공학석사 학위논문

Scheduling Adaptive Variable-Rate Tasks using Compositional Real-Time Scheduler for Automotive System

자동차 시스템을 위한 합성적 실시간 스케줄러를 이용한 적응형 가변주기 태스크의 스케줄링

2015년 2월

서울대학교 대학원

전기 · 컴퓨터공학부

이 지 화

Abstract

Scheduling Adaptive Variable-Rate Tasks using Compositional Real-Time Scheduler for Automotive System

Jihwa Lee

Department of Computer Science and Engineering

The Graduate School

Seoul National University

Periodic resource model allows integration of multiple real-time applications on a shared hardware with temporal isolation. Meanwhile, recent studies reported that certain automotive application (i.e., engine control) requires its so-called *adaptive variable-rate tasks* to be executed following the status of its control target (i.e., crankshaft rotation angle). Since such task is not studied in conjunction with periodic resource model, a new method is needed for scheduling them on a periodic resource. For this, this paper proposes a novel method for finding tight periodic resource parameters for given adaptive variable-rate tasks. Our method first reduces each of them into a periodic task where if it is schedulable on a periodic resource, the original task

is also guaranteed to be schedulable. Then optimal periodic resource parameters are found considering context switching overheads. To the best of our knowledge, this is the first effort to schedule adaptive variable-rate tasks using periodic resource model.

Keywords : Embedded Systems, Real-Time Systems, Automobile, Engine Application, Adaptive Variable-Rate Task

Student Number : 2012-23233

Contents

List of Figures

List of Tables

1 Introduction

Modern vehicles are equipped with various advanced vehicle safety and comfortability functions. To implement such system, the automotive industry is following the *federated architecture* where each vehicle function is implemented as a separate Electronic Control Unit (ECU) and they are eventually integrated through in-vehicle networks like the Controller Area Network (CAN) bus. As a result, the number of required ECUs in a vehicle is reaching more than 100 ECUs, which imposes serious problems on vehicle weight, fuel efficiency, passenger space, and wire harness complexity.

To solve such problems, automakers are now trying to integrated group of ECUs in a common control domain (e.g., power-train or chassis control) into a single more powerful ECU – *integrated architecture* [1]. Since vehicle control systems have strict timing requirements, however, just gathering the control applications in a single ECU does not make any sense, which will break every control function due to the unexpected time delays between them.

When integrating multiple real-time applications respecting each application's original timing requirement, *periodic resource model* (PRM) has been regarded as a decent solution [17] [18]. Using the PRM, an application's total resource requirement can be abstracted as a periodic resource characterized by Π and Θ, across the application's every constituting task. By carefully selecting these two parameters, every task

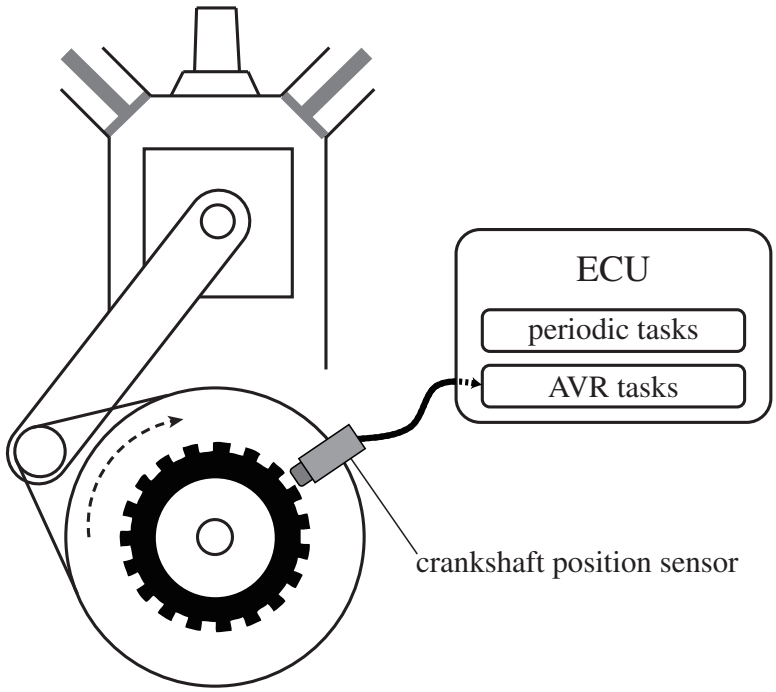

Figure 1: Crankshaft position sensor triggers AVR tasks

in the application is guaranteed to meet their deadline by just giving Θ time units to that application for every periodic resource period Π. Moreover, the application can maintain its legacy scheduling algorithm like Rate Monotonic (RM) and Earliest Deadline First (EDF) on the given periodic resource.

However, the PRM has a limitation that each task needs to be periodic. Unfortunately, real automotive system has more challenging task model, which cannot be directly handled with current method. For that, recent studies reported that the engine control application has tasks with continually varying period according to the engine's physical state, i.e., crankshaft rotation angle as in Fig. 1 and also vary-

ing execution times. For naming it, this paper uses *adaptive variable-rate tasks*, as originally proposed by [2]. Due to its unique timing characteristic, periodic resource model cannot be used for integrating such applications.

Motivated by this finding, this paper aims at scheduling adaptive variable-rate tasks using the PRM. As a specific problem, for given application with both periodic and adaptive variable-rate tasks, we try to find the optimal periodic resource parameters with the lowest utilization (i.e., $\frac{\Theta}{\Pi}$). As a solution, we first transform each adaptive variable-rate task to a corresponding periodic task, which then can be used instead of the original task when finding the proper Π and Θ for the target application. In order to guarantee that such found periodic resource parameters meet the original task's timing requirement, we proposes an algorithm that checks whether a given periodic task can safely replace the original adaptive variable-rate task or not, and we also proves its correctness with a mathematical induction. After reducing each adaptive variable-rate task to periodic tasks, we formulate our problem as a combinatorial search problem which can be solved by selectively searching the solution space.

The contributions of this paper can be summarized as follows:

- We identify the challenge when integrating automotive real-time applications for integrated system architecture and formulate the problem as how to schedule adaptive variable-rate tasks using the PRM.

- We propose an initial solution which first reduces each variable-rate task into

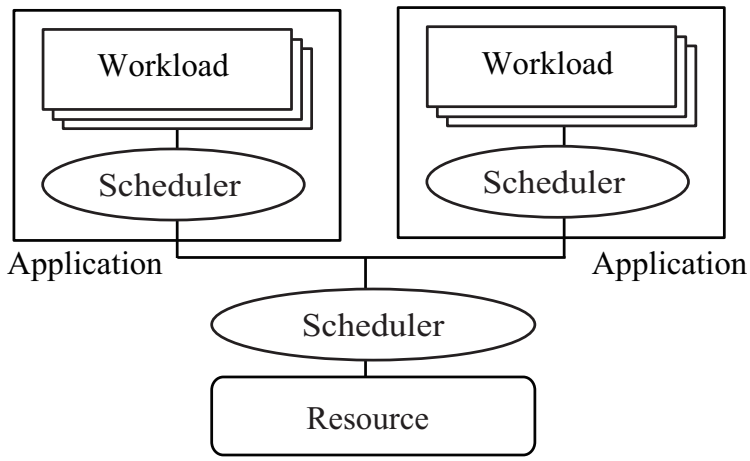

Figure 2: Hierarchical scheduling framework

a replaceable periodic task, with which we find the optimal periodic resource parameters.

The rest of this paper is organized as follows. Section 3 briefly introduces the background and defines our specific problem. Section 2 surveys the related work. Then Section 4 proposes our method for scheduling adaptive variable-rate tasks using the PRM. Section 5 evaluates our proposed method. Finally, Section 6 concludes the paper and brings up future works.

2 Related Work

A hierarchical scheduling framework is used to support hierarchical resource sharing among applications under variant scheduling policies. Fig. 2 illustrates an example of hierarchical scheduling framework which represented as a tree of nodes. Goyal

et al. [9] first introduced a hierarchical scheduling framework to support different scheduling policies for different applications for a multimedia system.

The hierarchical scheduling framework guarantees independent execution of multiple applications, referred partitioning. Such a partitioning enables functional separation of the application for compositional verification and validation and prevents one partitioned function from causing a failure of another partitioned function. Therefore, the hierarchical scheduling framework is especially useful for open systems, where applications are independently developed and validated. For example, the hierarchical scheduling framework allows an application to be developed independently under its own scheduler and then later integrated in a system that has a different meta-level scheduler.

For real-time systems, a large body of researches have been conducted on hierarchical scheduling frameworks. Deng and Liu [6] introduced a two-level real-time scheduling framework for open systems. Kuo and Li [11] presented an exact schedulability condition for a two-level framework under the RM scheduler, with harmonic period assumption. Lipari et al. [12] analyzed exact schedulability conditions for the two-level framework under the EDF scheduler, while setting a server between system scheduler and each component scheduler. Mok et al. [14] studied the bounded-delay resource partition model for the hierarchical scheduling framework. Feng and Mok [8] studied the component abstraction and composition problems with the bounded-

delay resource partition model.

There have been studies on the problem with the PRM. The PRM can specify the periodic resource allocation guarantees provided to a component from its parent component. Saewong et al. [16] introduced an exact RM schedulability conditions based on the worst-case response time analysis and time demand analysis. Shin and Lee [17] proposed the PRM and presents exact schedulability analysis for the model. Easwaran et al. [2] described a compositional analysis technique of components using the PRM. They also considered the selection of a PRM model that can schedule the system using minimum bandwidth. However, they only considered periodic task and sporadic task in their schedulability analysis. Asberg et al. [1] presented application integration with the use of the hierarchical scheduling framework in domain of automotive system, AUTOSAR. They addressed component isolation at runtime. However, they also did not deal with AVR task as the task model.

The periodic task model has been extensively studied in the real-time system community. Extensions to the periodic task model such as sporadic task model which have minimum inter-arrival time and arbitrary deadline have been researched. Although these task models address various relationships between their periods and deadlines, the task parameters themselves are static. The problem related to AVR tasks was first presented by Buttle [4], as a common practice adopted in automotive engine control application. Kim et al. [10] presented a task model for engine control

tasks with variable inter-arrival times and execution times depending on the angular velocity of the engine. They studied the system with a single AVR task which is running at the highest priority. Pollex et al. [15] addressed a sufficient schedulability analysis for AVR task under fixed-priority scheduling. Moreover, the analysis was formulated with continuous domain with very high complexity. Davis et al. [5] presented a sufficient schedulability analysis for the AVR task under fixed-priority scheduling based on an Integer Linear Programming (ILP) formulation. Their approach was based on a quantization of the angular velocity which may cause pessimism in the analysis. Buttazzo et al. [3] addressed schedulability analysis of AVR tasks under EDF scheduling. In addition, they presented a design method for computing the set of switching rotation speeds that keep the overall utilization bound. Biondi et al. [2] presented an exact schedulability analysis of AVR task under fixed-priority scheduling. Moreover, they discussed an efficient method for reducing the complexity of the exact analysis. All of this work has focused on deriving schedulability conditions of AVR task under EDF or fixed-priority scheduling policy and no research has been conducted in domain of hierarchical scheduling framework.

In this paper, we address issues of scheduling AVR tasks on the PRM and introduce a method for reducing an AVR task to a periodic task. Then, we can consider the reduced AVR tasks as periodic tasks. Furthermore, we optimize the PRM parameters with workload which consists of set of periodic tasks with reduced AVR tasks.

3 Background and Problem Description

3.1 Task Model

We consider L real-time applications originally separated in their own ECUs and later integrated into a single ECU. For those applications, corresponding periodic resources $\{\Gamma_1, \Gamma_2, ..., \Gamma_L\}$ are generated where each Γ_i is characterized by its resource period Π_i and resource budget Θ_i. Following these periodic resource parameters, the low-level scheduler first schedules periodic resources by giving Θ_i time units during every Π_i period to each periodic resource (or application). Basically, this low-level scheduler provides the temporal isolation among applications.

Each real-time application is then composed of N preemptive tasks $\{\tau_1, \tau_2, ..., \tau_N\}$. Each τ_i can be either a periodic task P or an AVR task R. For a periodic task, we assume it is a implicit-deadline task characterized by (p_i, e_i) where p_i is its period and e_i is the worst-case execution time. For an AVR task, we employ the task model proposed in [2]. Such τ_i generates an infinite sequence of jobs, $J_{i,1}$, $J_{i,2}$, $\cdots$, each of which is sequentially triggered by the physical status of the control target, i.e., engine crankshaft angle. The release time of the k-th job is denoted by $t_{i,k}$ and the inter-release time between two consecutive jobs is denoted by $T_{i,k} = t_{i,k+1} - t_{i,k}$. As time t flows, the inter-release time $T_{i,k}$ continually varies as the engine's angular velocity $\omega(t)$ also continually changes. Note that since engine typically has its minimum and maximum running RPM, $\omega(t)$ can only change within a certain range

denoted by $[\omega^{min}, \omega^{max}]$. Hence the inter-release time $T_{i,k}$ also falls within a fixed range denoted by $[T^{min}, T^{max}]$.

In addition to the varying inter-release times, AVR task τ_i also has varying execution times since AVR tasks are implemented as a set of M_i different execution modes $\{(C_i^m, \omega_i^m), m = 1, 2, \cdots, M_i\}$, which is determined at each job release time $t_{i,k}$ according to the engine's angular velocity at that moment $\omega(t_{i,k})$. When $\omega(t_{i,k})$ is within $(\omega_i^m, \omega_i^{m+1}]$ where $\omega^{M_i+1} = \omega^{min}$ and $\omega^1 = \omega^{max}$, the m-th mode is activated which is characterized by the constant worst-case execution time C_i^m. Also note that C_i^m monotonically decreases as $\omega(t_{i,k})$ increases meaning that the AVR task does less work when the engine speed is faster. The insight behind this can be explained that as the control frequency increases we can skip certain functions in the job without sacrificing the overall control performance. As the result, the execution time of an AVR task can be expressed as a non-increasing step function of the angular velocity as illustrated by Fig. 3. When developing engine control applications, AVR tasks are commonly used to precisely control fuel injection and ignition timimings. For more details, readers are referred to [2].

Given a job $J_{i,k}$ released with angular velocity $\omega_{i,k}$, the minimum inter-arrival time $\tilde{T}^m(\omega_{i,k})$ to have the next job J_{k+1} released in mode m is modeled as follows [2]:

$$\tilde{T}^m(\omega_{i,k}) = \frac{2\Delta\theta}{\omega_{i,k} + \omega^m}. \tag{1}$$

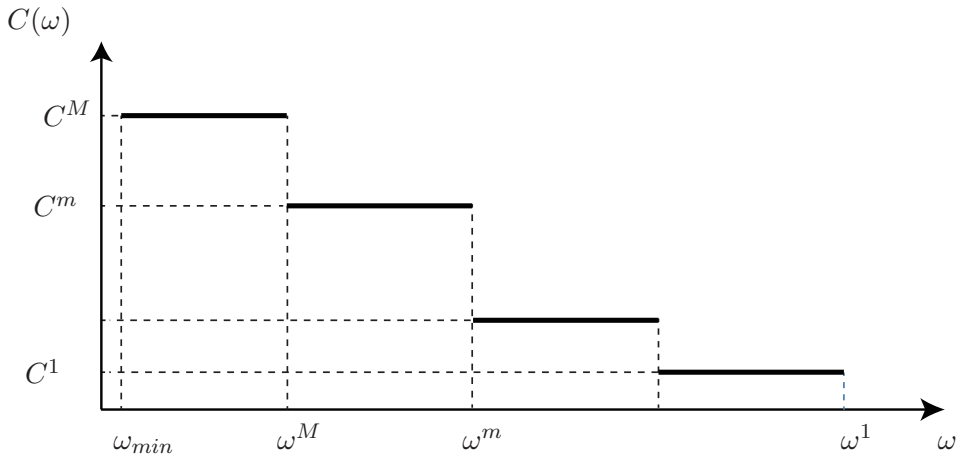

Figure 3: Execution time of an AVR task as a function of the angular velocity ω

3.2 Periodic Resource Model

The PRM is modeled as [17]. $\Gamma(\Pi, \Theta)$ describes a resource allocation of Θ time units every Π time units, where a resource period Π is a positive integer and a resource allocation time Θ is a real number in $(0, \Pi]$.

Fig. 4 illustrates hierarchical scheduling on the PRM. Component 1 and component 2 are child components that consists of periodic tasks and AVR tasks under EDF scheduling. Component 0 is a parent component that schedules the two child components which are abstracted by PRM interfaces $\Gamma_1(\Pi_1, \Theta_1)$ and $\Gamma_2(\Pi_2, \Theta_2)$, respectively.

For a PRM Γ, its supply bound function $sbf_\Gamma(t)$ is defined to compute the mini-

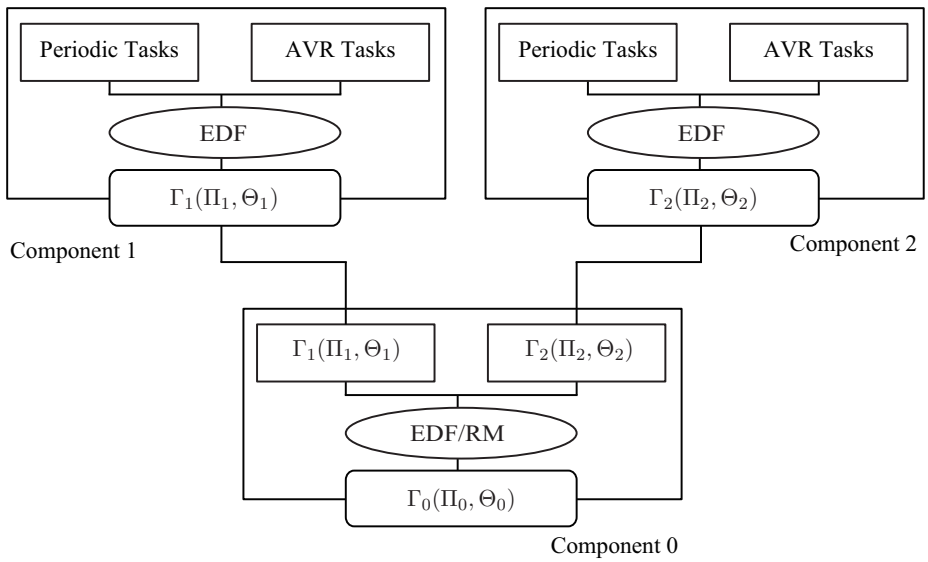

Figure 4: Hierarchical scheduling using PRM

mum resource supply for every interval length t as follows [17]:

$$sbf_{\Gamma}(t) = \left\lfloor \frac{t - (\Pi - \Theta)}{\Pi} \right\rfloor \cdot \Theta + \epsilon_s,$$

where

$$\epsilon_s = max(t - 2(\Pi - \Theta) - \Pi \left\lfloor \frac{t - (\Pi - \Theta)}{\Pi} \right\rfloor, 0).$$

For a periodic task set $W = \{P_1, P_2, ..., P_n\}$, a resource demand bound function $dbf_W(t)$ that evaluates the maximum resource demand of W under EDF scheduling

during t time units as follows [17]:

$$dbf_W(t) = \sum_{P_i \in W} \left\lfloor \frac{t}{p_i} \right\rfloor \cdot e_i.$$

Given a task set W, W is schedulable on the PRM Γ under EDF scheduling, if and only if

$$\forall 0 < t \leq LCM_W \quad dbf_W(t) \leq sbf_\Gamma(t),$$

where LCM_W is *the least common multiple* of p_i for all $P_i \in W$ [17].

3.3 Problem Description

Given a task set which consists of periodic tasks and AVR tasks, we address the following problems:

- find periodic tasks that can reduce given the AVR tasks and generate a new task set that the given AVR tasks are replaced by reduced tasks, and
- find optimal scheduling parameters of PRM $\Gamma(\Pi, \Theta)$ for the newly generated workload.

4 Our Proposed Method

Our method consists of three steps. First, we use a heuristic algorithm to check if the given a AVR task is reducable or not with a pair of (p, e). Second, we perform exhaustive search to find (p, e) pairs that can schedule the AVR task and draw the solution space with (p, e) pairs. On the solution space, we select the (p, e) pair which have the minimum utilization e/p. Second, we find PRM $\Gamma(\Pi, \Theta)$ parameters that can schedule given workload which consists of periodic tasks and reduced AVR tasks by exhaustive search. Also, we find an optimal parameter pair (Π, Θ) which have the minimum utilization Θ/Π on the solution space.

As we use a reducing approach which reduces R to P, we denote the reduced AVR task as R_P. R_P consists of two parameters (p, e) since it is identical to periodic task. In addition, the workload W^* is consist of every periodic task $P \in W$ and reduced tasks R_P which derived from $R \in W$. Eventually, our goal is scheduling W^* on PRM.

4.1 Reducing Approach

We conduct time-demand analysis to ensure the schedulability of reducing approach. The resource demand of $P(p, e)$ during time t is defined as follows:

$$demand_P(t) = \left\lfloor \frac{t}{p} \right\rfloor \cdot e$$

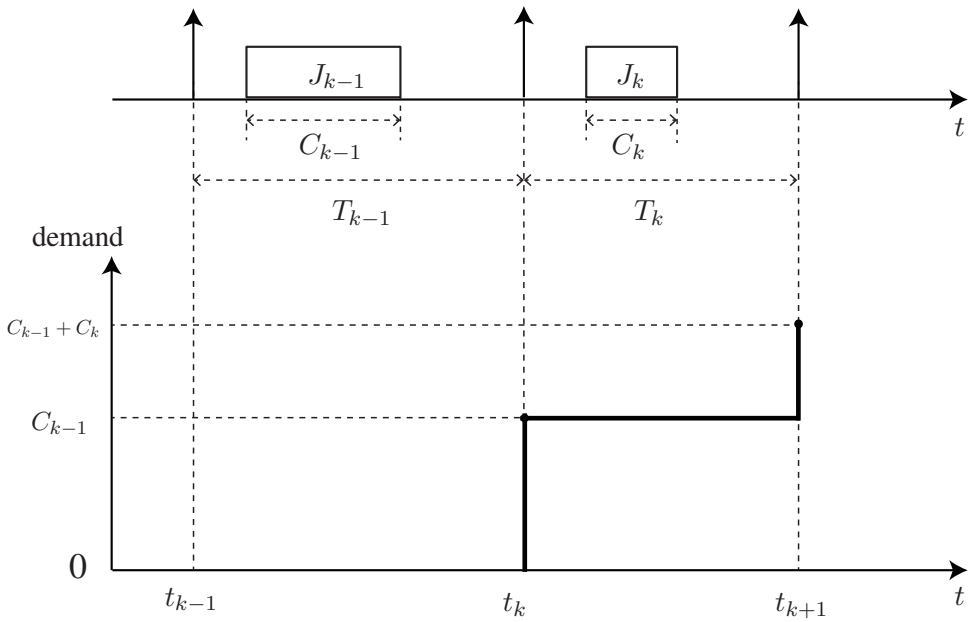

Figure 5: Drawing the demand graph of an AVR task

And we denote $demand_R(t)$ as a demand function of R during time t. It is accumulative step function that every demand increasing is appearing on the deadline of jobs. Since we assume the relative deadline is equal to its inter-arrival time, the resource demand of J_k appears at the release time of next job J_{k+1} as much as C_k. Fig. 5 shows $demand_R$ of two consecutive jobs, during $[t_{k-1}, t_{k+1}]$.

Fig. 6 displays single job demand a job released at time t_k. The figure illustrates possible inter-arrival time on the x-axis and the corresponding demand requirements on the y-axis. In the figure, T^m_{min} denotes the minimum inter-arrival time between t_k and t_{k+1} that requires demand increasing of C^m at time t_{k+1}. Similarly, T^m_{max} denotes the maximum inter-arrival time between t_k and t_{k+1} that requires demand

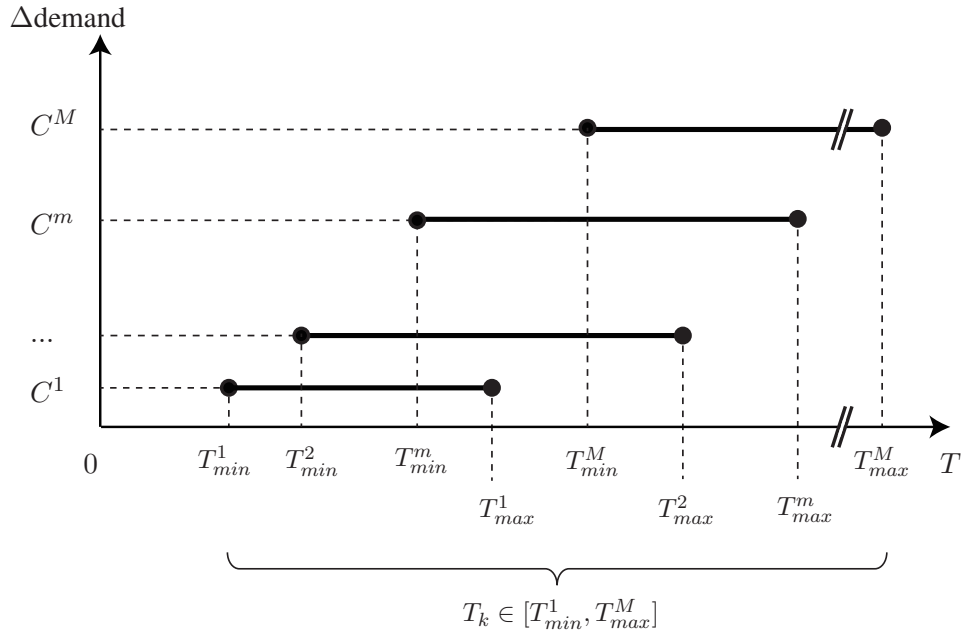

Figure 6: Single job demand with range of possible inter-arrival time T_k and the corresponding demand requirement

increasing of C^m at time t_{k+1}. At time t_k, if ω_k belongs to the mode m, the minimum inter-arrival time to t_{k+1} is given as $\tilde{T}^1(\omega^m)$, i.e., $T^m_{min} = \tilde{T}^1(\omega^m)$. We use the maximum angular velocity of mode m, i.e., ω^m, instead of ω_k in the worst-case assumption to achieve the minimum inter-arrival time with enduring little pessimism. If the AVR task is running on mode m at time t_k, it needs C^m at time t_{k+1}. Since the inter-arrival time T_k lies in the range of $[T^1_{min}, T^M_{max}]$, $t_{k+1} \in [t_k + T^1_{min}, t_k + T^M_{max}]$.

Fig. 7 draws $demand_{R_P}$ and the single job demand simultaneously and compares two demand functions by overlapping. The gray shaded area denotes possible the single job demand and the bold dashed-lines denote the worst-case demand with

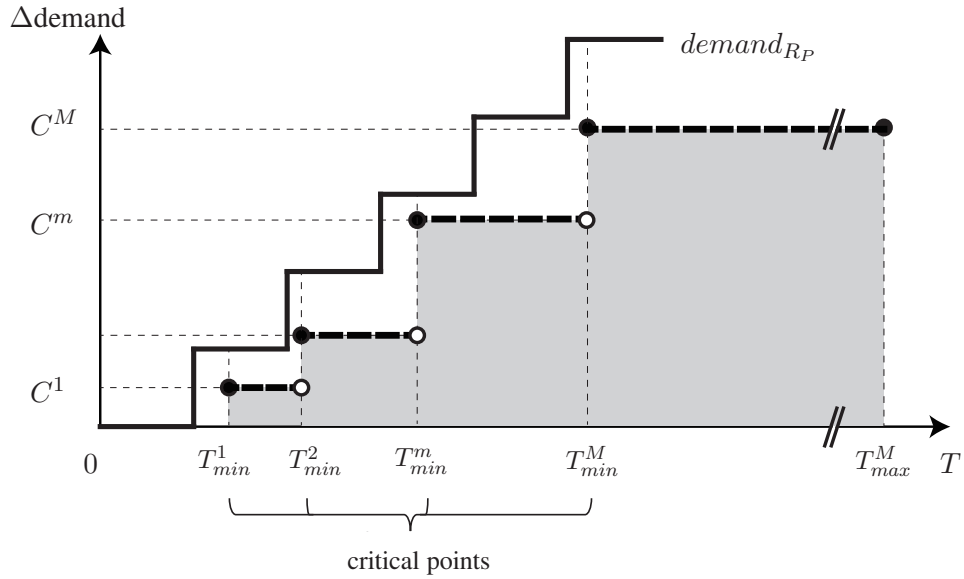

Figure 7: Comparison of $demand_{R_P}$ and $demand_R$ with the critical points

given inter-arrival time T. Note that the demand increasing only appears on M points, i.e., at time instant T^m_{min}, where $m = 1, 2, ..., M$. We name it as critical points. The following theorem allows reducing an AVR task to a periodic task in the mean of comparing the resource demand under EDF scheduling.

Theorem 1. *An AVR task R can be reduced to a periodic task R_P under EDF scheduling, if and only if*

$$demand_R(t) \leq demand_{R_P}(t), \tag{2}$$

$for\ \forall t \geq 0.$

Proof. To show the necessity, we prove the contrapositive, i.e., if Eq.(2) is false, R is

cannot be reduced to R_P, that means some jobs of R are not schedulable under EDF scheduling. If the demand of R under EDF scheduling during t exceeds the demand of R_P during t, there is clearly no feasible schedule.

To show the sufficiency, we prove the contrapositive, i.e., if R is cannot be reduced to R_P, Eq.(2) is false. Let t_2 be the first time instant at which a job of R, say $J_{R,i}$, misses its deadline. And let t_1 be the latest time instant at which the demand of R_P was idle or was executing a job of R whose deadline is before t_2 and which was released at t_1. Since $J_{R,i}$ misses its deadline at t_2, the demand placed on R in the time interval $[t_1, t_2)$ is greater than the demand of R_P in the same time interval length $t_2 - t_1$. □

To reduce an AVR task to a periodic task, we may perform exhaustive search to find the worst-case demand requirement all possible job release scenarios during the all time $t \leq 0$ and compare the demand of periodic task, based on Theorem 1. However, that approach is very expensive in terms of computational complexity and not practical. Therefore, we introduce an effective heuristic algorithm to find out the worst-case of demand during the inter-arrival time of every job release.

Theorem 2. *Eq.(2) for $\forall t \geq 0$, if we select p and e which satisfy*

$$C^m \leq \left\lfloor \frac{T^m_{min}}{p} \right\rfloor \cdot e, \tag{3}$$

where m=1, 2, ..., M.

Proof. We start by considering a general property of comparison between non-decreasing step functions. Since $demand_R$ and $demand_{R_P}$ are non-decreasing step functions, we only need to check the every job deadline, the release of next job under our assumption, is $t = t_k$, where $k \in \{1, 2, ...\}$.

Now, the rest of the proof is by induction on the time instants at which every job releases.

Step 1: For the first job release, i.e., for $t = t_1$, the first job of R, $J_{R,1}$ is released and the first job of P, $J_{P,1}$ is released synchronously at time $t = 0$. We select (p, e) to satisfy Eq.(3).

Comparison between two non-decreasing step functions, we only need to check some points that increase the demand, i.e., critical points, since they dominate all other points behind it until the next demand increasing point appears. Therefore, Eq.(3) is true for $0 \leq t \leq T^M_{max}$.

Step 2: Suppose Eq.(2) is true for some $t = t_k$, that is

$$demand_R(t_k) \leq demand_{R_P}(t_k). \tag{4}$$

Step 3: Prove Eq.(2) is true for $t = t_{k+1}$, that is

$$demand_R(t_{k+1}) \leq demand_{R_P}(t_{k+1}). \tag{5}$$

Let $T = t_{k+1} - t_k$, then left side of Eq.(5) becomes

$$demand_R(t_k + T) = demand_R(t_k) + C^m.$$

Now we add C^m to both sides of Eq.(4)

$$demand_R(t_k) + C^m \leq demand_{R_P}(t_k) + C^m.$$

And we are showing that

$$demand_{R_P}(t_k) + C^m \leq demand_{R_P}(t_{k+1}) = \left\lfloor \frac{t_{k+1}}{p} \right\rfloor \cdot e \tag{6}$$

$$C^m \leq \left\lfloor \frac{t_{k+1}}{p} \right\rfloor \cdot e - demand_{R_P}(t_k) \tag{7}$$

$$C^m \leq (\left\lfloor \frac{t_{k+1}}{p} \right\rfloor - \left\lfloor \frac{t_k}{p} \right\rfloor) \cdot e \tag{8}$$

$$C^m \leq (\left\lfloor \frac{t_k + T}{p} \right\rfloor - \left\lfloor \frac{t_k}{p} \right\rfloor) \cdot e \tag{9}$$

$$C^m \leq \left\lfloor \frac{T}{p} \right\rfloor \cdot e \tag{10}$$

Eq.(10) holds by the Step 1, since $T \in [T^1_{min}, T^M_{max}]$.

By Step 1, 2, and 3, $demand_R(t) \leq demand_{R_P}(t)$ is true for every job release $t = t_k$, that means Eq.(2) holds for $\forall t \geq 0$. □

Based on the Theorem 2, we introduce the check reducable algorithm. The algorithm tells if the given $R(C_k)$ is reducable or not to given $R_P(p, e)$.

Algorithm 1 Check Reducable

Input: $R_P(p, e)$, $R(C_k)$
Output: whether $R_P(p, e)$ can schedule $R(C_k)$ or not
begin procedure CheckReducable(R_P, C_k)
1. **for each** mode m **in** C_k **do**
2. $\quad T^m_{min} \leftarrow \tilde{T}^1(\Omega^m)$;
3. $\quad C^{next} \leftarrow C(\Omega^m)$;
4. $\quad$ **if** $demand_{R_P}(T^m_{min}) < C^{next}$ **then**
5. $\quad\quad$ **return false;**
6. **end for**
7. **return true;**

end procedure

Algorithm 1 starts from the assumption that the first job of R and P are release at $t = 0$ simultaneously. The algorithm searches for the worst-case demand by R, by changing the current execution mode m in line 1. We select the minimum inter-arrival time to the next job release in line 2. To calculate the minimum inter-arrival time, the algorithm uses the maximum ω for each mode m and assumes that the next job having ω^1. Since we know the execution time of the next release time with given mode m, simply use $C(\Omega^m)$ for the demand comparison, in line 4. The algorithm returns true only if $R(C_k)$ can be reduced with given $P(p, e)$.

4.2 Search and Selection

We perform exhaustive search to select $R(p, e)$. Out of many (p, e) pairs, we select the one having the minimum $U = e/p$. Table 1 describes execution time of each mode in the example AVR task.

Fig. 8 shows the solution space of (p, e) pairs that can schedule the example AVR

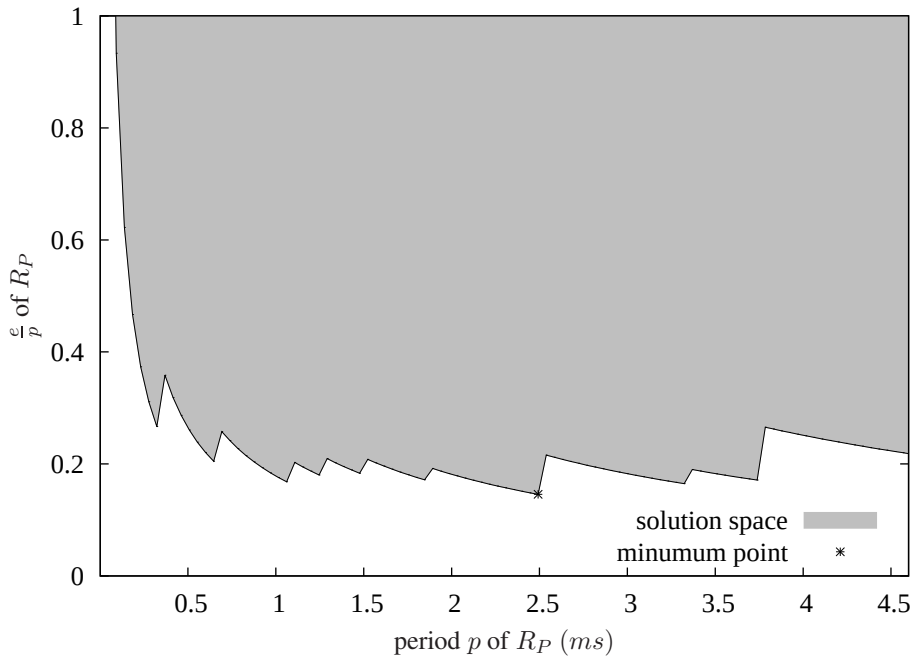

Figure 8: Solution space of $R_P(p, e)$

Table 1: An example of execution time table from [2]

mode m	1	2	3	4	5	6
rpm	6500	5500	4500	3500	2500	1500
$C^m(\mu s)$	246	277	343	424	576	965

task of Table. 1. In this example, our method selects the marked minimum point as $R_P(p, e)$. Since the selection of minimum utilization of a task in the task set do not guarantee the minimum utilization of PRM, i.e., Θ/Π, our method is not an optimal in this dimension. We discuss the difference between the result of our method and the optimal utilization of PRM, which is derived from the much more time-consuming exhaustive search in subsection 5.

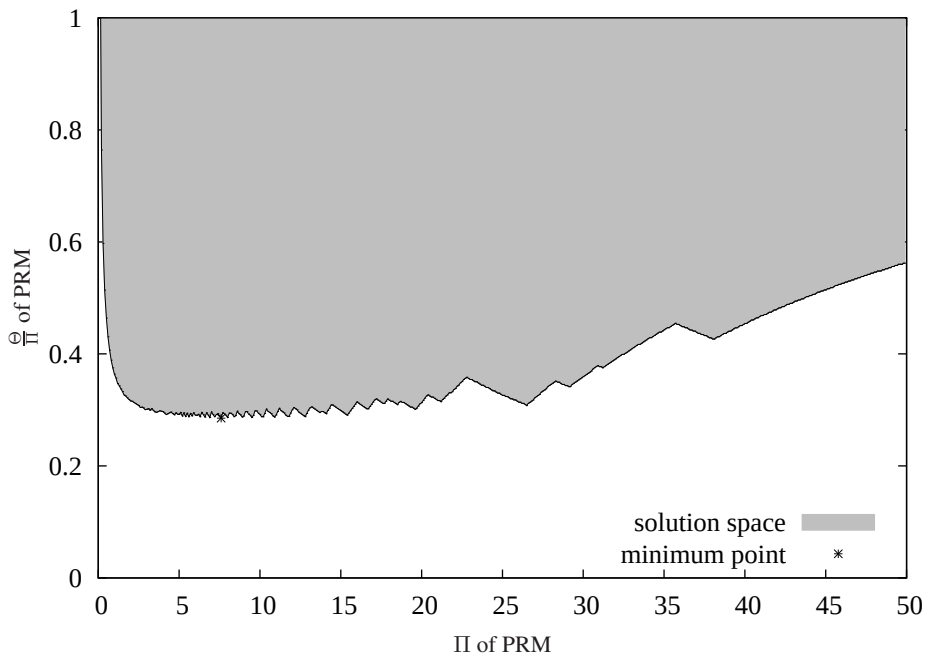

Figure 9: PRM parameter optimization with context switch overhead $50\mu s$

4.3 Optimizing PRM Parameters

Now we have W^* which consists of periodic tasks and reduced tasks, i.e., $W^* = \{P\} \cup \{R_P\}$. We conduct exhaustive search to select Π and Θ of PRM to schedule W^* effectively. To serve more practical requirement, we consider the context switch overhead by adding $2 \times CS$ to execution time of every periodic task [13] and also to the Θ of PRM component interface [7].

Fig. 9 shows PRM parameter optimization of the task set $\{P(50, 7), P(75, 9)\}$ with assuming context switch overhead $CS = 20\mu s$. Likewise, we select the marked minimum point as PRM $\Gamma(\Pi, \Theta)$.

5 Experiment

We evaluate our method in simulation study with randomly generated periodic tasks and AVR tasks. The number of periodic task n_P is uniformly selected from $\{4, 5, ..., 10\}$. For each periodic task, the period p_i is uniform-randomly selected in the range of $[1, 20]ms$. The execution time is uniform-randomly selected in the range of $(0,\ p_i \times U^P_{max}]$ ms, where $U^P_{max} = \frac{0.3}{n_P}$ to suppress the sum of utilization of periodic tasks as much as 0.3.

The number of modes in an AVR task M_i is uniform-randomly selected from $\{3, 4, ..., 8\}$. For each mode, the execution time is randomly generated in according to two aspects, i) the maximum utilization is bounded to $U^R_{bound} = 0.1$ and ii) the execution time is monotonic decreasing as the mode decreased. By the second aspect, the peak utilization of an AVR task is reduced as the number of modes M increasing. The maximum angular velocity is set to $6000rpm$ and the minimum angular velocity is set to $600rpm$ as [2].

In the simulation study, we compare the following three methods:

- Our method: an AVR task is reduced to a periodic task (p, e) with having minimum utilization, named R_{min}.
- Naive method: an AVR task is reduced to a sporadic task with the minimum inter-release time and the maximum execution time among the all execution modes, denoted as R_{naive}.

- Binary method: as an enhanced version of the naive method, an AVR task is firstly reduced to a sporadic task as same as the naive method and try to minimize the utilization e/p as possible, by means of dividing e or p by 2 recursively as the schedulability test can success until they reach preset minimum values, similar to the binary search. We call it R_{bin}.

First, we compare the utilization of single reduced task R_{min}, R_{naive}, and R_{bin}. For 100 randomly generated AVR tasks, we reduced them to R_{min}, R_{naive}, and R_{bin} and calculated the average utilization of 100 pairs, respectively. The average utilization of R_{min} is 0.0316 and the average utilization of R_{naive} is 0.0474, and R_{bin} results in 0.0355. R_{bin} has 25% less utilization than the utilization of R_{naive} and R_{min} has 33% less utilization than the utilization of R_{naive} in average. Our method presents the least utilization since the method conducts an exhaustive search to find the minimum utilization.

Second, we compare the capacity of PRM $\frac{\Theta}{\Pi}$ that is derived from the reduced task set with R_{min}, R_{naive}, and R_{bin}. In detail for the task set, we randomly generate a set of periodic tasks. In addition to the task set, we generate n_R AVR tasks, where $n_R = 1, 2, ..., 10$. And we reduce the AVR tasks to R_{min}, R_{naive}, and R_{bin}. With this reduced task set, we evaluate the optimal PRM parameters with considering context switch overhead. Fig. 10 displays the result of Θ/Π as changing the number of AVR tasks in the task set. When n_R is small, the Θ/Π gap between two methods are

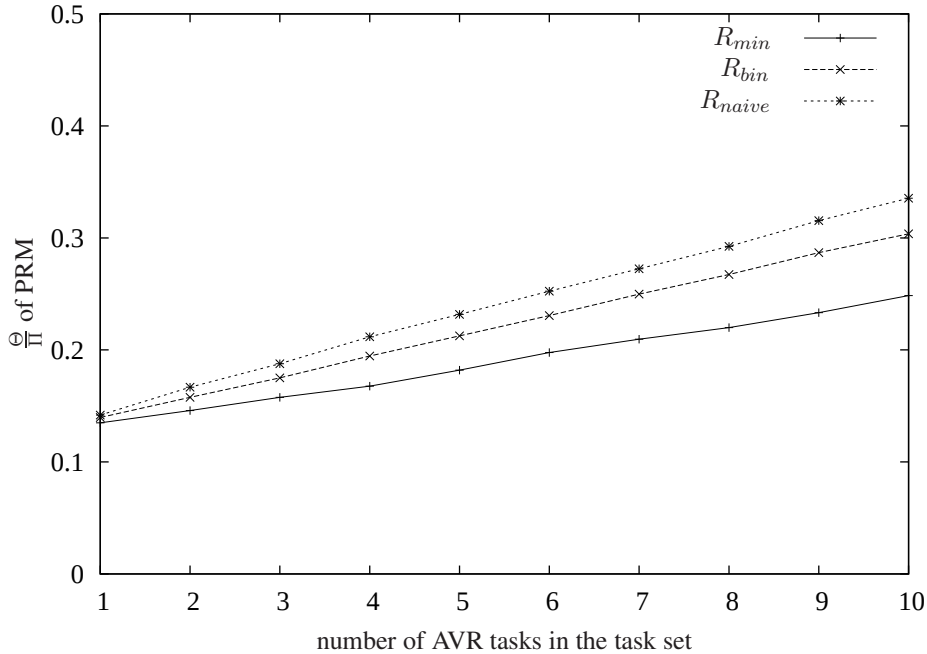

Figure 10: Utilization of $\Gamma(\Pi, \Theta)$ by changing number of AVR tasks

relatively small. But as the number of AVR tasks increasing, the gap increases proportional to the number of AVR tasks. Our method shows the least utilization Θ/Π for all the number of AVR tasks in the task set.

Third, we compare the utilization e/p for a single reduced task by changing the number of modes in an AVR task. For each M, we randomly generate 100 AVR tasks which having M modes with the maximum execution time bound and the minimum execution time bound. Fig. 11 shows the result for the three methods. The average of utilization e/p for 100 tasks are marked on the y-axis. Since the maximum execution time is fixed, R_{naive} selects the maximum execution time as e. And the minimum inter-arrival time is also given because the ω_{max} is fixed. Therefore, R_{naive} always

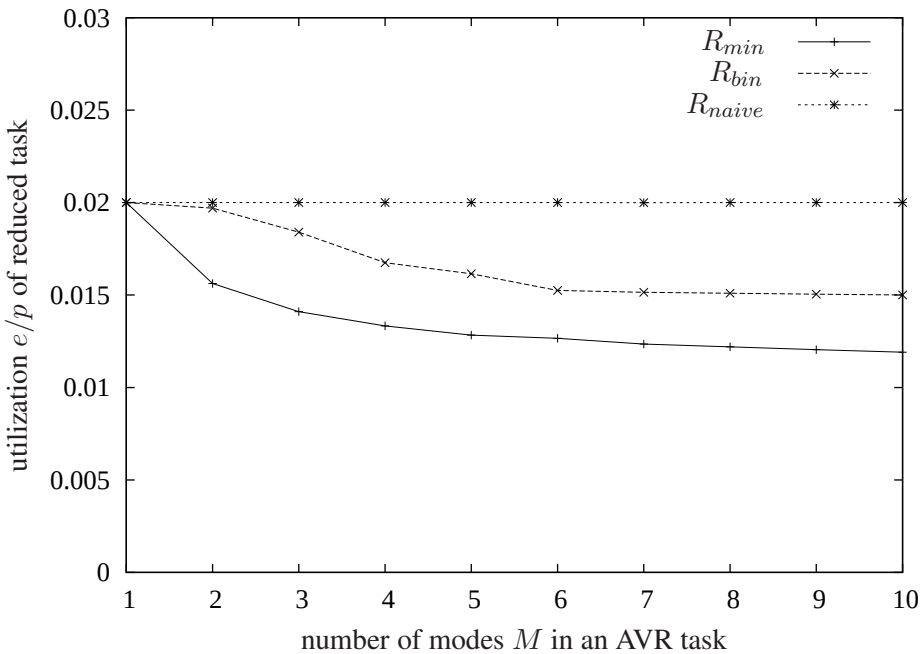

Figure 11: Utilization of reduced tasks as changing the number of modes in an AVR task

selects the same (p, e) and stays fixed e/p value regardless of the number of modes in an AVR task. On the other hand, R_{min} and R_{bin} can enjoy more chance to reduce e/p when the number of modes increases, by selecting minimum execution time over the possible p. When the number of mode is 1, the three methods results in the same value because there is no other choice for the (p, e). Note that the increasing of the number of modes yields the decreasing of total utilization of the AVR task since the execution time of mode m should be always lower than the execution time of mode $(m - 1)$. That property contributes to the overall e/p optimizing of R_{min} and R_{bin}. Our method always shows less or equal utilization than the result from other two methods since the method pays more computation time to find the minimum utilization parameters.

Fourth, we evaluate the pessimism of our method in the dimension of the utilization of PRM, i.e., Θ/Π. Our method selects the (p, e) pair with having minimum utilization, the time complexity of this kind of exhaustive search depends on the number of quantization step of the period and execution time. In addition to that, we can derive the optimal minimum value of Θ/Π in more time-consuming way by which piling up another domain of exhaustive search for the Θ and Π on our method, with enduring much larger time-consuming calculations. Since the time complexity for deriving the optimal Θ/Π is particularly large, we measure the computation time for 100 randomly-generated small task sets with having only one AVR task. To run the

experiments, two methods were running on four-core Intel i5 at 3.3GHz. Our method presents Θ/Π =0.1416 in 69ms and the optimal method shows Θ/Π =0.1427 in 6235ms. The difference between the result from our method and the optimal value is 0.8% and considered relatively small. However, the computation time difference between two methods is significantly large.

6 Conclusion

We proposed a effective method for scheduling AVR tasks on the PRM to leverage system integration. For doing that, we present a heuristic method that reduce an AVR task to a periodic task. And we find optimal PRM parameters for the workload which consists of periodic tasks and reduced AVR tasks. With our method, the AVR task is more effectively scheduled on the PRM compared with the previously used naive method. The simulation results confirm that our method finds (p, e) which have approximately 33% less utilization compared with the result from naive method.

As future work, we plan to extend our method for fixed-priority scheduling, not only under the EDF scheduling policy. In addition, we will find a method of exact schedulability condition for AVR task on PRM, without reducing the AVR task to a periodic task.

References

[1] Mikael Asberg, Moris Behnam, Farhang Nemati, and Thomas Nolte. Towards hierarchical scheduling in autosar. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2009.

[2] Alessandro Biondi, Alessandra Melani, Mauro Marinoni, Marco Di Natale, and Giorgio Buttazzo. Exact interference of adaptive variable-rate tasks under fixed-priority scheduling. In *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS)*, 2014.

[3] Giorgio C Buttazzo, Enrico Bini, and Darren Buttle. Rate-adaptive tasks: Model, analysis, and design issues. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2014.

[4] D Buttle. Real-time in the prime-time. In *Kyenote speech given at the 24th Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[5] Robert I Davis, Timo Feld, Victor Pollex, and Frank Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *Proceedings of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.

[6] Zhong Deng and Jane W-S Liu. Scheduling real-time applications in an open environment. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 308–319. IEEE, 1997.

[7] Arvind Easwaran, Insup Lee, Insik Shin, and Oleg Sokolsky. Compositional schedulability analysis of hierarchical real-time systems. In *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2007.

[8] Xiang Feng and Aloysius K Mok. A model of hierarchical real-time virtual resources. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 26–35. IEEE, 2002.

[9] Pawan Goyal, Xingang Guo, and Harrick M Vin. A hierarchical cpu scheduler for multimedia operating systems. In *OSDI*, volume 96, pages 107–121, 1996.

[10] Junsung Kim, Karthik Lakshmanan, and Ragunathan Raj Rajkumar. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *Proceedings of the IEEE/ACM Third International Conference on Cyber-Physical Systems*, 2012.

[11] Tei-Wei Kuo and Ching-Hui Li. A fixed-priority-driven open environment for real-time applications. In *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*, pages 256–267. IEEE, 1999.

[12] Giuseppe Lipari and Sanjoy K Baruah. Efficient scheduling of real-time multi-task applications in dynamic systems. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 166–166. IEEE Computer Society, 2000.

[13] Jane WS Lui. Real-time systems, 2000.

[14] Aloysius K Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE*, pages 75–84. IEEE, 2001.

[15] Victor Pollex, Timo Feld, Frank Slomka, Ulrich Margull, Ralph Mader, and Gerhard Wirrer. Sufficient real-time analysis for an engine control unit with constant angular velocities. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2013.

[16] Saowanee Saewong, Ragunathan Raj Rajkumar, John P Lehoczky, and Mark H Klein. Analysis of hierarchical fixed-priority scheduling. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 173–173. IEEE Computer Society, 2002.

[17] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the 24th IEEE Real-Time Systems Symposium (RTSS)*, 2003.

[18] Insik Shin and Insup Lee. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):30, 2008.

요약(국문초록)

자동차 시스템을 위한
합성적 실시간 스케줄러를 이용한
적응형 가변주기 태스크의 스케줄링

주기적 자원 모델(periodic resource model)은 주기적 태스크(periodic task)들로 구성된 여러 가지 실시간 시스템들이 하나의 하드웨어를 공유하되 시간적으로 분리하여 사용하는 방법으로 시스템의 통합을 제공하는 합성적 실시간 스케줄러(compositional real-time scheduler)이다. 동시에, 최근 자동차 응용 프로그램(예를 들어, 엔진 조절 태스크)에 관한 연구들은 조절 대상의 상태값(예를 들어, 크랭크축의 회전 각도)에 따라 그 주기를 조절하는 적응형 가변주기 태스크(adaptive variable-rate task)를 제안하고 있다. 주기적 자원 모델에서 주기적 태스크와 산발적 태스크(sporadic task)를 스케줄 하는 방법들은 연구되었지만, 적응형 가변주기 태스크를 주기적 자원 모델에서 스케줄하는 연구는 아직 수행되지 않았기에, 그를 위한 새로운 방법이 필요하다.

이 논문에서는 적응형 가변주기 태스크를 주기적 자원 모델에서 효율적으로 스케줄링 할 수 있는 방법을 제안한다. 우리의 방법은 적응형 가변주기 태스크를 주기적 태스크로 변환하는 것으로 접근하며, 우리의 방법으로 변환된 태스크가 주기적 자원 모델에서 스케줄 가능하다면

원래의 적응형 가변주기 태스크 또한 주기적 자원 모델에서 스케줄 가능함을 보인다. 그리고 우리의 방법으로 변환된 태스크를 주기적 자원 모델이 기존에 제공하고 있던 주기적 태스크를 스케줄하는 방법을 그대로 적용하여 스케줄 할 수 있다. 그 후, 문맥변환비용 (context switch overhead)을 고려하여, 주기적 태스크와 적응형 가변주기 태스크들로 이루어진 태스크 집합을 효율적으로 스케줄 할 수 있는 주기적 자원 모델의 최적화된 인자를 찾는다. 우리가 조사한 해온 바에 따르면, 이것은 적응형 가변주기 태스크를 주기적 자원 모델에서 스케줄하는 최초의 연구이다.

주요어 : 임베디드 시스템, 실시간 시스템, 자동차, 엔진 응용 프로그램, 적응형 가변주기 태스크
학 번 : 2012-23233