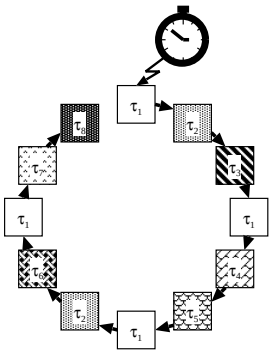

Figure 11: Graphical representation and code structure of a multi-rate time-driven AFAP cyclic executive with cycle time $T^{\mathcal{S}}$. In this example, τ_1 is executed four times in the cycle, τ_2 is executed twice, and τ_3, τ_4, τ_5, τ_6, τ_7 and τ_8 are executed once.

Similar to the single-rate time-driven AFAP, tasks experience (*bounded*) *start jitter*.

4.2.3 Multi-rate periodic

For a multi-rate periodic cyclic executive, we focus on the strict case, where every job in the cycle is started at a specific time. In particular, for the 'basic strict' case we assume that every task is started at the worst-case start-time relative to the start of the cycle assuming each job needs its worst-case execution time. Figure 12 illustrates such a multi-rate periodic cyclic executive with cycle time $T^{\mathcal{S}}$.

```
#define N 12 /* number of jobs in the cycle */

int k = 0, /* cycle counter */
    idx,   /* index         */
    time;
int task[N] = {1, 2, 3, 1, 4, 5, 1, 2, 6, 1, 7, 8};

/* wait until time Φ, the start of the cycle */
time = Φ ;
sleep( time );
while( 1 ){
   for( idx = 0; idx < N; ++idx ){
      τ_task[idx];
      /* wait until the start of the next task */
      time = time + WC_task[idx];
      sleep( time );
   }
   /* wait until the start of the next cycle */
   k = k + 1;
   time = Φ + k * T^6;
   sleep( time );
}
```

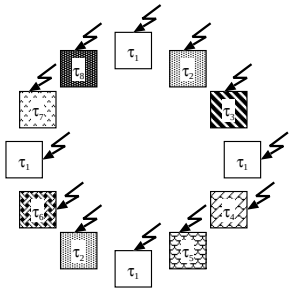

Figure 12: Graphical representation and code structure of a (basic strict) multi-rate periodic cyclic executive with cycle time $T^{\mathcal{S}}$. In this example, τ_1 is executed four times in the cycle, τ_2 is executed twice, and τ_3, τ_4, τ_5, τ_6, τ_7 and τ_8 are executed once.

In a strict case, the start time of every job is statically determined. Observe that, unlike the case of a single-rate periodic cyclic executive, the inter start-times of jobs of a task may fluctuate, however. As an example, consider Figure 12. The inter start-times of the jobs $\iota_{1,0}, \iota_{1,1}, \ldots, \iota_{1,4}$ of task τ_1 are given by:

$$\begin{aligned} s_{1,1} - s_{1,0} &= WC_1 + WC_2 + WC_3; \\ s_{1,2} - s_{1,1} &= WC_1 + WC_4 + WC_5; \\ s_{1,3} - s_{1,2} &= WC_1 + WC_2 + WC_6; \\ s_{1,4} - s_{1,3} &= WC_1 + WC_7 + WC_8. \end{aligned}$$

The inter start-times differ as soon as not all right-hand sides of these equations are equal. Hence, although every specific job in the cycle is started strictly periodically, the inter start-times of successive jobs of a task within a cycle and between cycles may fluctuate.

In a non-strict case, not all jobs are started at a statically determined time, but some are started immediately after the previous task completed. Hence, the multi-rate time-driven AFAP cyclic executive is a special case of a non-strict multi-rate periodic cyclic executive.

5 Exact worst-case analysis for single-rate cyclic executives

In this section, we first revisit the general exact condition for the worst-case schedulability of a system $\mathcal{S}$ when tasks are scheduled by a basic cyclic executive. We subsequently present two example systems, which we will use in the remainder of the section for illustration purposes. Next, we consider analysis for each of the three basic single-rate cyclic executives. Apart from providing exact worst-case analysis for each cyclic executive, we also consider lower and upper

bounds for the fraction of time that can be gained for background processing by the time-driven AFAP and periodic cyclic executives compared to the AFAP version. Finally, the single-rate cyclic executives are also compared with respect to schedulability.

5.1 Worst-case schedulability condition revisited

Given the cyclic nature of the single-rate cyclic executive and the fact that every task is executed exactly once in the cycle, it is sufficient to consider at most 2, or 1 *pair* of, successive jobs of τ_i. Hence, we can simplify Equation (13) to

$$\underset{1 \leq i \leq n}{\forall} f_{i,1} - s_{i,0} \leq WD_i^{\mathcal{S}}. \tag{19}$$

To determine schedulability, we therefore have to test at most n conditions.

In the remainder of this section, we will stick to Equation (13), however, to ease comparison between the exact conditions of the multi-rate executives presented in the next section with their specializations for single-rate executives.

5.2 Two example systems

Tables 2 and 3 present task sets $\mathcal{T}^{\mathcal{S}_2}$ and $\mathcal{T}^{\mathcal{S}_3}$ and associated system deadlines of systems $\mathcal{S}_2$ and $\mathcal{S}_3$, respectively. These two example systems will be used for illustration purposes in the remainder of this section.

Table 2: Task characteristics of task set $\mathcal{T}^{\mathcal{S}_2}$ and associated system deadlines of system $\mathcal{S}_2$.

	BC	WC	$WD^{\mathcal{S}_2}$
τ_1	1	2	10
τ_2	2	4	14

Table 3: Task characteristics of task set $\mathcal{T}^{\mathcal{S}_3}$ and associated system deadlines of system $\mathcal{S}_3$.

	BC	WC	$WD^{\mathcal{S}_3}$
τ_1	2	3	11
τ_2	1	2	14
τ_3	3	4	17

5.3 Single-rate AFAP

Theorem 1. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a single-rate AFAP cyclic executive iff (i.e. if-and-only-if) the following condition holds*

$$\underset{1 \leq i \leq n}{\forall} WC_i + \sum_{1 \leq j \leq n} WC_j \leq WD_i^{\mathcal{S}}. \tag{20}$$

Proof. The proof is given by construction. For a single-rate AFAP cyclic executive, all other tasks are executed exactly once between the execution of two successive jobs of a task τ_i. Hence, we can write

$$f_{i,k+1} \leq s_{i,k} + WC_i + \sum_{1 \leq j \leq n} WC_j$$

and derive

$$f_{i,k+1} - s_{i,k} - WC_i \leq \sum_{1 \leq j \leq n} WC_j.$$

Because this bound on $f_{i,k+1} - s_{i,k}$ is tight by construction for all τ_i, the result now immediately follows from Equation (13). □

Based on Theorem 1, we derive that system $\mathcal{S}_2$ meets all its worst-case deadlines; see also Figure 13. Similarly, we derive that system $\mathcal{S}_3$ is not schedulable, because $WD_1^{\mathcal{S}_2}$ may be missed, i.e. $\sum_{1 \leq i \leq |\mathcal{T}^{\mathcal{S}_3}|} WC_i + WC_1 = 12 > WD_1^{\mathcal{S}_3} = 11$; see also Figure 14.

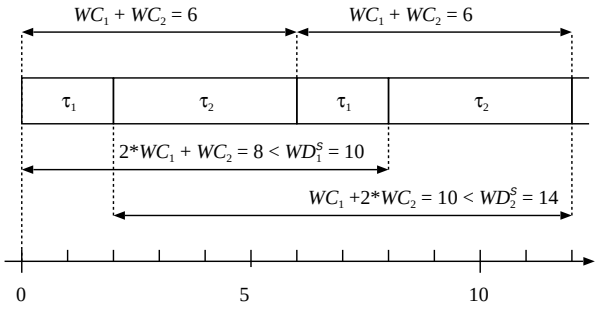

Figure 13: A situation of worst-case execution for system $\mathcal{S}_2$ under a single-rate AFAP cyclic executive. Both system deadlines $WD_1^{\mathcal{S}_2}$ and $WD_2^{\mathcal{S}_2}$ are met.

5.4 Single-rate time-driven AFAP

In this section, we first present exact worst-case analysis for the single-rate time-driven AFAP cyclic executive. We subsequently compare the schedulability of the single-rate AFQP cyclic executive with the single-rate time-driven AFAP cyclic executive. Finally, we provide lower and upper bounds for the fraction of time that can be gained for background processing by the time-driven AFAP cyclic executive compared to the AFAP version.

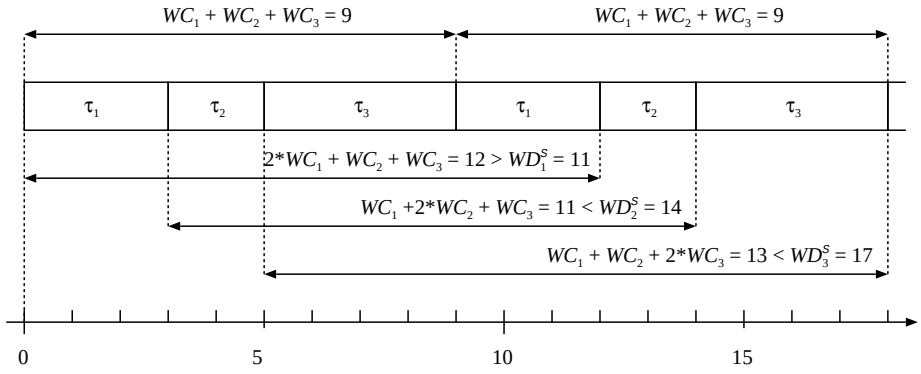

Figure 14: A situation of worst-case execution for system $\mathcal{S}_3$ under a single-rate AFAP cyclic executive. System deadline $WD_1^{\mathcal{S}_3}$ is missed.

5.4.1 Exact worst-case analysis

Theorem 2. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a single-rate time-driven AFAP cyclic executive iff (i.e. if-and-only-if) the following condition holds*

$$\underset{T^{\mathcal{S}}}{\exists} \sum_{1 \le i \le n} WC_i \le T^{\mathcal{S}} \le \min_{1 \le i \le n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{1 \le j < i} (WC_j - BC_j) + WC_i \right) \right\}. \tag{21}$$

Proof. We first observe that the lower bound on $T^{\mathcal{S}}$, given by $\sum_{1 \le i \le n} WC_i \le T^{\mathcal{S}}$, is required to guarantee that the (worst-case) executions of the tasks 'fit' in a cycle.

Consider a task τ_i. The term $f_{i,k+1} - s_{i,k}$ is maximized for a single-rate time-driven AFAP cyclic executive when job $\iota_{i,k}$ starts *as early as possible* and job $\iota_{i,k+1}$ finishes *as late as possible* in their successive cycles. Denoting the cycle time by $T^{\mathcal{S}}$, we find (by construction)

$$\begin{aligned} s_{i,k} &\ge k \times T^{\mathcal{S}} + \sum_{1 \le j < i} BC_j \\ f_{i,k+1} &\le (k+1) \times T^{\mathcal{S}} + \sum_{1 \le j \le i} WC_j. \end{aligned}$$

Hence,

$$f_{i,k+1} - s_{i,k} \le T^{\mathcal{S}} + \sum_{1 \le j < i} (WC_j - BC_j) + WC_i.$$

Because this bound on $f_{i,k+1} - s_{i,k}$ is tight by construction for all τ_i, the condition

$$\underset{1 \le i \le n}{\forall} T^{\mathcal{S}} + \sum_{1 \le j < i} (WC_j - BC_j) + WC_i \le WD_i^{\mathcal{S}}$$

now immediately follows from Equation (13). Hence, an upper bound on $T^{\mathcal{S}}$ is

given by

$$T^{\mathcal{S}} \leq \min_{1 \leq i \leq n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{1 \leq j < i} (WC_j - BC_j) + WC_i \right) \right\}.$$

Together with the lower bound on $T^{\mathcal{S}}$ given earlier, this concludes the proof. □

Based on Theorem 2 we conclude that system $\mathcal{S}_2$ meets all its worst-case deadlines under a single-rate time-driven AFAP cyclic executive, because there exists a cycle time $T^{\mathcal{S}_2}$ that satisfies the relation $6 \leq T^{\mathcal{S}_2} \leq \min\{8, 9\} = 8$. Figure 15 shows a situation of worst-case execution for system $\mathcal{S}_2$ for a cycle time $T^{\mathcal{S}_2} = 8$. As can be seen, both system deadlines are met. Observe that

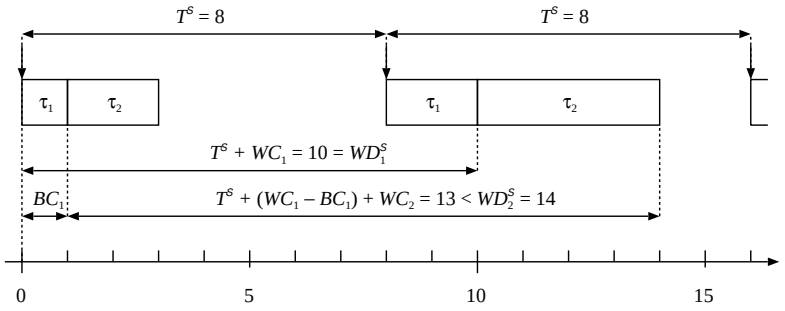

Figure 15: A situation of worst-case execution for system $\mathcal{S}_2$ under a single-rate time-driven AFAP cyclic executive with $T^{\mathcal{S}_2} = 8$. Both system deadlines $WD_1^{\mathcal{S}_2}$ and $WD_2^{\mathcal{S}_2}$ are met.

increasing the cycle time beyond 8 will cause system deadline $WD_1^{\mathcal{S}_2}$ to be missed.

Based on Theorem 2 we conclude that system $\mathcal{S}_3$ is not schedulable under a single-rate AFAP cyclic executive, because there does not exist a cycle time $T^{\mathcal{S}_3}$ that satisfies the relation $9 \leq T^{\mathcal{S}_3} \leq \min\{11 - 3, 14 - (3 - 2 + 2), 17 - (3 - 2 + 2 - 1 + 4)\} = \min\{8, 11, 11\} = 8$. Figure 16 shows a situation of worst-case execution for system $\mathcal{S}_3$ for a cycle time $T^{\mathcal{S}_3} = 9$, illustrating that $\mathcal{S}_3$ is not schedulable.

5.4.2 Comparison with an AFAP cyclic executive

Next, we compare single-rate AFAP and single-rate time-driven AFAP cyclic executives. We first present a lemma stating that if a system meets its worst-case deadlines under a single-rate time-driven AFAP cyclic executive then it will also meet its worst-case deadlines under a single-rate AFAP cyclic executive. We subsequently present an example system that meets its worst-case deadlines under a single-rate AFAP cyclic executive but is not schedulable by a single-rate time-driven AFAP cyclic executive. We therefore conclude that the single-rate AFAP cyclic executive dominates the single-rate time-driven AFAP cyclic executive.

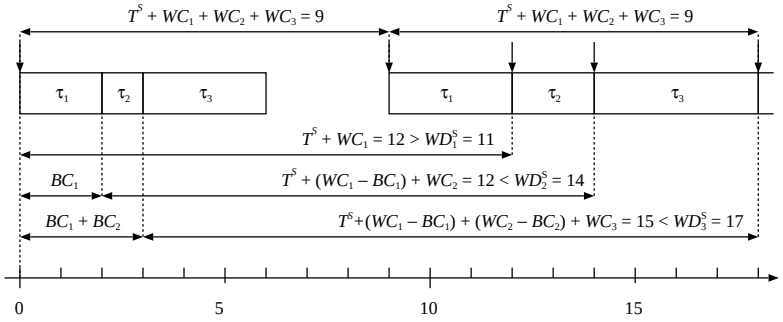

Figure 16: A situation of worst-case execution for system $\mathcal{S}_3$ under a single-rate time-driven AFAP cyclic executive with $T^{\mathcal{S}_3} = 9$. System deadline $WD_1^{\mathcal{S}_3}$ is not met, whereas both system deadlines $WD_2^{\mathcal{S}_3}$ and $WD_3^{\mathcal{S}_3}$ are met.

Lemma 1. *When a system $\mathcal{S}$ is schedulable under a single-rate time-driven AFAP cyclic executive, it is also schedulable under a single-rate AFAP cyclic executive.*

Proof. We first recall from Section 3.2 that the best-case schedulability condition is independent of the cyclic executive. To prove the lemma, we therefore only have to show that when the worst-case deadlines of $\mathcal{S}$ are met under a single-rate time-driven AFAP cyclic executives, these deadlines are also met under a single-rate AFAP cyclic executive. We now prove the lemma by deriving Equation (20) from Equation (21).

$$
\begin{aligned}
& \underset{T^{\mathcal{S}}}{\exists} \sum_{1 \le i \le n} WC_i \le T^{\mathcal{S}} \le \min_{1 \le i \le n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{1 \le j < i} (WC_j - BC_j) + WC_i \right) \right\} \\
\Leftrightarrow \quad & \sum_{1 \le i \le n} WC_i \le \min_{1 \le i \le n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{1 \le j < i} (WC_j - BC_j) + WC_i \right) \right\} \\
\Leftrightarrow \quad & \underset{1 \le i \le n}{\forall} \sum_{1 \le j \le n} WC_j \le WD_i^{\mathcal{S}} - \left(\sum_{1 \le j < i} (WC_j - BC_j) + WC_i \right) \\
\Rightarrow \quad & \{ \underset{1 \le i \le n}{\forall} WC_i \ge BC_i \} \underset{1 \le i \le n}{\forall} \sum_{1 \le j \le n} WC_j \le WD_i^{\mathcal{S}} - WC_i \\
\Leftrightarrow \quad & \underset{1 \le i \le n}{\forall} WC_i + \sum_{1 \le j \le n} WC_j \le WD_i^{\mathcal{S}}
\end{aligned}
$$

□

Now consider system $\mathcal{S}_4$ described in Table 4. Based on Theorem 1, we conclude that system $\mathcal{S}_4$ meets all its worst-case deadlines under a single-rate AFAP cyclic executive; see also Figure 17.

Based on Theorem 2, we conclude that system $\mathcal{S}_4$ is not schedulable under a single-rate time-driven AFAP cyclic executive, however, because there is no $T^{\mathcal{S}_4}$ that satisfies the relation $8 \le T^{\mathcal{S}_4} \le \min\{9, 7\} = 7$. Figure 18 shows a

Table 4: Task characteristics of task set $\mathcal{T}^{\mathcal{S}_4}$ and associated system deadlines of system $\mathcal{S}_4$.

	BC	WC	$WD^{\mathcal{S}_4}$
τ_1	1	3	12
τ_2	2	5	14

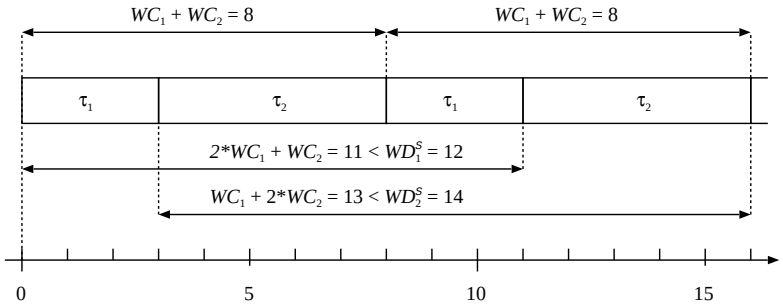

Figure 17: A situation of worst-case execution for system $\mathcal{S}_4$ under a single-rate AFAP cyclic executive. Both system deadlines $WD_1^{\mathcal{S}_4}$ and $WD_2^{\mathcal{S}_4}$ are met.

situation with a worst-case execution and a cycle time $T^{\mathcal{S}_4} = WC_1 + WC_2 = 8$, where the system deadline $WD_2^{\mathcal{S}_4}$ is missed.

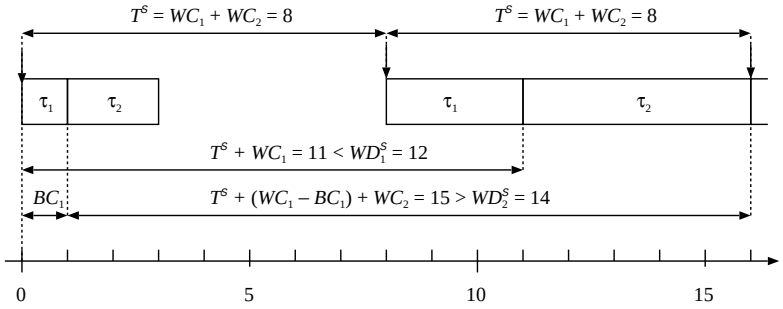

Figure 18: A situation of worst-case execution for system $\mathcal{S}_4$ under a single-rate time-driven AFAP cyclic executive with a cycle time $T^{\mathcal{S}_4} = WC_1 + WC_2 = 8$. System deadline $WD_2^{\mathcal{S}_4}$ is missed.

Theorem 3. *The single-rate AFAP cyclic executive dominates the single-rate time-driven AFAP cyclic executive.*

Proof. Follows immediately from Lemma 1 and system $\mathcal{S}_4$. □

5.4.3 Bounds on fraction of spare time

Finally, we consider bounds on the fraction of time that can be gained for background processing by a single-rate time-driven AFAP cyclic executive compared to a single-rate AFAP cyclic executive.

The slack per cycle of a single time-driven cyclic executive is equal to $T^{\mathcal{S}} - \sum_{1 \leq i \leq n} WC_i$. The gain time per cycle is at most $\sum_{1 \leq i \leq n} (WC_i - BC_i)$. The spare time per cycle is therefore bounded by $T^{\mathcal{S}} - \sum_{1 \leq i \leq n} WC_i$ and $T^{\mathcal{S}} - \sum_{1 \leq i \leq n} BC_i$.

Lemma 2. *Whenever a system $\mathcal{S}$ is schedulable under a single-rate time-driven AFAP cyclic executive with a cycle time of $T^{\mathcal{S}}$, a fraction $f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}})$ given by*

$$\frac{\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} WC_i\right)}{T^{\mathcal{S}}} \leq f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}}) \leq \frac{\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} BC_i\right)}{T^{\mathcal{S}}}$$

of the time can be gained for background processing by a single-rate time-driven AFAP cyclic executive compared to a single-rate AFAP cyclic executive.

Proof. The term $\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} WC_i\right)$ represents the worst-case (i.e. minimum) amount of spare time in a cycle and the term $\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} BC_i\right)$ represents the best-case (i.e. maximum) amount of spare time. The result therefore follows immediately. □

In case of system $\mathcal{S}_2$ and $T^{\mathcal{S}_2} = 8$, we find $\frac{1}{4} \leq f^{\mathcal{S}_2,\text{gain}}(8) \leq \frac{1}{2}$.

5.5 Single-rate periodic cyclic executive

By explicitly activating each task in the cycle, tasks do not experience any start jitter, as already described by Equation (17) in Section 4.1.3. This is also illustrated for system $\mathcal{S}_2$ in Figure 19. As illustrated in Figure 20, system $\mathcal{S}_4$

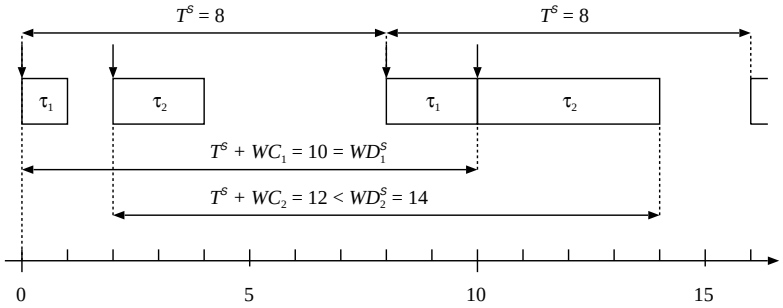

Figure 19: A situation of worst-case execution for system $\mathcal{S}_2$ under a (basic strict) single-rate periodic cyclic executive with a cycle time $T^{\mathcal{S}_2} = 8$. All deadlines of $\mathcal{S}_2$ are met.

meets all its worst-case deadlines under a single-rate periodic cyclic executive.

5.5.1 Exact worst-case analysis

By removing the start jitter through the usage of a (strict) single-rate periodic cyclic executive, we can adapt Theorem 2:

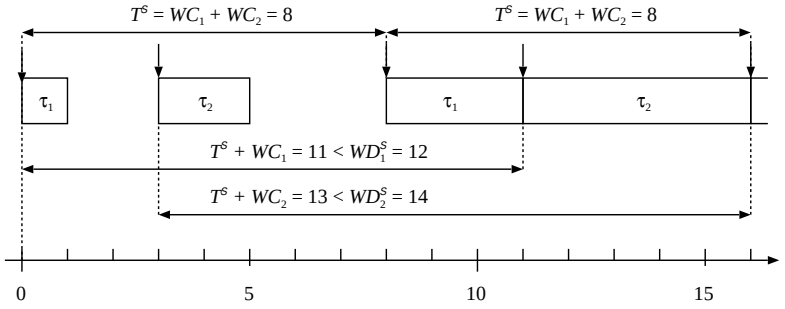

Figure 20: A situation of worst-case execution for system $\mathcal{S}_4$ under a (basic strict) single-rate periodic cyclic executive with a cycle time $T^{\mathcal{S}_4} = 8$. All deadlines of $\mathcal{S}_4$ are met.

Theorem 4. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a (strict) single-rate periodic cyclic executive iff the following condition holds*

$$\exists_{T^{\mathcal{S}}} \sum_{1 \le j \le n} WC_j \le T^{\mathcal{S}} \le \min_{1 \le i \le n} \left\{ WD_i^{\mathcal{S}} - WC_i \right\}. \tag{22}$$

Proof. Follows immediately from Equation (16) and by simply replacing BC_i by WC_i in Equation (21). □

For system $\mathcal{S}_4$ we find that it will meet all its worst-case deadlines under a single-rate periodic cyclic executive when $8 \le T^{\mathcal{S}_4} \le \min\{9, 9\} = 9$. Figure 21 illustrates a schedule for $T^{\mathcal{S}_4} = 9$.

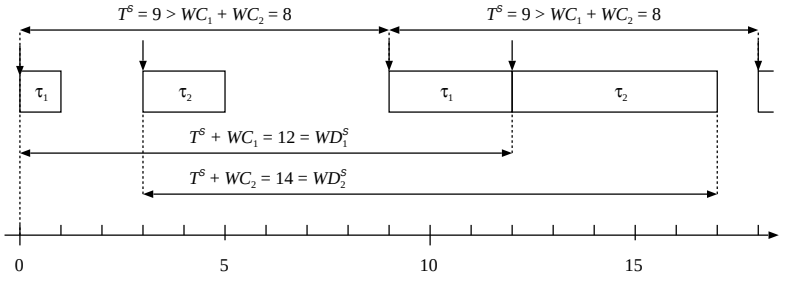

Figure 21: A situation of worst-case execution for system $\mathcal{S}_4$ under a (basic strict) single-rate periodic cyclic executive with a cycle time $T^{\mathcal{S}_4} = 9$. All deadlines of $\mathcal{S}_4$ are met.

5.5.2 Comparison with AFAP cyclic executives

Next, we compare the single-rate periodic cyclic executive with the other basic single-rate cyclic executives.

Theorem 5. *The single-rate AFAP cyclic executive and the single-rate periodic cyclic executive are equivalent, i.e. can schedule the same systems.*

Proof. We prove the theorem by showing that by taking $T^{\mathcal{S}}$ to be equal to the lower bound in Equation (22), Equation (22) and Equation (19) are equivalent. We first rewrite Equation (22) to

$$\underset{T^{\mathcal{S}}}{\exists} \underset{1\leq i\leq n}{\forall} \sum_{1\leq j\leq n} WC_j \leq T^{\mathcal{S}} \leq WD_i^{\mathcal{S}} - WC_i.$$

We now derive for $T^{\mathcal{S}} = \sum_{1\leq j\leq n} WC_j$

$$\begin{aligned} & \underset{1\leq i\leq n}{\forall} \sum_{1\leq j\leq n} WC_j \leq WD_i^{\mathcal{S}} - WC_i \\ \Leftrightarrow \quad & \underset{1\leq i\leq n}{\forall} \sum_{1\leq j\leq n} WC_j + WC_i \leq WD_i^{\mathcal{S}}, \end{aligned}$$

where the latter relation is identical to Equation (19) of Theorem 1. □

Theorem 6. *The single-rate periodic cyclic executive dominates the single-rate time-driven AFAP cyclic executive.*

Proof. Follows immediately from Theorems 3 and 5. □

5.5.3 Bounds on fraction of spare time

Finally, we consider bounds on the fraction of time that can be gained for background processing by a single-rate periodic cyclic executive compared to a single-rate AFAP cyclic executive.

Lemma 3. *Whenever a system $\mathcal{S}$ is schedulable under a single-rate time-driven AFAP cyclic executive with a cycle time of $T^{\mathcal{S}}$, a fraction $f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}})$ given by*

$$\frac{\left(T^{\mathcal{S}} - \sum_{1\leq i\leq n} WC_i\right)}{T^{\mathcal{S}}} \leq f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}}) \leq \frac{\left(T^{\mathcal{S}} - \sum_{1\leq i\leq n} BC_i\right)}{T^{\mathcal{S}}}$$

of the time can be gained for background processing by a single-rate periodic cyclic executive compared to a single-rate AFAP cyclic executive.

Proof. Similar to the proof of Lemma 2. □

In case of system $\mathcal{S}_4$ and $T^{\mathcal{S}_4} = 9$, we find $\frac{1}{9} \leq f^{\mathcal{S}_4,\text{gain}}(9) \leq \frac{2}{3}$.

6 Exact worst-case analysis for multi-rate cyclic executives

As mentioned in the Section 1.3, we assume that the sequence of tasks to be executed in the cycle is known.

Given the cyclic nature of the multi-rate cyclic executive and the fact that every task τ_i is executed m_i times in the cycle, it is sufficient to consider at most m_i+1, or m_i *pairs* of, successive jobs of τ_i. Hence, we can simplify Equation (13) to

$$\underset{1\leq i\leq n}{\forall} \left(\underset{0\leq k\leq m_i-1}{\forall} f_{i,k+1} - s_{i,k} \leq WD_i^{\mathcal{S}} \right). \tag{23}$$

To determine whether or not system $\mathcal{S}$ meets all its worst-case deadlines, we therefore have to test at most $N = \sum_{1 \leq i \leq n} m_i$ conditions, where N reduces to n for the special case of single-rate executives, i.e. when $\forall_{1 \leq i \leq n} \; m_i = 1$. From these m_i pairs, $m_i - 1$ pairs are executed in the same cycle and 1 pair is executed in two successive cycles.

6.1 Multi-rate AFAP

Theorem 7. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a multi-rate AFAP cyclic executive scheduling a sequence of tasks* `task[]` *iff the following condition holds*

$$\forall_{1 \leq i \leq n} \quad \max \left\{ \begin{array}{l} \max\limits_{0 \leq k < m_i - 1} \sum\limits_{\texttt{index}(\iota_{i,k}) \leq \ell \; \leq \texttt{index}(\iota_{i,k+1})} WC_{\texttt{task}[\ell]} \\ , \sum\limits_{\substack{\texttt{index}(\iota_{i,m_i-1}) \leq \ell \leq N-1 \; \vee \\ 0 \leq \ell \leq \texttt{index}(\iota_{i,0})}} WC_{\texttt{task}[\ell]} \end{array} \right\} \leq WD_i^{\mathcal{S}}. \tag{24}$$

Proof. The proof is based on a construction argument. We consider two cases, a first case where the jobs $\iota_{i,k}$ and $\iota_{i,k+1}$ are in the same cycle, i.e. $0 \leq k < m_i - 1$ and a second case where these jobs are in successive cycles, i.e. $k = m_i - 1$. For both cases, we write $f_{i,k+1} - s_{i,k}$ in terms of `task[]` and `int index( job )`, where we assume the start time of the system is given by 0, i.e. $s_{\texttt{task[0]},0} = 0$.

When the successive jobs $\iota_{i,k}$ and $\iota_{i,k+1}$ of task τ_i are in the same cycle, $f_{i,k+1} - s_{i,k}$ is bounded by the sum of the worst-case computation times of all jobs in the cycle starting with job $\iota_{i,k}$ up to and including job $\iota_{i,k+1}$, i.e.

$$f_{i,k+1} - s_{i,k} \leq \sum_{\texttt{index}(\iota_{i,k}) \leq \ell \leq \texttt{index}(\iota_{i,k+1})} WC_{\texttt{task}[\ell]}. \tag{25}$$

When the successive jobs $\iota_{i,k}$ and $\iota_{i,k+1}$ of task τ_i are in successive cycles, i.e. $k = m_i - 1$, $f_{i,k+1} - s_{i,k} = f_{i,m_i} - s_{i,m_i-1}$ is bounded by the sum of the worst-case computation times of all jobs (*i*) in the cycle starting with job ι_{i,m_i-1} up to the end of the cycle and (*ii*) in the successive cycle from the start of the cycle up to and including job ι_{i,m_i}, i.e.

$$f_{i,m_i} - s_{i,m_i-1} \leq \sum_{\substack{\texttt{index}(\iota_{i,m_i-1}) \leq \ell \leq N-1 \; \vee \\ 0 \leq \ell \leq \texttt{index}(\iota_{i,0})}} WC_{\texttt{task}[\ell]}. \tag{26}$$

The term $f_{i,k+1} - s_{i,k}$ is therefore bounded by the maximum of the right-hand sides of Equations (25) and (26). Because this bound on $f_{i,k+1} - s_{i,k}$ is tight by construction for all τ_i, the result now immediately follows from Equation (13). □

Let the cycle for system $\mathcal{S}_2$ be given by `task[] = {1, 2, 1, 3}`, i.e. $m_1 = 2$

and $m_2 = m_3 = 1$. Using Theorem 7, we find for task

- τ_1: $\max\{8, 10\} = 10 \leq WD_1^{\mathcal{S}_3} = 11$, hence the deadline $WD_1^{\mathcal{S}_3}$ is met;
- τ_2: $\max\{-\inf, 14\} = 14 \leq WD_2^{\mathcal{S}_3} = 14$, hence the deadline $WD_2^{\mathcal{S}_3}$ is met;
- τ_3: $\max\{-\inf, 16\} = 16 \leq WD_3^{\mathcal{S}_3} = 17$, hence the deadline $WD_3^{\mathcal{S}_3}$ is met.

We therefore conclude that system $\mathcal{S}_2$ meets all its worst-case deadlines under a multi-rate AFAP cyclic executive using the cycle `{1, 2, 1, 3}`.

Figure 22 shows a situation of worst-case execution for system $\mathcal{S}_3$.

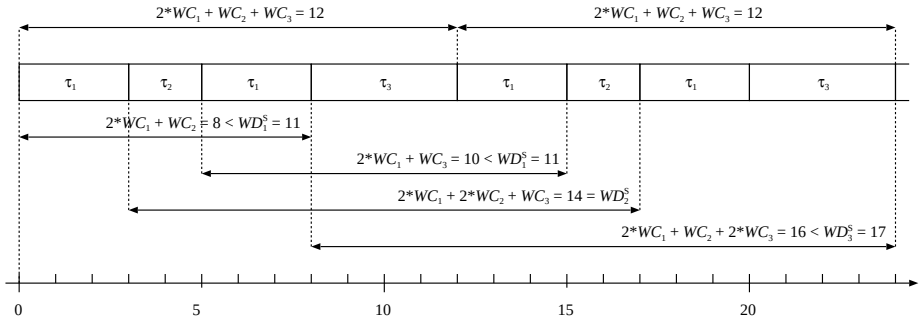

Figure 22: A situation of worst-case execution for system $\mathcal{S}_3$ under a multi-rate AFAP cyclic executive using the cycle `{1, 2, 1, 3}`. All deadlines of $\mathcal{S}_3$ are met.

6.2 Multi-rate time-driven AFAP

Similar to Section 5.4, we first present exact worst-case analysis for the multi-rate time-driven AFAP cyclic executive, subsequently compare the schedulability of the multi-rate AFQP cyclic executive with the multi-rate time-driven AFAP cyclic executive, and finally provide lower and upper bounds for the fraction of time that can be gained for background processing by the time-driven AFAP cyclic executive compared to the AFAP version.

6.2.1 Exact worst-case analysis

Theorem 8. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a multi-rate time-driven AFAP cyclic executive scheduling a sequence of tasks* `task[]` *iff the following condition holds*

$$\bigwedge_{1 \leq i \leq n} \max_{0 \leq k < m_i - 1} \sum_{\texttt{index}(\iota_{i,k}) \leq \ell \leq \texttt{index}(\iota_{i,k+1})} WC_{\texttt{task}[\ell]} \leq WD_i^{\mathcal{S}}$$

$$\wedge \bigvee_{T^{\mathcal{S}}} \sum_{1 \leq i \leq n} m_i \times WC_i \leq T^{\mathcal{S}} \leq \min_{1 \leq i \leq n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{0 \leq \ell \leq \texttt{index}(\iota_{i,0})} WC_{\texttt{task}[\ell]} - \sum_{0 \leq \ell \leq \texttt{index}(\iota_{i,m_i-1})-1} BC_{\texttt{task}[\ell]} \right) \right\}. \quad (27)$$

Proof. The proof is based on a construction argument. Similar to the multi-rate AFAP case, we consider two cases, a first case where $\iota_{i,k}$ and $\iota_{i,k+1}$ are in the same cycle, i.e. $0 \leq k < m_i - 1$ and a second case where these jobs are in successive cycles, i.e. $k = m_i - 1$.

When the successive jobs $\iota_{i,k}$ and $\iota_{i,k+1}$ of task τ_i are in the same cycle, $f_{i,k+1} - s_{i,k}$ is bounded by the sum of the worst-case computation times of all jobs in the cycle starting with job $\iota_{i,k}$ up to and including job $\iota_{i,k+1}$, as for the case of multi-rate AFAP, i.e. Equation (25).

For the successive jobs ι_{i,m_i-1} and ι_{i,m_i} of task τ_i, we first observe that $T^{\mathcal{S}}$ shall satisfy Equation (16), otherwise the (worst-case) executions of the tasks do not 'fit' in a cycle. Next, tight bounds on s_{i,m_i-1} and f_{i,m_i} are given by

$$\begin{aligned} s_{i,m_i-1} &\geq \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,m_i-1})-1} BC_{\mathtt{task}[\ell]} \\ f_{i,m_i} &\leq T^{\mathcal{S}} + \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,0})} WC_{\mathtt{task}[\ell]} \end{aligned}$$

Hence,

$$f_{i,m_i} - s_{i,m_i-1} \leq T^{\mathcal{S}} + \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,0})} WC_{\mathtt{task}[\ell]} - \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,m_i-1})-1} BC_{\mathtt{task}[\ell]}. \tag{28}$$

The term $f_{i,k+1} - s_{i,k}$ is therefore bounded by the maximum of the right-hand sides of Equations (25) and (28). Because this bound on $f_{i,k+1} - s_{i,k}$ is tight by construction for all τ_i, the condition

$$\underset{1 \leq i \leq n}{\forall} T^{\mathcal{S}} + \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,0})} WC_{\mathtt{task}[\ell]} - \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,m_i-1})-1} BC_{\mathtt{task}[\ell]} \leq WD_i^{\mathcal{S}},$$

now immediately follows from Equation (13). Hence, and upper bound on $T^{\mathcal{S}}$ is given by

$$T^{\mathcal{S}} \leq \min_{1 \leq i \leq n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,0})} WC_{\mathtt{task}[\ell]} - \sum_{0 \leq \ell \leq \mathtt{index}(\iota_{i,m_i-1})-1} BC_{\mathtt{task}[\ell]} \right) \right\}.$$

□

Based on Theorem 8, we can make the following derivation for system $\mathcal{S}_3$. Because only $m_1 > 1$, we only need to check the first clause for τ_1, and find $\sum_{\mathtt{index}(\iota_{1,0}) \leq \ell \leq \mathtt{index}(\iota_{1,1})} WC_{\mathtt{task}[\ell]} = WC_1 + WC_2 + WC_1 = 8 < WD_1^{\mathcal{S}_3} = 11$. Hence that clause is satisfied. Next, we need to check the second clause, and find $12 \leq T^{\mathcal{S}_3} \leq \min\{11-(3-3), 14-(5-2), 17-(12-5)\} = \min\{11, 11, 10\} = 10$. We therefore conclude that system $\mathcal{S}_3$ is not schedulable under a multi-rate time-driven AFAP cyclic executive. Figure 23 shows a worst-case execution for system $\mathcal{S}_3$ for a cycle time $T^{\mathcal{S}_3} = 12$. As can be observed from the figure, $WD_1^{\mathcal{S}_3}$ and $WD_2^{\mathcal{S}_3}$ are both exceeded by 1 and $WD_3^{\mathcal{S}_3}$ is exceeded by 2.

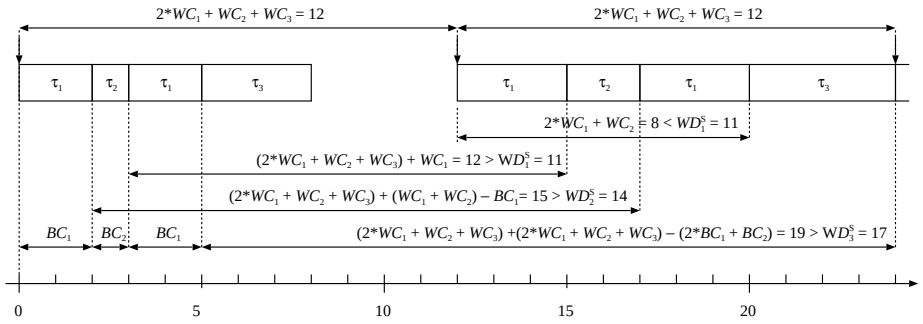

Figure 23: A situation of worst-case execution for system $\mathcal{S}_3$ under a multi-rate time-driven cyclic executive with $T^{\mathcal{S}_3} = 12$ and using the cycle {1, 2, 1, 3}. All three system deadlines $WD_1^{\mathcal{S}_3}$, $WD_2^{\mathcal{S}_3}$ and $WD_3^{\mathcal{S}_3}$ are missed.

6.2.2 Comparison with an AFAP cyclic executive

Next, we compare multi-rate AFAP and multi-rate time-driven AFAP cyclic executives. We first present a lemma stating that if a system is schedulable by a multi-rate time-driven AFAP cyclic executive, then it is also schedulable by a multi-rate AFAP cyclic executive. Through system $\mathcal{S}3$, we already presented an example that is schedulable by a multi-rate AFAP cyclic executive but not by a multi-rate time-driven AFAP cyclic executive. We therefore conclude that the multi-rate AFAP cyclic executive dominates the multi-rate time-driven AFAP cyclic executive.

Lemma 4. *When a system $\mathcal{S}$ with a sequence of tasks* `task[]` *is schedulable under a multi-rate time-driven AFAP cyclic executive, it is also schedulable under a multi-rate AFAP cyclic executive.*

Proof. Given Theorems 7 and 8, we have to prove that

$$\sum_{\substack{\texttt{index}(\iota_{i,m_i-1})\leq\ell\leq N-1 \ \vee \\ 0\leq\ell\leq\texttt{index}(\iota_{i,0})}} WC_{\texttt{task}[\ell]} \leq T^{\mathcal{S}} + \sum_{0\leq\ell\leq\texttt{index}(\iota_{i,0})} WC_{\texttt{task}[\ell]} - \sum_{0\leq\ell\leq\texttt{index}(\iota_{i,m_i-1})-1} BC_{\texttt{task}[\ell]}.$$

We first remove the term $\sum_{0\leq\ell\leq\texttt{index}(\iota_{i,0})} WC_{\texttt{task}[\ell]}$ from both sides of the relation.

$$\sum_{\texttt{index}(\iota_{i,m_i-1})\leq\ell\leq N-1} WC_{\texttt{task}[\ell]} \leq T^{\mathcal{S}} - \sum_{0\leq\ell\leq\texttt{index}(\iota_{i,m_i-1})-1} BC_{\texttt{task}[\ell]}$$

Next, we replace $T^{\mathcal{S}}$ by its lower bound $\sum_{1\leq\ell\leq N-1} WC_{\texttt{task}[\ell]}$ and remove the term $\sum_{\texttt{index}(\iota_{i,m_i-1})\leq\ell\leq N-1} WC_{\texttt{task}[\ell]}$ from both sides of the relation, yielding

$$0 \leq \sum_{0\leq\ell<\texttt{index}(\iota_{i,m_i-1})} \left(WC_{\texttt{task}[\ell]} - BC_{\texttt{task}[\ell]}\right).$$

Given $BC_i \leq WC_i$, the result follows. □

Theorem 9. *The multi-rate AFAP cyclic executive dominates the multi-rate time-driven AFAP cyclic executive.*

Proof. Follows immediately from Lemma 4 and system $\mathcal{S}_3$. □

6.2.3 Bounds on fraction of gain time

Finally, we consider bounds on the fraction of time that can be gained for background processing by a multi-rate time-driven AFAP cyclic executive compared to a multi-rate AFAP cyclic executive.

Lemma 5. *Whenever a system $\mathcal{S}$ with a sequence of tasks* `task[]` *is schedulable under a multi-rate time-driven AFAP cyclic executive with a cycle time of $T^{\mathcal{S}}$, a fraction $f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}})$ given by*

$$\frac{\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} m_i \times WC_i\right)}{T^{\mathcal{S}}} \leq f^{\mathcal{S},\text{gain}}(T^{\mathcal{S}}) \leq \frac{\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} m_i \times BC_i\right)}{T^{\mathcal{S}}}$$

of the time can be gained for background processing by a multi-rate time-driven AFAP cyclic executive compared to a multi-rate AFAP cyclic executive.

Proof. The term $\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} m_i \times WC_i\right)$ represents the worst-case (i.e. minimum) amount of spare time in a cycle and the term $\left(T^{\mathcal{S}} - \sum_{1 \leq i \leq n} m_i \times BC_i\right)$ represents the best-case (i.e. maximum) amount of spare time. The result therefore follows immediately. □

6.3 Multi-rate periodic cyclic executive

6.3.1 Exact worst-case analysis

By removing the start jitter through the usage of a (strict) multi-rate periodic cyclic executive, we can adapt Theorem 8:

Theorem 10. *A system $\mathcal{S}$ will meet all its worst-case deadlines under a (strict) multi-rate periodic cyclic executive scheduling a sequence of tasks* `task[]` *iff the following condition holds*

$$\begin{aligned}
&\underset{1 \leq i \leq n}{\forall} \max_{0 \leq k < m_i - 1} \sum_{\texttt{index}(\iota_{i,k}) \leq \ell \leq \texttt{index}(\iota_{i,k+1})} WC_{\texttt{task}[\ell]} \leq WD_i^{\mathcal{S}} \\
\wedge \underset{T^{\mathcal{S}}}{\exists} &\sum_{1 \leq i \leq n} m_i \times WC_i \leq T^{\mathcal{S}} \leq \min_{1 \leq i \leq n} \left\{ WD_i^{\mathcal{S}} - \right. \\
&\left. \left(\sum_{0 \leq \ell \leq \texttt{index}(\iota_{i,0})} WC_{\texttt{task}[\ell]} - \sum_{0 \leq \ell \leq \texttt{index}(\iota_{i,m_i-1})-1} WC_{\texttt{task}[\ell]} \right) \right\}. \quad (29)
\end{aligned}$$

Proof. Follows immediately from Equation (18) and by simply replacing BC_i by WC_i in Equation (27). □

Based on Theorem 10, we can make the following derivation for system $\mathcal{S}_3$. Because the first clause of Equation (29) is identical to the first clause of Equation (27), we already know that this clause is satisfied. Next, we need to check the second claues, and find $12 \leq T^{\mathcal{S}_3} \leq \min\{11-(3-5), 14-(5-3), 17-(12-8)\} = \min\{13, 12, 13\} = 12$. We therefore conclude that system $\mathcal{S}_3$ under a multi-rate periodic cyclic executive. Figure 24 shows a worst-case execution for system $\mathcal{S}_3$ for a cycle time $T^{\mathcal{S}_3} = 12$.

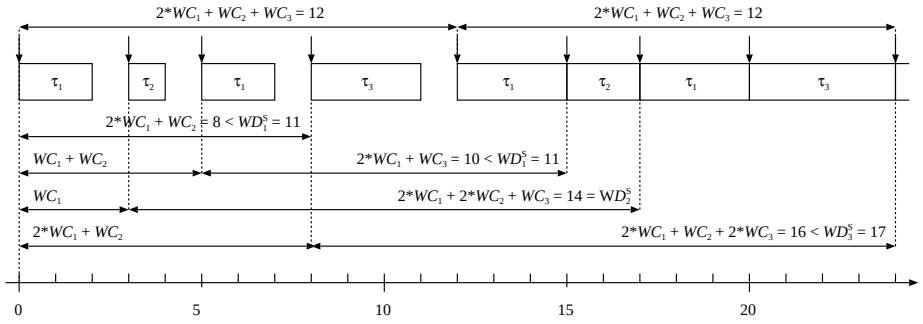

Figure 24: A situation of worst-case execution for system $\mathcal{S}_3$ under a (basic strict) multi-rate periodic cyclic executive using the cycle {`1, 2, 1, 3`} and a cycle time $T^{\mathcal{S}_3} = 12$. All system deadlines of $\mathcal{S}_3$ are met.

We observe that because system deadline $WD_2^{\mathcal{S}_3}$ is just met, as also became clear from applying Theorem 10, it is not possible to increase the cycle time. We further observe that task τ_1 is not started strictly periodically, i.e. $s_{1,1} - s_{1,0} = 5 < s_{1,2} - s_{1,1} = 9$.

6.3.2 Comparison with AFAP cyclic executives

Theorem 11. *The multi-rate AFAP cyclic executive and the multi-rate periodic cyclic executive are equivalent, i.e. can schedule the same systems.*

Proof. We prove the theorem by showing that by taking $T^{\mathcal{S}}$ to be equal to the lower bound in Equation (29), Equation (29) and Equation (23) are equivalent. We first observe that the first clause of Equation (29) is identical to the first part of Equation (23). Next, we rewrite the second clause of Equation (29) to

$$\underset{1\leq i\leq n}{\forall} \quad \underset{T^{\mathcal{S}}}{\exists} \quad \sum_{1\leq j\leq n} m_j \times WC_j \leq T^{\mathcal{S}} \leq WD_i^{\mathcal{S}} - \left(\sum_{0\leq \ell\leq \texttt{index}(\iota_{i,0})} WC_{\texttt{task}[\ell]} - \sum_{0\leq \ell\leq \texttt{index}(\iota_{i,m_i-1})-1} WC_{\texttt{task}[\ell]} \right).$$

We now derive for $T^{\mathcal{S}} = \sum\limits_{1 \le j \le n} m_j \times WC_j$

$$\begin{aligned} & \forall\limits_{1 \le i \le n} \quad \sum_{1 \le j \le n} m_j \times WC_j \le WD_i^{\mathcal{S}} - \\ & \qquad \left(\sum_{0 \le \ell \le \mathtt{index}(\iota_{i,0})} WC_{\mathtt{task}[\ell]} - \sum_{0 \le \ell \le \mathtt{index}(\iota_{i,m_i-1})-1} WC_{\mathtt{task}[\ell]} \right) \\ \Leftrightarrow \quad & \forall\limits_{1 \le i \le n} \quad \sum_{\substack{\mathtt{index}(\iota_{i,m_i-1}) \le \ell \le N-1 \ \vee \\ 0 \le \ell \le \mathtt{index}(\iota_{i,0})}} WC_{\mathtt{task}[\ell]} \le WD_i^{\mathcal{S}}, \end{aligned}$$

where the latter is identical to the second part of Equation (23) of Theorem 7. □

Theorem 12. *The multi-rate periodic cyclic executive dominates the multi-rate time-driven AFAP cyclic executive.*

Proof. Follows immediately from Theorems 9 and 11. □

6.3.3 Bounds on fraction of gain time

Lemma 6. *Whenever a system $\mathcal{S}$ with a sequence of tasks* `task[]` *is schedulable under a multi-rate periodic cyclic executive with a cycle time of $T^{\mathcal{S}}$, a fraction $f^{\mathcal{S},\mathrm{gain}}(T^{\mathcal{S}})$ given by*

$$\frac{\left(T^{\mathcal{S}} - \sum_{1 \le i \le n} m_i \times WC_i\right)}{T^{\mathcal{S}}} \le f^{\mathcal{S},\mathrm{gain}}(T^{\mathcal{S}}) \le \frac{\left(T^{\mathcal{S}} - \sum_{1 \le i \le n} m_i \times BC_i\right)}{T^{\mathcal{S}}}$$

of the time can be gained for background processing by a multi-rate periodic cyclic executive compared to a multi-rate AFAP cyclic executive.

Proof. Similar to the proof of Lemma 5. □

In case of system $\mathcal{S}_3$ and $T^{\mathcal{S}_3} = 12$, we find $0 \le f^{\mathcal{S}_3,\mathrm{gain}}(12) \le \frac{1}{3}$.

7 Discussion

In this section, we put the contents of this document in perspective by discussing various related issues.

7.1 Polling tasks and task types

For real-time systems, various types of tasks are typically distinguished based on their activation pattern or constraints on the inter-arrival time of jobs. Well-known examples include *periodic* tasks, with a fixed inter-arrival time of jobs, *sporadic* tasks [12], with a minimal inter-arrival time of jobs, and *elastic* tasks [13], with both a minimal and a maximal inter-arrival time of jobs.

The constraint expressed by Equation (7) for a polling task does not specifically correspond to any of these basic task types, because the period T_i is essentially a *maximal* inter-arrival time of jobs.

We observe that although tasks scheduled by a single- or multi-rate AFAP cyclic executive may experience drift, and therefore behave as elastic tasks, this behavior is an *emergent* property of the system rather than a characteristic of the task as determined by a designer.

7.2 A refinement of the task model

In Section 2, it is implicitly assumed that:

- tasks provide a response just before their completion;
- best-case computation times are a proper lower bound for the amount of time required by tasks when they handle events.

Both assumptions need not hold, however.

When providing the response is *not* the last action of a task, e.g. because there may be a number of internal actions after the task provided the response[5], Equations (8) and (13) are *pessimistic*[6]. In that case, we may like to refine our model by explicitly considering a finalization time f_i^r, a worst-case computation time $WC_i^r \leq WC_i$, a worst-case response time $WR_i^r \leq WR_i$, and a worst-case task deadline[7] WD_i^r, denoting the worst-case values for the case where the response is given earlier. This would yield refined worst-case schedulability conditions for $\mathcal{S}$, i.e. for periodic polling tasks

$$\bigvee_{1 \leq i \leq n} T_i + WD_i \leq WD_i^{\mathcal{S}} \wedge WR_i^r \leq WD_i^r,$$

and for regularly polling tasks

$$\bigvee_{1 \leq i \leq n} \left(\bigvee_{0 \leq k} f_{i,k+1}^r - s_{i,k} \leq WD_i^{\mathcal{S}} \right).$$

The best-case computation time BC_i is probably assumed when the event e_i does *not* occur. Hence, we may like to refine our model by explicitly considering a best-case computation time $BC_i^{e,r} \geq BC_i$ and a best-case response time $BR_i^{e,r} \geq BR_i$ denoting the best-case values for the case where (*i*) the event occurred and (*ii*) the minimal computation time for internal actions after the task provided the response are taken into account. Whereas ignoring the former may make Equations (10) and (14) *pessimistic* ignoring the latter may make them *optimistic*[8]. These two additional assumptions would yield refined best-case schedulability conditions for $\mathcal{S}$, i.e. for periodic polling tasks

$$\bigvee_{1 \leq i \leq n} BC_i^{e,r} \geq BD_i^{e,r} \geq BD_i^{\mathcal{S}}.$$

and for regularly polling tasks

$$\bigvee_{1 \leq i \leq n} BC_i^{e,r} \geq BD_i^{\mathcal{S}}.$$

[5] In [14], the notion of *observable event of a task* is introduced. The last observable event of a task (in our case a response) may not be the end of a task's execution.

[6] Schedulability analysis is *pessimistic* when it deems a system unschedulable, whereas the system is actually schedulable.

[7] In [3], such a deadline is termed an *internal deadline*.

[8] Schedulability analysis is *optimistic* when it deems a system schedulable, whereas the system is actually unschedulable.

We merely observe that the worst-case analysis of time-driven AFAP cyclic executives need not be adapted based on these new insights for best-case computation times, despite the fact that the analysis for those executives also depends on best-case computation times.

7.3 Equivalence and dominance of cyclic executives

In Theorems 5 and 11 we proved that for single-rate as well as multi-rate the AFAP and periodic cyclic executives are equivalent, i.e. each pair of cyclic executives can schedule the same systems. The advantages of a periodic cyclic executive over an AFAP cyclic executive is that the former prevents *drift* and may improve *energy efficiency*, as explained in Section 4. Moreover, a multi-rate cyclic executive may schedule systems that can not be scheduled by a single-rate cyclic executive, as illustrated by means of system $\mathcal{S}_3$. Because the single-rate cyclic executives are specializations of the multi-rate cyclic executives, we conclude that multi-rate cyclic executives dominate single-rate cyclic executives.

7.4 A note on task ordering

Whereas the order of tasks in the sequence to be executed in a cycle for the single-rate AFAP and single-rate periodic cyclic executives is immaterial, i.e. Theorems 1 and 4 are independent of the order of the tasks, this order has an influence on the schedulability for a system for the single-rate time-driven AFAP cyclic executive.

We illustrate the latter for an example system $\mathcal{S}_5$. We start by rewriting Theorem 2, using `task[]` and `index()`, to support an arbitrary, but fixed, order of tasks in a cycle.

Lemma 7. *A system $\mathcal{S}$ with a sequence of tasks* `task[]`*, that contains every task of the set $\mathcal{T}^{\mathcal{S}}$ exactly once, is schedulable by a single-rate time-driven AFAP cyclic executive iff the following condition holds*

$$\exists_{T^{\mathcal{S}}} \sum_{1 \le i \le n} WC_i \le T^{\mathcal{S}} \le \min_{1 \le i \le n} \left\{ WD_i^{\mathcal{S}} - \left(\sum_{0 \le k < \texttt{index}(\iota_{i,0})} \left(WC_{\texttt{task[}k\texttt{]}} - BC_{\texttt{task[}k\texttt{]}} \right) + WC_i \right) \right\}. \quad (30)$$

Proof. Follows immediately from Theorem 2 and the definitions of `task[]` and `index()`. □

The task characteristics of $\mathcal{T}_5^{\mathcal{S}}$ and associated system deadlines of $\mathcal{S}_5$ are given in Table 5. Based on Lemma 7, we conclude that $\mathcal{S}_5$ is not schedulable for `task[] = {`τ_1, τ_2`}` under a single-rate time-driven AFAP cyclic executive, because there doesn't exist a cycle time $T^{\mathcal{S}_5}$ that satisfies the relation $11 \le T^{\mathcal{S}_5} \le \min\{16-4, 18-(4-3+7)\} = 10$. For `task[] = {`$\tau_2, \tau_1$`}`, however, we find $11 \le T^{\mathcal{S}_5} \le \min\{18-7, 16-(7-6+4)\} = 11$, making $\mathcal{S}_5$ schedulable.

We define an *optimal task ordering* for a set $\mathcal{T}^{\mathcal{S}}$ of tasks of a system $\mathcal{S}$ scheduled by means of a single-rate time-driven cyclic executive as an ordering that will make $\mathcal{S}$ schedulable, if such an ordering exists. We leave the derivation of

Table 5: Task characteristics of task set $\mathcal{T}^{\mathcal{S}_5}$ and associated system deadlines of system $\mathcal{S}_5$.

	BC	WC	$WD^{\mathcal{S}_5}$
τ_1	4	3	16
τ_2	7	6	18

an optimal task ordering algorithm for single-rate time-driven cyclic executives as future work.

For multi-rate cyclic executives, the task ordering will in general have an influence on schedulability. We also leave optimal task ordering algorithms for multi-rate cyclic executives as future work.

7.5 A simple approach to determine schedulability

Given the fact that multi-rate cyclic executives dominate single-rate cyclic executives, a simple approach to determine schedulability of a system by a basic periodic cyclic executive is as follows:

1. Check whether or not a system $\mathcal{S}$ is schedulable by a single-rate AFAP using Theorem 1.
2. If it is, it can scheduled by a single-rate period cyclic executive with a maximum cycle time $T^{\mathcal{S}}$ derived by means of Theorem 4.
3. If it is not, search for an appropriate sequence with multiple jobs for tasks making the system schedulable with a muti-rate AFAP using Theorem 7.
4. If a sequence can be found, it can scheduled by a multi-rate period cyclic executive with a maximum cycle time $T^{\mathcal{S}}$ derived by means of Theorem 10.
5. Otherwise, this simple approach fails and other more advanced approaches are required, such as, for example, splitting tasks in sub-tasks or using other types of schedulers, as already concluded by C. Douglass Locke [2]. Further elaboration falls outside the scope of this document, however.

7.6 A note on 'advanced strict' cyclic executives

In this document, we assumed 'basic strict' cyclic executives, where every job is started at the worst-case start-time relative to the start of the cycle, assuming each job needs its worst-case execution time. When we lift this assumption, we may increase the cycle time, as illustrated by the following example, which only differs with respect to system $\mathcal{S}_3$ for $D_1^{\mathcal{S}_6}$ and $D_2^{\mathcal{S}_6}$ (i.e. $D_1^{\mathcal{S}_3} = 11$ and $D_2^{\mathcal{S}_3} = 14$).

As illustrated in Figure 25, all system deadlines are met when the task set $\mathcal{T}^{\mathcal{S}_6}$ is scheduled by means of a 'basic strict' multi-rate periodic cyclic executive. Applying Theorem 10, we find $12 \leq T^{\mathcal{S}_6} \leq \min\{10-(3-5), 15-(5-3), 17-(12-8)\} = \min\{12, 13, 13\} = 12$. Hence, $T^{\mathcal{S}_6}$ cannot be increased for a 'basic strict' version of the multi-rate periodic cyclic executive. Observe that task τ_1

Table 6: Task characteristics of task set $\mathcal{T}^{\mathcal{S}_6}$ and associated system deadlines of system $\mathcal{S}_6$. System $\mathcal{S}_6$ only differs from $\mathcal{S}_4$ for the values given in **bold**.

	BC	WC	$WD^{\mathcal{S}_6}$
τ_1	2	3	**10**
τ_2	1	2	**15**
τ_3	3	4	17

is not started strictly periodically, i.e. $s_{1,1} - s_{1,0} = 5$, whereas $s_{1,2} - s_{1,1} = 7$. Using Equation (4) and assuming $T_1 = T^{\mathcal{S}}/2$, we find $SJ_1 = 1$.

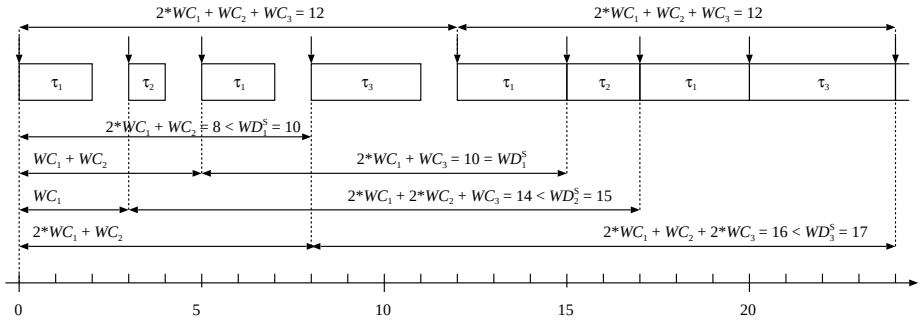

Figure 25: A situation of worst-case execution for system $\mathcal{S}_6$ under a (basic strict) multi-rate periodic cyclic executive using the cycle {`1, 2, 1, 3`} and a cycle time $T^{\mathcal{S}_6} = 12$. All system deadlines of $\mathcal{S}_6$ are met.

For an 'advanced strict version', we may *delay* the activation of job $\iota_{1,1}$ with 1 time unit in the cycle and increase $T^{\mathcal{S}_6}$ to 13, without causing deadline misses; see Figure 26. Hence, the advanced strict case allows a further increase of the fraction of gain time. We make three observations. Firstly, it is not possible to further increase $T^{\mathcal{S}_6}$, because each of the system deadlines are just met. Secondly, task τ_1 is still not started strictly periodically, i.e. $s_{1,1} - s_{1,0} = 6$, whereas $s_{1,2} - s_{1,1} = 7$. Using Equation (4) and assuming $T_1 = T^{\mathcal{S}}/2$, we find $SJ_1 = \frac{1}{2}$. Thirdly, it is not immediately clear how the periods and deadlines of the tasks in $T^{\mathcal{S}_6}$ shall be chosen to construct a cyclic executive as described in [1], where the minor cycle and a major cycle are taken to be the GCD and LCM of the periods of the tasks, respectively, all tasks meet their deadlines and Equation (11) is satisfied. We leave further elaboration as future work.

7.7 Other advanced approaches

In this document, we exclusively looked at a single processor, basic (strict) cyclic executives, and independent (and non-preemptive) polling tasks. Advanced approaches lifting our assumptions can be found in the literature. Without further elaboration, we merely observe that by assuming non-strict single-rate and multi-rate periodic cyclic executives, there is again a need for best-case computation times of tasks, similar to single-rate and multi-rate time-driven AFAP cyclic executives.

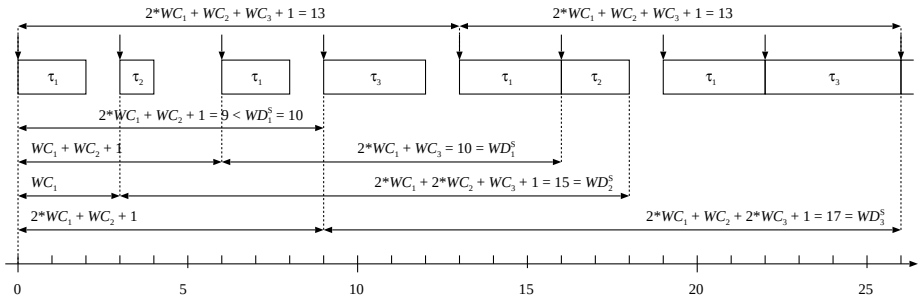

Figure 26: A situation of worst-case execution for system $\mathcal{S}_6$ under an advanced strict multi-rate periodic cyclic executive using the cycle {1, 2, 1, 3} and a cycle time $T^{\mathcal{S}_6} = 13$. Note that 1 unit of idle time has been inserted before jobs $\iota_{1,1}$ and $\iota_{1,3}$. All system deadlines of $\mathcal{S}_6$ are met.

As mentioned in, for example, Section 4.1.2, the spare time in a cycle could be used to execute other tasks. Apart from simply allowing other tasks to execute in the background, it is also possible to adapt the static schedule at runtime to allow other ready tasks to run earlier. As an example, reconsider Figure 24. Whenever τ_2 does not experience a worst-case execution, τ_1 can be started *earlier*, i.e. immediately after task τ_2, because the second execution of τ_1 in the major cycle has 1 unit of time slack. Similarly, whenever task τ_1 does not experience a worst-case execution, τ_3 can be started *earlier*, but no earlier than at time 13 relative to the start of the major cycle. Hence, we may either immediately start τ_1 after τ_2 or immediately start τ_3 after τ_1.

An advanced approach called *slot shifting* was proposed by Gerhard Fohler [15, 16], to efficiently combine off-line scheduling of periodic tasks and online scheduling of aperiodic tasks for distributed systems and advanced task models including various kinds of constraints for the periodic tasks. The approach was later extended by Damir Isovic [17] by also considering sporadic tasks for online scheduling. It is interesting to see that within the context of, for example, Time Sensitive Networks (TSN), combining off-line (e.g. the IEEE 802.1Qbv Enhancements to Scheduled Traffic, i.e. the Time-Aware Shaper (TAS) [18]) and online (e.g. the IEEE 802.1Qav Forwarding and Queuing Enhancements for Time-Sensitive Networks, in particular Credit-Based Shaping (CBS) [19]) scheduling techniques is again a topic of (research) interest.

8 Conclusion

In this document, we considered scheduling of independent polling tasks on a single processor using basic cyclic executives. Unlike existing approaches that typically assume periods and deadlines for *tasks*, we focussed on whether or not the deadlines of the *system* are met. A major advantage of our approach is that there is no need to decide upon the specific periods and deadlines of the tasks to determine schedulability of the system.

We addressed the following topics. Firstly, we presented exact analysis for basic (strict) cyclic executives. Because best-case analysis turned out to be trivial, focus of this document has been on exact worst-case analysis (Theorems 1,

2, 4, 7, 8, and 10). Secondly, we showed that for the exact schedulability test for single-rate and multi-rate time-driven AFAP cyclic executives we need both worst-case and best-case computation times of tasks (see Theorems 2 and 8). Thirdly, we presented methods to determine the maximum cycle time for

- single-rate and multi-rate time-driven AFAP cyclic executives (see Theorems 2 and 8);
- (strict) single-rate and multi-rate periodic cyclic executives (see Theorems 4 and 10).

Fourthly, we compared cyclic executives (see Theorems 3, 5, 6, 9, 11, and 12), and concluded that the (strict) multi-rate periodic cyclic executive dominates all other cyclic executives considered. Finally, we presented methods to determine bounds on the fraction of time that can be gained for background processing by time-driven AFAP and periodic cyclic executives compared to the AFAP versions (see Lemmas 2, 3, 5, and 6).

We put the contents of this document in perspective by discussing various related issues in Section 7.

Acknowledgements

We thank Pieter J.L. Cuijpers and Mitra Nasri Nasrabadi for their constructive comment on a previous version of this document.

References

[1] G. Buttazzo, *Hard real-time computing systems - predictable scheduling algorithms and applications (*3^{rd} *edition)*. Springer, 2011.

[2] C. D. Locke, "Software architecture for hard real-time applications: Cyclic executives vs. fixed priority executives," *Real-Time Systems*, vol. 4, no. 1, pp. 37–53, March 1992.

[3] A. Burns, "Preemptive priority based scheduling: An appropriate engineering approach," in *Advances in Real-Time Systems*, S. Son, Ed. Prentice-Hall, 1994, pp. 225–248.

[4] J. Liu, *Real-Time Systems*. Prentice Hall, 2000.

[5] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages (*4^{th} *edition)*. Addison-Wesley, 2009.

[6] H. Kopetz, *Real-time systems - Design Principles for Distributed Embedded Applications (*2^{nd} *edition)*. Springer, 2011.

[7] B. Åkesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. Davis, "A Survey of Industry Practice in Real-time Systems," in *Proc.* 31^{st} *IEEE Real-Time Systems Symposium (RTSS)*, December 2020.

[8] T. Baker and A. Shaw, "The cyclic executive model and Ada," *Real-Time Systems*, vol. 1, no. 1, pp. 7–25, June 1989.

[9] D. Kalinsky, "A Survey of Task Schedulers," March 2016, last visited: July 2020. [Online]. Available: http://www.kalinskyassociates.com/Wpaper3.html

[10] H. Rivera-Verduzco and R. Bril, "Towards best-case response times of real-time tasks under fixed-priority scheduling with preemption thresholds," in *Proc. 22^{nd} IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2017.

[11] "Patriot Missile Defense - Software Problem Led to System Failure at Dhahran, Saudi Arabia," GAO/IMTEC-92-26, February 1992. [Online]. Available: https://www.gao.gov/products/IMTEC-92-26

[12] A.-L. Mok, "Fundamental design problems of distributed systems for the hard-real-time environment," Ph.D. dissertation, Massachusetts Institute of Technology, May 1983. [Online]. Available: http://www.lcs.mit.edu/publications/pubs/pdf/MIT-LCS-TR-297.pdf

[13] G. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Proc. 19^{th} IEEE Real-Time Systems Symposium (RTSS)*, December 1998, pp. 286–295.

[14] R. Gerber and S. Hong, "Semantics-based compiler transformations for enhanced schedulability," in *Proc. 14^{th} IEEE Real-Time Systems Symposium (RTSS)*, December 1993, pp. 232–242.

[15] G. J. Fohler, "Flexibility in statically scheduled hard real-time systems," Ph.D. dissertation, Technischen Universität Wien, April 1994.

[16] G. Fohler, "Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems," in *Proc. 16^{th} Real-Time Systems Symposium (RTSS)*, December 1995, pp. 152–161.

[17] D. Isović and G. Fohler, "Efficient scheduling of sporadic, aperiodic, and periodic tasks with complex constraints," in *Proc. 21^{st} IEEE Real-Time Systems Symposium (RTSS)*, November 2000, pp. 207–216.

[18] IEEE, "IEEE 802.1 Qbv - IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) bridges and Virtual Bridged Local Area Networks Amendment: Enhancements for Scheduled Traffic," Institute of Electrical and Electronics Engineers, New York, NY, Tech. Rep., March 2016.

[19] ——, "IEEE 802.1 Qav - IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) bridges and Virtual Bridged Local Area Networks Amendment: Forwarding and Queuing Enhancements for Time Sensitive Networks," Institute of Electrical and Electronics Engineers, New York, NY, Tech. Rep., April 2010.

Glossary

best-case computation time A lower bound on the computation time of any job of a task. 3, 5, 7, 9, 11, 34, 37, 38

best-case response time The infimum of the response times of the jobs of a task. 5, 9, 34

best-case system deadline A lower bound on the time-interval from the occurrence of an (input) event of a system to its (output) response. 3, 5–7

best-case task deadline A lower bound on the response time of all jobs of a task. 6, 9

drift unbounded jitter. 12, 13, 15, 33, 34

gain time the amount of time that has been allocated to tasks, but is not used, i.e. when tasks execute for less than their worst-case computation time. 14, 23, 30, 33, 36

jitter fluctuations in the inter-arrival time of certain timing events, such as arrival times, start times or finalization times of tasks. 41

job the execution of a task. 5

maximal inter-arrival time An upper bound on the time between consecutive arrivals. 33

minimal inter-arrival time A lower bound on the time between consecutive arrivals. 3, 5, 7, 33

slack the amount of time that has not been allocated to tasks. 14, 23, 38

spare time the combination of slack and gain time. 14, 22, 23, 26, 31, 37

start jitter fluctuations in the interval between start times of subsequent jobs of a task. 7, 13, 14, 16, 24, 31

worst-case computation time An upper bound on the computation time of any job of a task. 3, 5, 7, 10, 13, 16, 27, 28, 34, 41

worst-case response time The supremum of the response times of the jobs of a task. 5, 7, 8, 34

worst-case system deadline An upper bound on the time-interval from the occurrence of an (input) event of a system to its (output) response. 3, 5–7

worst-case task deadline An upper bound on the response time of all jobs of a task. 6, 8, 34

Acronyms

AFAP as fast as possible. 4

GCD Greatest Common Divider. 10, 37

LCM Least Common Multiple. 10, 37

Science Reports **Department of Mathematics and Computer Science**
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: a.gouma@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2015):

15/01	Önder Babur, Tom Verhoeff and Mark van den Brand	Multiphysics and Multiscale Software Frameworks: An Annotated Bibliography
15/02	Various	Proceedings of the First International Workshop on Investigating Dataflow In Embedded computing Architectures (IDEA 2015)
15/03	Hrishikesh Salunkhe, Alok Lele, Orlando Moreira and Kees van Berkel	Buffer Allocation for Realtime Streaming Applications Running on a Multi-processor without Back-pressure
15/04	J.G.M. Mengerink, R.R.H. Schiffelers, A. Serebrenik, M.G.J. van den Brand	Evolution Specification Evaluation in Industrial MDSE Ecosystems
15/05	Sarmen Keshishzadeh and Jan Friso Groote	Exact Real Arithmetic with Pertubation Analysis and Proof of Correctness
15/06	Jan Friso Groote and Anton Wijs	An $O(m \log n)$ Algorithm for Stuttering Equivalence and Branching Bisimulation
17/01	Ammar Osaiweran, Jelena Marincic Jan Friso Groote	Assessing the quality of tabular state machines through metrics
17/02	J.F. Groote and e.P. de Vink	Problem solving using process algebra considered insightful
18/01	L. Sanchez, W. Wesselink and Algorithm, T.A.C. Willemse	BDD-Based Parity Game Solving: A comparison of Zielonka's Recursive Priority Promotion and Fixpoint Iteration
18/02	Mahmoud Talebi Rates	First-order Closure Approximations for Middle-Sized Systems with Non-linear
18/03	Thomas Neele, Tim A.C. Willemse and Jan Friso Groote	Solving Parameterised Boolean Equation Systems with Infinite Data Through Quotienting (Technical Report)
18/04	Daan Leermakers, Behrooz Razeghi, Shideh Rezaeifar, Boris Škorić, Olga Taran Slava Voloshynovskiy	Optical PUF statistics
19/01	Maurice Laveaux, Jan Friso Groote and Tim A.C. Willemse	Correct and Efficient Antichain Algorithms for Refinement Checking
19/02	Thomas Neel, Tim A.C. Willemse and Wieger Wesselink	Partial-Order Reduction for Parity Games with an Application on Parameterised Boolean Equation Systems (Technical Report)
19/03	David N. Jansen, Jan Friso Groote, Jeroen J.A. Keiren and Anton Wijs	A simpler $O(m \log n)$ algorithm for branching bisimilarity on labelled transition systems
19/04	Alexander Fedotov, Jeroen J.A. Keiren and Julien Schmaltz	Sound idle and block equations for finite state machines in xMAS
19/05	J.F. Groote, J.J.A. Keiren, B. Luttik, E.P. de Vink and T.A.C. Willemse	Modelling and Analysing Software in mCRL2
20/01	Behrooz Razeghi, Slava Voloshynovskiy, Marzieh Gheisari, Teddy Furon, Laurent Amsaleg, Daan Leermakers, Boris Skoric	Design of Practical IoT Identification Methods - Deliverable 2.2 Month 34 Nov 2019
20/02	Reinder J. Bril	Exact analysis for basic cyclic executives