Tighter schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks

DOI:
10.1109/ICECCS.2011.11

Document status and date:
Published: 01/01/2011

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

2011 16th IEEE International Conference on Engineering of Complex Computer Systems

Tighter schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks

Moris Behnam, Thomas Nolte
Mälardalen Real-Time Research Centre
P.O. Box 883, SE-721 23 Västerås, Sweden
moris.behnam@mdh.se

Reinder J. Bril
Technische Universiteit Eindhoven (TU/e)
Den Dolech 2, 5612 AZ Eindhoven
The Netherlands

***Abstract*—In this paper, we show that both global as well as local schedulability analysis of synchronization protocols based on the stack resource policy (SRP) and overrun without payback for hierarchical scheduling frameworks based on fixed-priority preemptive scheduling (FPPS) are pessimistic. We present tighter global and local schedulability analysis, illustrate the improvements of the new analysis by means of examples, and show that the improved global analysis is both uniform and sustainable. We evaluate the new global and local schedulability analysis based on an extensive simulation study and compare the results with the existing analysis.**

I. INTRODUCTION

Background: Over the years, there has been a growing attention for hierarchical scheduling of real-time systems due to its ability to provide temporal isolation between multiple real-time subsystems executing upon a common processing platform. The Hierarchical Scheduling Framework (HSF) provides means for decomposing a complex system into well-defined parts called subsystems, and a subsystem provides an introspective interface that specifies the timing properties of the subsystem precisely. This implies that subsystems can be independently developed, analyzed and tested, and later assembled without introducing unwanted temporal interference.

Supporting global resource sharing between subsystems is a major challenge, since it increases the complexity of the analysis of a system considerably. Due to this complexity, most of the proposed techniques are based on some simplifying assumptions which make the analysis easier, e.g., [1], [2]. The consequence of these assumptions is that they add pessimism in the analysis which increases the required CPU resources of systems. For some systems, the pessimism in the analysis is not significant and can be ignored, but for others it may be significant.

As large extents of embedded systems are resource constrained, a tight analysis is instrumental in a successful deployment of HSF techniques in real applications. We therefore aim at reducing potential pessimism in existing schedulability analysis for HSFs that support sharing of global shared resources. Looking further at existing industrial real-time systems, fixed priority preemptive scheduling (FPPS) is the de facto standard of task scheduling, hence we focus on an HSF with support for FPPS for tasks within a subsystem. Having such support will simplify migration to and integration of existing legacy applications into the HSF, avoiding a too big technology revolution for engineers.

Our current research efforts are directed towards the conception and realization of a two-level HSF that is based on (*i*) FPPS for both *global scheduling* of budgets (allocated to subsystems) and *local scheduling* of tasks (within a subsystem), (*ii*) the periodic resource model [3] for budgets, and (*iii*) the Stack Resource Policy (SRP) [4] for both inter- and intra-subsystem resource sharing. For such an HSF, two mechanisms have been studied that prevent depletion of a budget during global resource access, i.e. *skipping* [1] and *overrun* [2]. Note that, budget depletion during global resource access may cause tasks from other subsystems missing their deadline. The overrun mechanism comes in two flavors, i.e. *with payback* and *without payback*.

In this paper, we aim at tighter analysis for the overrun mechanism without payback, assuming the same introspective interface for subsystems as the existing analysis.

Contributions: We show that existing global and local schedulability analysis of synchronization protocols based on SRP and overrun without payback for two-level hierarchical scheduling based on FPPS is pessimistic. We present tighter global and local analysis assuming that the deadline of a subsystem holds for the sum of its normal budget and its overrun budget, and illustrate the improvements by means of examples. We identify the system parameters that have a great effect on the improvement of the proposed global and local analysis. In addition we evaluate the improvements that both global and local new analysis can achieve compared with the traditional analysis, in terms of CPU resources, by exploring the system load [2] in a simulation study.

Overview: This paper has the following structure. In Section II we present related work. A real-time scheduling model is the topic of Section III. The existing global and local schedulability analysis is recapitulated in Section IV, and tighter global and local analysis is presented in Sections V and VI, respectively. Section VII presents a simulation study evaluating the improvement that both global and local new analysis can achieve. The paper is concluded in Section VIII.

The work in this paper is supported by the Swedish Foundation for Strategic Research (SSF), via the research programme PROGRESS.

978-0-7695-4381-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ICECCS.2011.11

II. Related work

During the past decade, there has been considerable interest on hierarchical scheduling of real-time systems [5], [6], [7], [3]. Deng and Liu [5] proposed a two-level HSF for open systems, where subsystems may be developed and validated independently. Kuo and Li [6] and Lipari and Baruah [7] presented schedulability analysis techniques for such two-level frameworks with FPPS and Earliest Deadline First (EDF) global schedulers, respectively. Shin and Lee [3] proposed the periodic resource model $\Gamma(\Pi, \Theta)$ to specify guaranteed periodic CPU allocations, where $\Pi \in \mathbb{R}^+$ is a period and $\Theta \in \mathbb{R}^+$ is a periodic allocation time ($0 < \Theta \leq \Pi$). Easwaran, and Lee [8] proposed the explicit deadline periodic (EDP) resource model $\Omega(\Pi, \Theta, \Delta)$ that extends the periodic resource model by explicitly distinguishing a relative deadline $\Delta \in \mathbb{R}^+$ for the allocation time Θ ($0 < \Theta \leq \Delta \leq \Pi$).

For synchronization protocols in HSFs, two mechanisms have been studied to prevent depletion of a budget during global resource access, i.e. *skipping* and *overrun* (*with payback* and *without payback*). The idea of skipping in the context of HSFs, was used by the SIRAP protocol [1], and its associated analysis supports composability. It works as follows: when a job tries to access a global shared resource, it will be granted the access to the resource if the remaining subsystem budget is enough to lock and release the global resource before budget depletion. Otherwise, the access to the shared resource will be delayed until the next activation period. Overrun with payback was first introduced in the context of aperiodic servers in [9]. The mechanism was later (re-) used for a synchronization protocol in the context of two-level hierarchical scheduling in [10] and extended with overrun without payback. Overrun mechanism works as follows: when the budget of a subsystem depletes and an internal job has not released the lock of a global shared resource, the subsystem overruns its budget and the job continues its execution until it releases the locked resource. This mechanism is called overrun with payback if the subsystem budget is decreased by the amount of the overrun in the next activation period followed by an overrun, otherwise it is called overrun without payback. The analysis presented in [10] does not support independent subsystems development, i.e., the parameters of the other subsystems should be available in order to perform the analysis of each subsystem. However an analysis supporting composability was described in [2], [11].

In this paper we present tighter global and local analysis for overrun without payback and we evaluate, by means of simulation study, the improvements that the new tighter analysis can achieve compared with the traditional analysis, in terms of CPU resources.

III. Real-time scheduling model

We consider a two-level hierarchical FPPS model using the periodic resource model to specify guaranteed CPU allocations to tasks of subsystems and using a synchronization protocol for mutual exclusive resource access to global resources based on SRP[1] and overrun without payback.

System model: A system Sys contains a set $\mathcal{R}$ of M global logical resources $R_1, R_2, \ldots, R_M$, a set $\mathcal{S}$ of N subsystems $S_1, S_2, \ldots, S_N$, a set $\mathcal{B}$ of N budgets for which we assume a periodic resource model [3], and a single processor. Each subsystem S_s has a dedicated budget associated to it. In the remainder of this paper, we leave budgets implicit, i.e. the timing characteristics of budgets are taken care of in the description of subsystems. Subsystems are scheduled by means of FPPS and have fixed, unique priorities. For notational convenience, we assume that subsystems are given in order of decreasing priorities, i.e. S_1 has the highest priority and S_N has the lowest priority.

Subsystem model: Each subsystem S_s contains a set $\mathcal{T}_s$ of n_s periodic tasks $\tau_{s,1}, \tau_{s,2}, \ldots, \tau_{s,n_s}$ with fixed, unique priorities, which are scheduled by means of FPPS. For notational convenience, we assume that tasks are given in order of decreasing priorities, i.e. τ_1 has highest priority and τ_{n_s} has lowest priority. The set $\mathcal{R}_s$ denotes the subset of M_s global resources accessed by subsystem S_s. The maximum time that a subsystem S_s executes while accessing resource $R_l \in \mathcal{R}$ is denoted by X_{sl}, where $X_{sl} \in \mathbb{R}^+ \cup \{0\}$ and $X_{sl} > 0 \Leftrightarrow R_l \in \mathcal{R}_s$. The timing characteristics of S_s are specified by means of a triple $< P_s, Q_s, \mathcal{X}_s >$, where $P_s \in \mathbb{R}^+$ denotes its (budget) period, $Q_s \in \mathbb{R}^+$ its (normal) budget, and $\mathcal{X}_s$ the set of maximum execution access times of S_s to global resources. The maximum value in $\mathcal{X}_s$ (or zero when $\mathcal{X}_s$ is empty) is denoted by X_s.

Task model: The timing characteristics of a task $\tau_{si} \in \mathcal{T}_s$ are specified by means of a quartet $< T_{si}, C_{si}, D_{si}, \mathcal{C}_{si} >$, where $T_{si} \in \mathbb{R}^+$ denotes its minimum inter-arrival time, $C_{si} \in \mathbb{R}^+$ its worst-case computation time, $D_{si} \in \mathbb{R}^+$ its (relative) deadline, $\mathcal{C}_{si}$ a set of maximum execution times of τ_{si} to global resources, where $C_{si} \leq D_{si} \leq T_{si}$ and $P_s \leq T_{si}$ [2]. The set $\mathcal{R}_{si}$ denotes the subset of $\mathcal{R}_s$ accessed by task τ_{si}. The maximum time that a task τ_{si} executes while accessing resource $R_l \in \mathcal{R}$ is denoted by c_{sil}, where $c_{sil} \in \mathbb{R}^+ \cup \{0\}$, $C_{si} \geq c_{sil}$, and $c_{sil} > 0 \Leftrightarrow R_l \in \mathcal{R}_{si}$.

Resource model: The *CPU supply* refers to the amount of CPU allocation that a virtual processor can provide. The supply bound function $\mathtt{sbf}_\Omega(t)$ of the EDP resource model $\Omega(\Pi, \Theta, \Delta)$ that computes the minimum possible CPU supply for every interval length t is given by [3]

$$\mathtt{sbf}_\Omega(t) = \begin{cases} t - (k+1)(\Pi - \Theta) + (\Pi - \Delta) & \text{if } t \in V^{(k)} \\ (k-1)\Theta & \text{otherwise,} \end{cases} \tag{1}$$

[1]The focus of this paper is on synchronization protocols for *global* logical resources. We do therefore not consider local logical resources.

where $k = \max\left(\lceil\left(t-(\Delta-\Theta)\right)/\Pi\rceil, 1\right)$ and $V^{(k)}$ denotes an interval $[k\Pi + \Delta - 2\Theta, k\Pi + \Delta - \Theta]$.

The supply bound function $\mathtt{sbf}_\Gamma(t)$ of the periodic resource model $\Gamma(\Pi, \Theta)$ is a special case of (1), i.e. with $\Delta = \Pi$.

Synchronization protocol: Overrun without payback prevents depletion of a budget of a subsystem S_s during access to a global resource R_l by temporarily increasing the budget of S_s with X_{sl}, the maximum time that S_s executes while accessing R_l. To be able to use SRP in an HSF for synchronizing global resources, its associated ceiling terms needs to be extended.

Resource ceiling: With every global resource R_l, two types of resource ceilings are associated; an *external* resource ceiling RC_l for global scheduling and an *internal* resource ceiling rc_{sl} for local scheduling. According to SRP, these ceilings are defined as

$$RC_l = \min(N, \min\{s \mid X_{sl} > 0\}), \quad (2)$$

$$rc_{sl} = \min(n_s, \min\{i \mid c_{sil} > 0\}). \quad (3)$$

Note that we use the outermost $\min$ in (2) and (3) to define RC_l and rc_{sl} also in those situations where no subsystem uses R_l and no task of $\mathcal{T}_s$ uses R_l, respectively.

System/subsystem ceiling: The system/subsystem ceilings are dynamic parameters that change during the execution. The system/subsystem ceiling is equal to the lowest external/internal resource ceiling of a currently locked resource in the system/subsystem.

Under SRP, a task τ_{si} can only preempt the currently executing task τ_{sj} (even when accessing a global resource) if the priority of τ_{si} is greater (i.e. the index i is lower) than S_s its subsystem ceiling. A similar condition for preemption holds for subsystems.

Concluding remarks: The maximum time X_{sl} that S_s executes while accessing R_l can be reduced by assigning a value to rc_{sl} that is *smaller* than the value according to SRP. For HSRP [10], the internal resource ceiling is therefore set to the highest priority, i.e. $rc_{sl}^{\mathrm{HSRP}} = 1$. Decreasing rc_{sl} may cause a subsystem to become unfeasible for a given budget [12], however, because the tasks with a priority higher than the old ceiling and at most equal to the new ceiling may no longer be feasible.

The results in this paper apply for any internal resource ceiling rc'_{sl} where $rc_{sl} \geq rc'_{sl} \geq rc_{sl}^{\mathrm{HSRP}} = 1$.[2]

IV. Recap of existing schedulability analysis

In this section, we briefly recapitulate the global schedulability analysis presented in [10] and the local schedulability analysis described in [2], [11]. Although the global schedulability analysis presented in [2], [11] looks different, it is based on the analysis described in [10] and therefore yields the same result.

[2] Because $rc_{sl}^{\mathrm{HSRP}} = 1$ for $R_l \in \mathcal{R}_s$, $X_{sl} = \max_i c_{sil}$. Hence, from $c_{sil} < Q_s$ we derive $X_s < Q_s$.

For illustration purposes, we will use an example system Sys_{I} containing two subsystems S_1 and S_2 sharing a global resource R_1. The characteristics of the subsystems are given in Table I.

subsystem	P_s	$Q_s + X_s$
S_1	5	2
S_2	7	$Q_2 + X_2$

Table I
SUBSYSTEM CHARACTERISTICS OF Sys_{I}.

A. Global analysis

The worst-case response time WR_s of subsystem S_s is given by the smallest $x \in \mathbb{R}^+$ satisfying[3]

$$x = B_s + (Q_s + X_s) + \sum_{t<s} \left\lceil \frac{x}{P_t} \right\rceil (Q_t + X_t), \quad (4)$$

where B_s is the maximum blocking time of S_s by lower priority subsystems, i.e.

$$B_s = \max(0, \max\{X_{tl} \mid t > s \wedge X_{tl} > 0 \wedge RC_l \leq s\}). \quad (5)$$

Note that we use the outermost $\max$ in (5) to define B_s also in those situations where the set of values of the innermost $\max$ is empty. To calculate WR_s, we can use an iterative procedure based on recurrence relationships, starting with a lower bound, e.g. $B_s + \sum_{t \leq s}(Q_t + X_t)$. The condition for global schedulability is given by

$$\underset{1 \leq s \leq N}{\forall} WR_s \leq P_s. \quad (6)$$

We observe that the global analysis is similar to basic analysis for FPPS with resource sharing, where the period P_s of a subsystem S_s serves as deadline for the sum of the normal budget Q_s and the overrun budget X_s. Hence the interference of higher priority subsystems S_t is based on the sum $Q_t + X_t$. We will therefore use a superscript P to refer to this basic analysis for subsystems, e.g. WR_s^{P}.

In the sequel, we are not only interested in the worst-case response time of a subsystem S_s for particular values of B_s, Q_s, and X_s, but in the value as a function of the sum of these three values. We will therefore use a functional notation when needed, e.g. $WR_s(B_s + Q_s + X_s)$.

The global feasibility area of the existing analysis is illustrated for our example system Sys_{I} in Figure 1(a). Note that the y-axis is excluded, because we assume the capacity of subsystems to be positive, i.e. $Q_2 > 0$.

[3] Strictly spoken, [10] uses (4) *excluding* X_s for WR_s. The smallest positive solution of (4) is required to be at most equal to P_s to prevent additional interference of the next activation of (the budget of) S_s.

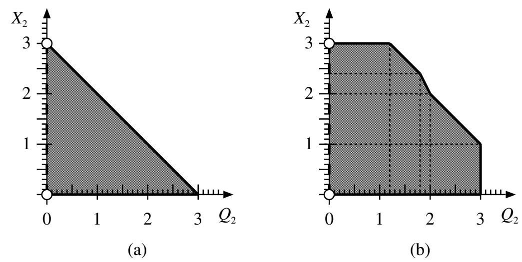

Figure 1. Global feasibility area assuming (a) FPPS and (b) tighter global analysis.

B. Local analysis

The existing condition for local schedulability of a subsystem S_s [2] is given by

$$\forall_{1\le i\le n_s} \exists_{0<x\le D_{si}} b_{si} + C_{si} + \sum_{j<i} \left\lceil \frac{x}{T_{sj}} \right\rceil \cdot C_{sj} \le \mathtt{sbf}_{\Gamma_s}(x), \tag{7}$$

where b_{si} is the maximum blocking time of τ_{si} by lower priority tasks, i.e.

$$b_{si} = \max(0, \max\{c_{sjl} \mid j > i \wedge c_{sjl} > 0 \wedge rc_{sl} \le i\}), \tag{8}$$

and $\mathtt{sbf}_{\Gamma_s}(x)$ is the supply bound function of the periodic resource model $\Gamma_s(P_s, Q_s)$ for the subsystem S_s under consideration. Note that we use the outermost max in (8) to define b_{si} also in those situations where the set of values of the innermost max is empty.

The value for X_{sl} depends on the local scheduler and the synchronization protocol. The maximum time that subsystem S_s executes while task τ_{si} accesses resource $R_l \in \mathcal{R}$ is denoted by X_{sil}, where $X_{sil} \in \mathbb{R}^+ \cup \{0\}$ and $X_{sil} > 0 \Leftrightarrow c_{sil} > 0$. For $c_{sil} > 0$, X_{sil} is given by [2]

$$X_{sil} = c_{sil} + \sum_{j<rc_{sl}} C_{sj}. \tag{9}$$

The value for X_{sl} is given by

$$X_{sl} = \max_{1\le i\le n_s} X_{sil}. \tag{10}$$

V. Tighter global analysis

As described in Section IV-A, the existing global schedulability analysis is based on FPPS, where the period P_s serves as deadline for the sum of the normal budget Q_s and overrun budget X_s.

A. Illustrating the improvement

The improvement of the global analysis is based on two observations:

1) *Limited pre-emption of overrun budget X_s*: while S_s is accessing R_l using X_s, it can only be pre-empted by subsystems with a priority higher than RC_l.
2) *Blocking starts* before *the execution based on the overrun budget X_s starts*: to use its overrun budget X_s, S_s needs to first lock a global resource.

From the first observation, we conclude that subsystem S_1 can not preempt S_2 during those intervals of time when S_2 is accessing resource R_1 in general, and when S_2 is executing based on its overrun budget X_2 in particular. This limited preempt-ability of subsystem S_2 gives rise to improved schedulability of S_2.

From the second observation, we conclude that whenever S_2 uses its overrun budget X_2, it must have locked R_1 already during the consumption of its normal budget Q_2, i.e. *before* it starts consuming its overrun budget X_2. Hence, the system ceiling is already set to the priority of S_1 before S_2 starts consuming X_2, preventing S_1 to preempt.

The resulting improvements is illustrated in Figure 1(b), which we briefly explain by means of an example.[4] Figure 2 shows a timeline with $Q_2 = 3.0$ and $X_2 = 1.0$, where the first job $\iota_{2,0}$ of S_2 locks R_1 just before the activation of S_1 at $t = 5$. Subsystem S_2 is therefore allowed to execute X_2 at $t = 5$, effectively *deferring* the execution of S_1. This has a number of consequences. Firstly, S_2 does not miss its deadline at time $t = 7$, as we would conclude from the existing analysis. Secondly, the worst-case response time of of S_2 is no longer assumed for $\iota_{2,0}$, activated at $t = 0$, but instead for $\iota_{2,1}$ activated at $t = 7$, because the deferred execution of S_1 gives rise to *additional* interference for $\iota_{2,1}$. Rather than having to consider only a single job to determine schedulability, we therefore have to consider all jobs in a so-called level-s active period, similar to the analysis for FPDS [14] and FPPS with preemption thresholds [15]. The level-2 active period starts at time $t = 0$, when both S_1 and S_2 become active, ends at time $t = 14$, when all pending work of S_1 and S_2 has been completed, and contains two jobs of S_2. Because both jobs meet their deadline, S_2 is schedulable.

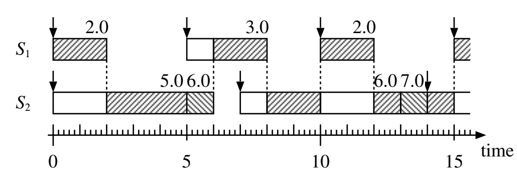

Figure 2. Timeline for $Q_2 = 3.0$ and $X_2 = 1.0$ assuming blocking starts before overrun.

B. Improving the global analysis

In this section, we first recapitulate the notion of a level-s active period. Next, we derive analysis for the worst-case finalization time WF^{Q}_{sk} of the normal budget Q_s of job ι_{sk} of subsystem S_s relative to start of the constituting level-s active period. Finally, we derive analysis for the worst-case response time WR_s of S_s.

[4]The interested reader is referred to [13] for a detailed explanation.

1) Level-s active period: The worst-case length WL_s of a level-s active period with $s \leq N$ is given by the smallest $x \in \mathbb{R}^+$ that satisfies

$$x = B_s + \sum_{t \leq s} \left\lceil \frac{x}{P_t} \right\rceil (Q_t + X_t). \tag{11}$$

To calculate WL_s, we can use an iterative procedure based on recurrence relationships, starting with a lower bound, e.g. $B_s + \sum_{t \leq s}(Q_t + X_t)$. The maximum number wl_s of jobs of S_s in a level-s active period is given by

$$wl_s = \left\lceil \frac{WL_s}{P_s} \right\rceil. \tag{12}$$

2) Worst-case finalization time: For a job ι_{sk} of S_s with $0 \leq k < wl_s$, we split the interval from the start of the level-s active period to the finalization of job ι_{sk} in two sub-intervals: a first sub-interval including the execution of the normal budget Q_s by job ι_{sk} and a second sub-interval from the finalization of Q_s by ι_{sk} till the finalization of ι_{sk}, i.e. including the execution of the overrun budget X_s.

Let WF^{Q}_{sk} denote the worst-case finalization time of the normal budget Q_s of job ι_{sk} with $0 \leq k < wl_s$ relative to the start of the constituting level-s active period. To determine WF^{Q}_{sk}, we have to consider up to three suprema. First, the sequence of jobs ι_{s0} till ι_{sk} experience a blocking $B_s \geq 0$ by lower priority subsystems in the worst-case situation. Similar to FPDS [14], the worst-case blocking is a supremum for $B_s > 0$ rather than a maximum. Second, the jobs ι_{s0} till $\iota_{s,k-1}$ need their overrun budget X_s to access global resources. Because the access to a global resource starts during the execution of the normal budget, the actual amount X of overrun budget used is a supremum rather than a maximum. Finally, the access to the global resource also starts "as late as possible" during the execution of job ι_{sk} in a worst-case situation, to maximize the interference of higher priority subsystems. This "as late as possible" also gives rise to a supremum rather than a maximum. The worst-case finalization time WF^{Q}_{sk} can therefore be described as

$$WF^{\mathrm{Q}}_{sk} = \lim_{Q \uparrow Q_s} \lim_{X \uparrow X_s} \lim_{B \uparrow B_s} WR^{\mathrm{P}}_s(B + k(Q_s + X) + Q),$$

where WR^{P}_s is the worst-case response time of a fictive subsystem S'_s with a period $P'_s = (k+1)T_s$, a normal budget $Q'_s = k(Q_s + X) + Q$, and a maximum blocking time B. Using the following equation from [14]

$$\lim_{x \uparrow C} WR^{\mathrm{P}}_i(x) = WR^{\mathrm{P}}_i(C) \tag{13}$$

we derive

$$WF^{\mathrm{Q}}_{sk} = WR^{\mathrm{P}}_s(B_s + (k+1)Q_s + kX_s). \tag{14}$$

3) Worst-case response time: Let job ι_{sk} of S_s access $R_l \in \mathcal{R}$. When ι_{sk} starts to consume its overrun budget, the subsystems S_{s-1} till S_{RC_l} are already blocked, and only subsystems with a priority higher than RC_l can therefore still pre-empt X_s. To determine the worst-case response time WR_{skl} of job ι_{sk} of S_s, we now introduce a fictive subsystem S'_{RC_l}, i.e. a subsystem that can only be preempted by tasks with a higher priority than RC_l. The preemptions during WF^{Q}_{sk} by subsystems S_{s-1} till S_{RC_l} are treated as *additional* blocking of S'_{RC_l}. The worst-case interference of the subsystems S_{s-1} till S_{RC_l} in the interval of length WF^{Q}_{sk} is denoted by $WI^{s-1}_{RC_l,k}$ and given by

$$WI^{s-1}_{RC_l,k} = \sum_{s-1 \geq t \geq RC_l} \left\lceil \frac{WF^{\mathrm{Q}}_{sk}}{P_t} \right\rceil (Q_t + X_t). \tag{15}$$

The worst-case response time WR_{skl} of job ι_{sk} of subsystem S_s when it accesses R_l is now given by

$$\begin{aligned} WR_{skl} &= \lim_{X \uparrow X_{sl}} WR^{\mathrm{P}}_{RC_l}(B'_{RC_l} + Q'_s + X)) - kP_s \\ &= WR^{\mathrm{P}}_{RC_l}(B'_{RC_l} + Q'_s + X_{sl})) - kP_s, \end{aligned} \tag{16}$$

where $WR^{\mathrm{P}}_{RC_l}$ represents the worst-case response time of a fictive subsystem S'_{RC_l} with a (budget) period P'_{RC_l} and a deadline equal to $(k+1)P_s$, a normal budget Q'_s equal to $(k+1)Q_s + kX_s$, an overrun budget X'_s equal to X_{sl}, and a maximum blocking time B'_{RC_l} given by

$$B'_{RC_l} = B_s + WI^{s-1}_{RC_l,k}. \tag{17}$$

When a subsystem uses multiple global resources, we have to be very careful. In particular, when the resource ceiling RC_l of resource $R_l \in \mathcal{R}_s$ is *larger* than $RC_{l'}$ of resource $R_{l'} \in \mathcal{R}_s$, i.e. *more* subsystems can preempt S_s during its access to R_l than to $R_{l'}$, and the maximum execution access time X_{sl} of S_s to R_l is *smaller* than $X_{sl'}$, the system may be schedulable for $R_{l'}$ but not for R_l. As an example consider a system containing 2 global resources R_1 and R_2 and 3 subsystems S_1, S_2, and S_3, where the subsystems have timing characteristics as given in Table II.

subsystem	P_s	Q_s	$X_{s,1}$	$X_{s,2}$
S_1	5	1	0.6	0
S_2	5	0.2	0	0.2
S_3	7	3	1	0.4

Table II
SUBSYSTEM CHARACTERISTICS OF Sys_{II}.

The schedulability of S_3 for $X_{3,1}$ follows immediately from the similarity of systems Sys_{I} and Sys_{II}, and the feasibility area shown in Figure 1(b). Subsystem S_3 just meets its deadline at $t = 7$ for its overrun budget $X_{3,2} = 0.4$ under worst-case conditions, i.e. a simultaneous release of all three subsystems at time $t = 0$ and resources accessed by both S_1 and S_2 requiring the usage of their overrun budgets at every activation; see Figure 3. Note that subsystem S_3 will miss its deadline at time $t = 7$ for an infinitesimal increase $\epsilon > 0$ of $X_{3,2}$. The worst-case response time for job ι_{sk} is

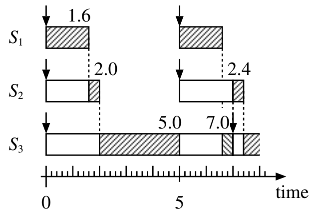

Figure 3. Subsystem S_3 just meets it deadline at $t = 7$ for $X_{3,2} = 0.4$.

therefore the maximum for all global resources accessed by S_s, i.e.

$$WR_{sk} = \max_{l} WR_{skl}. \quad (18)$$

Finally, the worst-case response time WR_s of subsystem S_s is given by [5]

$$WR_s = \max_{0 \leq k < wl_s} WR_{sk}. \quad (19)$$

C. Concluding remarks

In this section, we briefly discuss three aspects of the global analysis, i.e. the global analysis is (*i*) *uniform* and (*ii*) *sustainable* and (*iii*) will never give worse results than the original analysis. We conclude this section with a remark on the complexity of the analysis.

The analysis for FPDS [14] is not uniform for all tasks, i.e. the analysis for the lowest priority task differs from the analysis of the other tasks. This anomaly is caused by the fact that the lowest priority task cannot be blocked, i.e. its blocking time is zero, and the blocking time of all other tasks is a supremum rather than a maximum. Unlike the analysis for FPDS [14], the global analysis presented in this section is uniform. This is an immediate consequence of the fact that blocking of a global resource R_l by a subsystem S_s is already done during the execution of the normal budget, i.e. *before* the execution based on the overrun budget starts. As a result, subsystems S_{s-1} till S_{RC_l} cannot preempt S_s at the finalization time of Q_s.

As described in [16], a schedulability test is *sustainable* if any task system deemed schedulable by the test remains so if it behaves 'better' than mandated by its system specifications, i.e. sustainability requires that schedulability be preserved in situations in which it should be 'easier' to ensure schedulability. Given our scheduling model, we use the following definition for sustainability of our tighter global schedulability test.

Definition 1: A schedulability test for our real-time scheduling model for subsystems is *sustainable* if any system deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual job[s] are changed in any, some, or all of the following ways: (*i*) decreased normal budgets; (*ii*) decreased overrun budgets, (*iii*) later arrival times; and (*iv*) larger relative deadlines.

With this definition, sustainability of our global schedulability test immediately follows from (6), i.e. $WR_s \leq P_s = D_s$ and the fact that

- the maximum number wl_s of jobs of subsystem S_s in a level-s active period, and
- the worst-case finalization time WF_{sk}^{Q} in (14), the worst-case interference $WI_{RC_l,k}^{s-1}$ in (15), and the worst-case response time WR_{skl} in (16)

are strictly non-increasing for decreasing normal budgets, decreasing overrun budgets, and increasing budget periods of subsystems.

We will prove that the tighter global analysis will never give worse results than the original analysis i.e., it will give better results or in the worst case the same results as the original analysis. Looking at (12), if $wl_s > 1$, then the system will be unschedulable using the original analysis because the first job will miss its deadline according to the original analysis. While using the modified analysis, the same systems can be schedulable. If $wl_s = 1$ then $k = 0$ and X_s will not have any effect in (14) since $k = 0$. For modified analysis, only the subsystems with a higher priority than the resource ceiling of the resource being locked are able to preempt. Taking this into account can reduce the amount of interference considered due to higher priority subsystems in general and for $k = 0$ in particular. Which in turn can improve the results in terms of response time, schedulability and the CPU-resource requirement.

Finally, comparing the new global analysis (11-19) with the existing analysis (4) it is clear that the new analysis is more complex. For static systems, for which the set $\mathcal{S}$ of subsystems does not change during runtime, this additional complexity will not be very important since the analysis will be performed during the integration phase of the system, i.e. off-line. For dynamic systems, for which subsystems can be added or removed during runtime, the new analysis can be used when the original analysis fails to find a feasible solution.

VI. Tighter local analysis

Both the existing global schedulability analysis and the new global schedulability analysis assume a deadline for a subsystem S_s equal to its period P_s for the sum of the normal budget Q_s and the overrun budget X_s. The existing local schedulability analysis for the tasks of S_s is exclusively based on Q_s, however. Hence, when a system is feasible from a global scheduling perspective, the latest finalization time of Q_s is guaranteed to be at least X_s before the next activation of S_s. Hence, we can use the supply bound function $\mathtt{sbf}_{\Omega}(t)$ of the EDP resource model $\Omega_s(P_s, Q_s, \Delta_s)$ for overrun without payback rather than $\mathtt{sbf}_{\Gamma_s}(t)$ of $\Gamma_s(P_s, Q_s)$

[5]The interested reader is referred to [13], which explains the improvement in detail by means of a variety of timelines.

in (7), where $\Delta_s = P_s - X_s$. Because $X_s \geq 0$ for all subsystems (by definition), $\mathtt{sbf}_{\Gamma_s}(t) \leq \mathtt{sbf}_{\Omega_s}(t)$ for all subsystems. As a result, a subsystem may be schedulable according to the local analysis based on $\mathtt{sbf}_{\Omega_s}(t)$, but not be schedulable based on $\mathtt{sbf}_{\Gamma_s}(t)$.

Figure 4 shows an example of the supply bound functions $\mathtt{sbf}_{\Omega}(t)$ and $\mathtt{sbf}_{\Gamma}(t)$ for subsystem S_2 of system Sys_{I} with $Q_2 = 1.8$ and $X_2 = 2.4$.

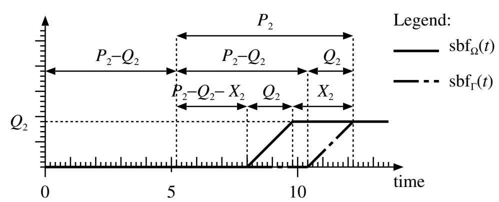

Figure 4. Supply bound functions $\mathtt{sbf}_{\Omega}(t)$ and $\mathtt{sbf}_{\Gamma}(t)$ for S_2 with $Q_2 = 1.8$ and $X_2 = 2.4$.

VII. EVALUATION

In this section, we evaluate the modified overrun without payback analysis (MONP), including both local and global new tighter analysis, with respect to CPU resource. We compare MONP with the traditional overrun without payback mechanism (ONP) using the notion of system load [2], as system load provides an indication of the system CPU requirement in the presence of shared resources. The comparison is carried out by means of simulation experiments. To show the performance of MONP relative to alternative approaches.

We start this section by briefly explaining the notion of system load and how it should be adapted for MONP.

System load: System load is defined as a quantitative measure to represent the minimum amount of CPU allocations necessary to guarantee the global schedulability of the system $\mathcal{S}$.

For ONP, system load $\mathsf{load}_{\mathsf{sys}}$ is calculated as follows:

$$\mathsf{load}_{\mathsf{sys}} = \max_{\forall S_s \in \mathcal{S}} \{\alpha_s\} \tag{20}$$

where

$$\alpha_s = \min_{0<x\leq P_s} \{ \frac{\mathtt{RBF}_s(x)}{x} \mid \mathtt{RBF}_s(x) \leq x \} \tag{21}$$

and

$$\mathtt{RBF}_s(x) = B_s + (Q_s + X_s) + \sum_{t<s} \left\lceil \frac{x}{P_t} \right\rceil (Q_t + X_t). \tag{22}$$

Note that x can be selected within a finite set of scheduling points [17], and that α_s is the smallest fraction of the CPU resource required to schedule a subsystem S_s (satisfying the schedulability condition presented in Section IV-A), assuming that the global resource supply function is $\alpha_s x$.

One can think of system load as decreasing the speed of the processor by the factor $\mathsf{load}_{\mathsf{sys}}$, which will increase the subsystems' normal budgets, the overrun budgets, and blocking times by a factor $1/\mathsf{load}_{\mathsf{sys}}$.

For MONP, evaluating system load is more complex than e.g., for ONP, because it has more than one response time equation for global schedulability analysis (see Section V), unlike the case for ONP which has only one equation. To perform the schedulability analysis for MONP, firstly, the value of wl_s should be evaluated in order to evaluate the range of k that is used by the other equations. However, we can not evaluate the value of wl_s in (12) without having the value of system load known. Without having the range of k, we can not use the equations (14) - (19) that are required in the calculation of system load. We solve this problem by using a binary search algorithm, such that the system load is selected by the search algorithm and corresponding system schedulability is checked. To do this we mutiply all subsystems normal budgets, maximum overrun budgets, and blocking times in equations (12) - (19) by a factor $1/\mathsf{load}_{\mathsf{sys}}$. If the system is schedulable, then the algorithm will select a lower system load and try again. If the system is unschedulable, then the algorithm will select a higher system load. The algorithm terminates if the selected system load $\mathsf{load}_{\mathsf{sys}} > 1$ and the system is unschedulable, or when the difference between the previous and the current system load is less than a given acceptance limit. Since we have used a binary search algorithm for MONP, the complexity of evaluating the system load is higher compared with ONP. However, note that we use the system load for comparison purposes only, hence it does not have any relationship with the complexity of the schedulability analysis.

The efficiency of MONP is measured by the amount of system load required for schedulability, relative to ONP.

Both the new tighter local and global analysis in MONP can decrease the system load. For the tighter local analysis, it has the potential to decrease the subsystem normal budget for certain subsystems, which in turn, can decrease the system load, since it decreases the effect of the interference from higher priority subsystems and the required normal budget of the subsystem itself, in equations (12) - (19). However, there is no guarantee that the improved local analysis can decrease the subsystem's normal budget. Note that $\mathtt{sbf}_{\Gamma_s}(x) < \mathtt{sbf}_{\Omega_s}(x)$ for $AP_s - 2Q_s - X_s < x < AP_s - Q_s$ where $A \in \mathbb{N} | A \geq 2$, and $\mathtt{sbf}_{\Gamma_s}(x) = \mathtt{sbf}_{\Omega_s}(x)$ otherwise. So looking at (7), the new local analysis depends on where the value of x (that makes the left hand side of the equation, which represent the resource demand, equal to the right hand side, which is the supply bound function) is located in the above mentioned ranges. If that value of x is in the range that makes $\mathtt{sbf}_{\Gamma_s}(x) < \mathtt{sbf}_{\Omega_s}(x)$ then it will decrease the subsystem normal budget, otherwise, it will not. The amount of improvement when using the new tighter local analysis compared with the original analysis, on the system load, depends on many factors such as the size of X_s, the subsystem period and the difference between the subsystem period and tasks' deadlines. The higher the value of X_s, the more improvement can be achieved. Also, if the difference between the subsystem's period and its tasks' deadlines is low, then the improvement in system load becomes higher. If the difference between the subsystem period and its tasks' deadlines is high, then the x that makes the left and right hand side of (7) equal becomes very far from the subsystem period. In this case a small increment in the subsystem's normal budget will be enough to cover the difference between $\mathtt{sbf}_{\Gamma_s}(x)$ and $\mathtt{sbf}_{\Omega_s}(x)$, which also affects the improvement in the system load.

Now we will explain the impact of the new tighter global analysis on the system load, and we will use Figure 5 for illustration. When the $X_s/\mathsf{load}_{\mathsf{sys}}$ part in Figure 5 is as large as possible, the new global analysis contributes with a larger improvement. The reason for this is that during this part there will be no (or limited) preemptions from higher priority subsystems. Hence, the difference between this part and the other part $(I+Q_s)/\mathsf{load}_{\mathsf{sys}}$ should be low to achieve a greater improvement, where I is the interference from higher priority subsystems (including the sum of Q_t+X_t of the higher priority subsystems) and also the blocking from lower priority subsystems. We can distinguish some cases in which the new global analysis can not reduce the system load. First, if the subsystem period of all subsystems are equal, then there will be no preemptions from higher priority subsystems during the overrun time. Since the tighter global analysis is based on removing the interference from higher priority subsystems during the overrun from the global analysis, the new tighter global analysis can not decrease the system load. Another case where the new global analysis can not decrease the system load, is when the subsystem that requires maximum CPU resources (i.e., the system load was computed based on its CPU requirement), is the highest priority subsystem or does not access a global shared resource.

Finally, combining both the local improvement and the global improvement can require lower system load. As mentioned previously, the new tighter local analysis has a potential to decrease the subsystem budget which will decrease the interference from higher priority subsystems and the budget of the subsystem itself $(I + Q_s)$.

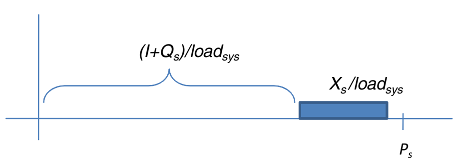

Figure 5. Considering MONP analysis for S_s.

A. Simulation setting

The simulation is performed by applying the modified overrun without payback analysis (MONP), including the new tighter local and global analysis, on 1000 different randomly generated systems. Initially, we assumed that each system consists of 5 subsystems and each subsystem contains 4 tasks. A task is assumed to access at most one globally shared resource and 2 tasks in each subsystem access globally shared resources and we assume that there is only one global shared resource.

For simplicity, we assume that the internal resource ceilings of the globally shared resources are equal to the highest task priority in each subsystem (i.e., $rc_{sl} = 1$), and $T_i = D_i$ for all tasks. For each simulation study the following settings are changed and a new 1000 systems is generated:

1) Critical section execution time CS_s. It specifies the maximum absolute time that a task may access a global shared resource. Changing this parameter does not require to generate new 1000 system, since changing only this parameter will not have effect on the other task parameters as we will show later.
2) Subsystem period P_s and task period T_{si}. The subsystem/task period is specified as a range with a lower and upper bound. The simulation program generates a subsystem/task period randomly within the specified range, following a uniform distribution.
3) Number of subsystems N.
4) System utilization $U^{\mathcal{S}}$. The sum of the utilization of all tasks in the system, is specified to a desired value.

The given system utilization is divided randomly among the subsystems. The assigned utilization to each subsystem is in turn divided randomly to the tasks that belong to that subsystem. Since the task period is generated to a value within the interval as specified, the execution time is derived from the desired task utilization. The critical section execution time is given as an input parameter, however, its value can not be greater than the execution time of its task. The critical section is therefore set to the minimum value of the task execution time and the given critical section execution time, i.e., $c_{sil} = \min(CS_s, C_{si})$. All randomized system parameters are generated following uniform distributions.

B. Simulation results

We have performed 4 different simulation studies and selected some range of values in order to highlight some properties of the new analysis as described below;

- **Study 1** is specified having critical section execution time $CS_s \in \{2,4,6,8\}$, task periods $T_{si} \in [140,1000]$, subsystem periods $P_s \in [40,70]$, $U^{\mathcal{S}} = 20\%$ and $N = 5$.
- **Study 2** increase the range of the subsystem periods P_s and task periods T_{si} (compared to Study 1) to
  A.) $P_s \in [50,200]$ and $T_{si} \in [400,1000]$,
  B.) $P_s \in [100,200]$ and $T_{si} \in [400,1000]$.
- **Study 3** change the number of subsystems (compared to Study 1) to $N \in \{4,6,8\}$.

- **Study 4** change the system utilization (compared to Study 1) to $U^{\mathcal{S}} \in \{10\%, 30\%\}$ with $CS_s = 2$.

CS_s	2	4	6	8
Q1 $\mathsf{load_{sys}}$ ONP	0.505	0.664	0.763	0.836
Median $\mathsf{load_{sys}}$ ONP	0.532	0.717	0.849	0.940
Q3 $\mathsf{load_{sys}}$ ONP	0.560	0.771	0.929	> 1
schedulable ONP	100%	100%	89.1%	67%
Q1 $\mathsf{load_{sys}}$ MONP	0.475	0.613	0.702	0.760
Median $\mathsf{load_{sys}}$ MONP	0.495	0.655	0.770	0.845
Q3 $\mathsf{load_{sys}}$ MONP	0.516	0.696	0.837	0.940
schedulable MONP	100%	100%	98.3%	84.7%
MONP/ONP med. improv.	7.4%	9.4%	10.2%	11.1%
MONP/ONP max. improv.	13.2%	19.6%	25.7%	30.0%

Table III
RESULTS OF STUDY 1

Table III shows results of **Study 1**. The results of each method (ONP and MONP) are shown using the median, lower quartile ($Q1$) and the higher quartile ($Q3$) of the system load values of the 1000 generated systems. It also shows the percentage of schedulable systems out of the 1000 generated systems. In addition, it shows the percentage of improvement in the system load based on the evaluated median (explained above) and the maximum improvement when using MONP compared with ONP. It is calculated as $100 * (\mathsf{load}_{\mathsf{sys}}^{\mathrm{ONP}} - \mathsf{load}_{\mathsf{sys}}^{\mathrm{MONP}})/\mathsf{load}_{\mathsf{sys}}^{\mathrm{MONP}}$, where $\mathsf{load}_{\mathsf{sys}}^{\mathrm{ONP}}$ is the median or maximum system load, depending on what is required to be evaluated.

For the case of $CS_s = 8$, some of the systems are unschedulable (i.e., having $\mathsf{load_{sys}} > 1$) using both ONP and MONP, because the systems that have $\mathsf{load_{sys}} > 100\%$. It is not important to find the actual system load for unschedulable systems.

Looking at the results in Table III, it is clear that MONP can give better results compared to traditional ONP in terms of a lower system load and more schedulable systems when increasing CS_s (same results are shown in Tables IV and V). In this study, the ratio CS_s/P_s is relatively high and this is the reason why MONP performs significantly better than ONP. This is the main characteristics that we are looking for.

In **Study 2**, we decrease the ratio CS_s/P_s by increasing the range of subsystem period. In Table IV, the subsystem period is selected as $P_s = [50, 200]$. In this case, the improvement that MONP can achieve is less than in **Study 1** because $X_s/\mathsf{load_{sys}}$ becomes less significant within the subsystem period P_s. In Table V, we change the range of subsystem period to $P_s = [100, 200]$.This causes MONP to give better results compared to the results in Table IV. The reason for this improvement is that when the difference between the minimum and maximum subsystem period is decreased, then also the maximum number of interferences (preemptions) from higher priority subsystems is decreased. This will decrease the contribution of the higher priority subsystems in equations (12) - (19) which in turn decreases the required subsystem load when using MONP.

Looking at the MONP/ONP med. improv. line in Table IV and Table V, the rate of the improvement when increasing CS_s from 2 to 4 is lower when increasing CS_s from 4 to 6 and when increasing CS_s from 6 to 8 (for example in Table IV, the difference between $4.9\% - 3.1\%$ is higher than $5.6\% - 4.9\%$). The reason for this is that increasing CS_s of tasks will increase the required subsystems normal budgets for their subsystems (see (7)) which, in turn, will increase the interference from higher priority subsystems in (12) - (19) and that limits the improvement that MONP can achieve.

CS_s	2	4	6	8
Q1 $\mathsf{load_{sys}}$ ONP	0.444	0.538	0.607	0.659
Median $\mathsf{load_{sys}}$ ONP	0.462	0.562	0.644	0.704
Q3 $\mathsf{load_{sys}}$ ONP	0.481	0.589	0.683	0.755
schedulable ONP	100%	100%	100%	99.8%
Q1 $\mathsf{load_{sys}}$ MONP	0.430	0.512	0.573	0.620
Median $\mathsf{load_{sys}}$ MONP	0.448	0.536	0.609	0.661
Q3 $\mathsf{load_{sys}}$ MONP	0.467	0.563	0.643	0.708
schedulable MONP	100%	100%	100%	100%
MONP/ONP med. improv.	3.1%	4.9%	5.6%	6.2%
MONP/ONP max. improv.	8.1%	12.3%	16.2%	16.8%

Table IV
RESULTS OF STUDY 2A, $P_s \in [50, 200]$

CS_s	2	4	6	8
Q1 $\mathsf{load_{sys}}$ ONP	0.446	0.522	0.579	0.623
Median $\mathsf{load_{sys}}$ ONP	0.468	0.548	0.611	0.662
Q3 $\mathsf{load_{sys}}$ ONP	0.488	0.572	0.643	0.699
schedulable ONP	100%	100%	100%	100%
Q1 $\mathsf{load_{sys}}$ MONP	0.432	0.497	0.545	0.583
Median $\mathsf{load_{sys}}$ MONP	0.454	0.518	0.573	0.616
Q3 $\mathsf{load_{sys}}$ MONP	0.473	0.543	0.604	0.651
schedulable MONP	100%	100%	100%	100%
MONP/ONP med. improv.	$3,1\%$	5.8%	$6,6\%$	7.5%
MONP/ONP max. improv.	6.4%	11.9%	16.5%	17.2%

Table V
RESULTS OF STUDY 2B, $P_s \in [100, 200]$

In **Study 3**, we investigate the effect of changing the number of subsystems. The results are shown in Table VI. We can see that increasing N will decrease the improvement that MONP can achieve over ONP. The reason for this is that increasing the number of subsystems will increase the interference I of the higher priority subsystems which, in turn, will decrease the improvement as explained in the previous section.

Finally, in **Study 4** we investigate the effect of changing the system utilization on the performance of MONP. The results in Table VII show that increasing the value of $U^{\mathcal{S}}$ will decrease the improvement that MONP can achieve over ONP. The reason for this is that increasing the value

N	4	5	6	8
Q1 $\mathsf{load}_{\mathsf{sys}}$ ONP	0.459	0.505	0.560	0.674
Median $\mathsf{load}_{\mathsf{sys}}$ ONP	0.483	0.531	0.590	0.708
Q3 $\mathsf{load}_{\mathsf{sys}}$ ONP	0.506	0.560	0.617	0.743
schedulable ONP	100%	100%	100%	100%
Q1 $\mathsf{load}_{\mathsf{sys}}$ MONP	0.430	0.475	0.526	0.637
Median $\mathsf{load}_{\mathsf{sys}}$ MONP	0.448	0.495	0.549	0.669
Q3 $\mathsf{load}_{\mathsf{sys}}$ MONP	0.467	0.516	0.575	0.702
schedulable MONP	100%	100%	100%	100%
MONP/ONP med. improv.	7.8%	7.4%	7.3%	5.9%

Table VI
RESULTS OF STUDY 3 FOR $CS_s = 2$

of U^S will increase the subsystem normal budget for all subsystems which increases the contribution of the higher priority subsystems in (12) - (19) and will limit the potential improvement of MONP as explained previously.

U^S	10%	20%	30%
Q1 $\mathsf{load}_{\mathsf{sys}}$ ONP	0.348	0.505	0.661
Median $\mathsf{load}_{\mathsf{sys}}$ ONP	0.376	0.531	0.690
Q3 $\mathsf{load}_{\mathsf{sys}}$ ONP	0.402	0.560	0.718
schedulable ONP	100%	100%	100%
Q1 $\mathsf{load}_{\mathsf{sys}}$ MONP	0.323	0.475	0.625
Median $\mathsf{load}_{\mathsf{sys}}$ MONP	0.344	0.495	0.649
Q3 $\mathsf{load}_{\mathsf{sys}}$ MONP	0.368	0.516	0.674
schedulable MONP	100%	100%	100%
MONP/ONP med. improv.	9.3%	7.4%	6.2%

Table VII
RESULTS OF STUDY 4 FOR $CS_s = 2$

VIII. CONCLUSION

In this paper we have shown that existing global and local schedulability analysis of synchronization protocols based on SRP and overrun without payback for two-level hierarchical scheduling based on FPPS is pessimistic. We presented a new tighter global and local analysis assuming that the deadline of a subsystem holds for the sum of its normal budget and its overrun budget, and shown that the global analysis is both uniform and sustainable. We have illustrated the improvements by means of examples, and have evaluated the improvement through an extensive simulation study. The evaluation results show that our novel analysis can improve the CPU requirement significantly for certain cases especially when the ratio between X_s/P_s is high, which makes the performance of the existing analysis low.

REFERENCES

[1] M. Behnam, I. Shin, T. Nolte, and M. Nolin, "SIRAP: A synchronization protocol for hierarchical resource sharing in real-time open systems," in *Proc. 7^{th} ACM and IEEE Int. Conference on Embedded Software (EMSOFT)*, Oct. 2007, pp. 279–288.

[2] M. Behnam, T. Nolte, and I. Shin, "Scheduling of semi-independent real-time components: Overrun methods and resource holding times," in *Proc. 13^{th} IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2008, pp. 575–582.

[3] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proc. 24^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2003, pp. 2–13.

[4] T. Baker, "Stack-based scheduling of realtime processes," *Real-Time Systems*, vol. 3, no. 1, pp. 67–99, Mar. 1991.

[5] Z. Deng and J.-S. Liu, "Scheduling real-time applications in open environment," in *Proc. 18^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 1997, pp. 308–319.

[6] T.-W. Kuo and C.-H. Li, "A fixed-priority-driven open environment for real-time applications," in *Proc. 20^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 1999, pp. 256–267.

[7] G. Lipari and S. Baruah, "Efficient scheduling of real-time multi-task applications in dynamic systems," in *Proc. 6^{th} IEEE Real-Time Technology and Applications Symposium (RTAS)*, May 2000, pp. 166–175.

[8] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using EDP resource models," in *Proc. 28^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2007, pp. 129–138.

[9] T. Ghazalie and T. Baker, "Aperiodic servers in a deadline scheduling environment," *Real-Time Systems*, vol. 9, no. 1, pp. 31–67, Jul. 1995.

[10] R. Davis and A. Burns, "Resource sharing in hierarchical fixed priority pre-emptive systems," in *Proc. 27^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2006, pp. 257–267.

[11] I. Shin, M. Behnam, T. Nolte, and M. Nolin, "Synthesis of optimal interfaces for hierarchical scheduling with resources," in *Proc. 29^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2008, pp. 209–220.

[12] M. Bertogna, N. Fisher, and S. Baruah, "Static-priority scheduling and resource hold times," in *Proc. 15^{th} Int. Workshop on Parallel and Distributed Real-Time Systems*, Mar. 2007, pp. 1–8.

[13] R. Bril, U. Keskin, M. Behnam, and T. Nolte, "Schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks revisited," Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, Tech. Rep. CSR 10-05, Jun. 2010.

[14] R. Bril, J. Lukkien, and W. Verhaegh, "Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption," *Real-Time Systems journal*, vol. 42, no. 1-3, pp. 63–119, Aug. 2009.

[15] J. Regehr, "Scheduling tasks with mixed preemption relations for robustness to timing faults," in *Proc. 23^{rd} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2002, pp. 315–326.

[16] S. Baruah and A. Burns, "Sustainable schedulability analysis," in *Proc. 27^{th} IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2006, pp. 159–168.

[17] G. Lipari and E. Bini, "A methodology for designing hierarchical scheduling systems," *J. Embedded Comput.*, vol. 1, no. 2, pp. 257–269, 2005.