Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Schedulability analysis for linear transactions under fixed priority hybrid scheduling*

Zhi-gang GAO†, Zhao-hui WU

(*School of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*)

†E-mail: gaozhigang@zju.edu.cn

Received July 26, 2007; revision accepted Jan. 30, 2008; published online May 5, 2008

Abstract: In hard real-time systems, schedulability analysis is not only one of the important means of guaranteeing the timelines of embedded software but also one of the fundamental theories of applying other new techniques, such as energy savings and fault tolerance. However, most of the existing schedulability analysis methods assume that schedulers use preemptive scheduling or non-preemptive scheduling. In this paper, we present a schedulability analysis method, i.e., the worst-case hybrid scheduling (WCHS) algorithm, which considers the influence of release jitters of transactions and extends schedulability analysis theory to timing analysis of linear transactions under fixed priority hybrid scheduling. To the best of our knowledge, this method is the first one on timing analysis of linear transactions under hybrid scheduling. An example is employed to demonstrate the use of this method. Experiments show that this method has lower computational complexity while keeping correctness, and that hybrid scheduling has little influence on the average worst-case response time (WCRT), but a negative impact on the schedulability of systems.

Key words: Real-time systems, Hybrid scheduling, Linear transactions, Worst-case response time (WCRT), Schedulability analysis

doi:10.1631/jzus.A071411 **Document code:** A **CLC number:** TP316

INTRODUCTION

In hard real-time embedded systems, timing requirements of software must be respected. Schedulability analysis is one of the most important means for guaranteeing the timelines of the embedded software. Today, such new techniques as energy savings and fault tolerance have flourished in the embedded field. Because most of the new techniques influence the running time of the software, the schedulability analysis becomes one of the fundamental theories as these new techniques are applied to hard real-time embedded systems.

The fixed priority scheduling has been widely used in embedded operating systems because of its simplicity and lower scheduling overheads. In hybrid scheduling, tasks have fixed priorities. Additionally, tasks can be preemptive or non-preemptive in order to reduce the overheads of systems and make resource synchronization easy to realize (Jeffay *et al.*, 1991). Hybrid scheduling is more general than preemptive scheduling and non-preemptive scheduling. Today, hybrid scheduling has been used in some hard real-time operating systems, e.g., it is one of the scheduling modes supported by OSEK/VXD (OSEK, 2003), a widely accepted standard in the automotive electronic industry.

A transaction (Damm *et al.*, 1989) is a sequence of related tasks. Compared to independent tasks, transactions can work well in modeling control process, improving control effect and schedulability (ARTIST2, 2005). Currently, there are many research efforts on the schedulability analysis of fixed priority preemptive scheduling (Liu and Layland, 1973; Gonzalez Harbour *et al.*, 1994; Tindell, 1994; Palen-

* Project supported by the National Natural Science Foundation of China (No. 60533040), the Hi-Tech Research and Development Program (863) of China (Nos. 2007AA010304 and 2007AA01Z129), and the Key Scientific and Technological Project of Hangzhou Technology Bureau, China (No. 20062412B01)

cia and Gonzalez Harbour, 1999; Redell, 2004; Henia and Ernst, 2005; Fisher *et al.*, 2007; Yomsi and Sorel, 2007). However, most of the research efforts on the fixed priority non-preemptive scheduling (Jeffay *et al.*, 1991; Khil *et al.*, 1997; Dolev and Keizelman, 1999; Baruah and Chakraborty, 2006) focus on the schedulability analysis of independent tasks under EDF (earliest deadline first) (Buttazzo, 1995). Wang and Wu (2004) proposed a schedulability analysis method on fixed priority hybrid scheduling. However, their method is only suitable for independent tasks. In this paper we extend the schedulability analysis method proposed by Gonzalez Harbour *et al.*(1994), and present a schedulability analysis method for linear transactions under fixed priority hybrid scheduling on a single processor—the WCHS (worst-case hybrid scheduling) algorithm. WCHS considers the release jitters of transactions and extends schedulability analysis to hybrid scheduling. Experiments show that WCHS has lower computational complexity while keeping correctness, and that hybrid scheduling has little influence on the average worst-case response time (WCRT) besides a negative impact on the schedulability of transactions.

RELATED WORK

Currently, there have been many research efforts focusing on the schedulability analysis of transactions. Tindell (1994) proposed a schedulability analysis method, which uses static offsets to describe the precedence constraints among tasks in a transaction. Palencia and Gonzalez Harbour (1999) extended the schedulability analysis method of Tindell and presented the WCDO (worst-case dynamic offsets) algorithm. WCDO introduces dynamic offsets and jitters to describe the influence of a task's response time variation on its succeeding tasks. Palencia and Gonzalez Harbour (1998) presented the WCDOPS algorithm, which considers the execution order of tasks among different jobs of a transaction. Redell (2004) extended the WCDOPS algorithm and proposed the WCDOPS+ algorithm, which considers not only the influence of the execution order of tasks in different jobs of a transaction, but also the influence of the priority structure of tasks on the execution order of tasks. Jiang (2006) presented a decoupled scheduling approach for distributed hard real-time embedded automotive systems that have tasks with precedence constraints, and his research efforts focus on the extensibility of scheduling.

In the task model with precedence relationship, the release time of tasks is described by using offsets and jitters. A transaction's WCRT is obtained by analyzing the WCRT of its tasks sequentially.

The HKL algorithm presented by Gonzalez Harbour *et al.*(1994) makes use of the canonical forms of tasks ["tasks" and "subtasks" in (Gonzalez Harbour *et al.*, 1994) are respectively equal to "transactions" and "tasks" in logic] to simplify the response time analysis of tasks. However, HKL assumes that the release jitters of transactions are zero and that the tasks are preemptive, which constrains its application.

SCHEDULABILITY ANALYSIS UNDER FIXED PRIORITY HYBRID SCHEDULING

Computational model and basic notation

There are n periodic transactions $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ in the system. A transaction Γ_i has the period T_i, WCET (worst-case execution time) C_i, and deadline D_i. Γ_i is composed of m tasks, $\tau_{i1}, \tau_{i2}, \ldots, \tau_{im}$ ($m \geq 1$). The task τ_{ik} ($m \geq k \geq 1$) has the priority P_{ik}, WCET C_{ik}, and period T_i. If τ_{ik} is a preemptive task, it can be preempted by other high-priority tasks. If τ_{ik} is a non-preemptive task, it can only be preempted before it is executed; but once τ_{ik} is executed, it cannot be interrupted by other tasks until it is completed. Γ_i is activated by a periodic external event (i.e., Γ_i arrives). The delay between Γ_i's activation instant a_i and its release instant r_i is its release jitter, denoted as J_i ($=r_i-a_i$), and $J_i<T_i$. When τ_{i1} is completed, it sends a message to τ_{i2}. τ_{i2} releases itself immediately when it receives the message from τ_{i1}. When τ_{i2} is completed, it sends a message to τ_{i3}. This process is repeated until the last task τ_{im} of Γ_i is completed. D_i may be less than, larger than, or equal to T_i. One instance of Γ_i activated by an external event is called a job of Γ_i. In this paper, we assume that different jobs of a transaction are executed sequentially (which is very common in control systems), i.e., the $(j+1)$th job will be permitted to be executed only after the jth job is completed. Moreover, we do not consider the blocked time caused by accessing mutually exclusive resources. The response time of the jth job of Γ_i ($\Gamma_{j,i}$), $R_{j,i}$, is the time interval between $\Gamma_{j,i}$'s

activation instant and completion instant. The maximum response time of all jobs of Γ_i is called its WCRT. The goal of the schedulability analysis is to decide whether all transactions meet their deadlines, i.e., their WCRT is no longer than their deadlines.

$P_{\min(m)}$ denotes the least priority of the tasks in Γ_m. When $P_{\min(m)} \geq P_{ij}$, Γ_m has multiply preemptive effect on τ_{ij}. When $\exists k, (P_{m1}, \cdots, P_{mk} \geq P_{ij}) \wedge (P_{m(k+1)} < P_{ij})$, Γ_m has singly preemptive effect on τ_{ij}. When $\exists k, l$, $(P_{mk} < P_{ij}) \wedge (P_{m(k+1)}, \cdots, P_{ml} \geq P_{ij}) \wedge (P_{m(l+1)} < P_{ij})$, Γ_m has blocking effect on τ_{ij}.

If a transaction consists of consecutive tasks whose priorities do not decrease, the transaction is said to be in canonical form. From the proof of (Gonzalez Harbour *et al.*, 1994), we know the response time of Γ_i is equal to that of its canonical form, Γ_i'. If Γ_m is not in canonical form, it can be transformed into its canonical form Γ_m' by using the algorithm of (Gonzalez Harbour *et al.*, 1994). A canonical form task τ_{ik}' may consist of multiple tasks of Γ_i.

Length of the busy period

Lehoczky (1990) introduced the concept of the busy period to derive the WCRT of a task under fixed priority scheduling with arbitrary deadlines. Gonzalez Harbour *et al.*(1994) showed no matter how long the deadlines of the tasks are, it is necessary to use the "busy period" to analyze the WCRT of tasks with multiple subtasks.

If L_i is a time interval during which only the tasks whose priorities are larger than or equal to $P_{\min(i)}$ exist, and all jobs of Γ_i released during L_i are all completed during L_i, L_i is called a Γ_i-busy period.

In the following discussion, we use Γ_i as an example to explain how to calculate its busy period. For readability reasons, in most cases, we use "busy period" to denote "Γ_i-busy period".

Lehoczky (1990) proposed that the longest response time of a task can be obtained from the jobs of the task in a busy period, and that the maximum response time in all the jobs in the Γ_i-busy period is the WCRT of Γ_i. We must identify the length of the Γ_i-busy period to obtain the job number in the Γ_i-busy period before we calculate the response time of each job of Γ_i.

In fact, multiply preemptive effect, singly preemptive effect and blocking effect on τ_{ij} are all relevant to the Γ_i-busy period: if Γ_m has multiply preemptive effect on τ_{ij}, Γ_m may be executed multiple times (i.e., multiple jobs of Γ_m may be activated) during the Γ_i-busy period and they may preempt τ_{ij} multiple times; if Γ_m has singly preemptive effect on τ_{ij}, Γ_m is executed only once (i.e., one job of Γ_m is activated) during the Γ_i-busy period; if Γ_m has blocking effect on τ_{ij}, Γ_m may block Γ_i once during the Γ_i-busy period.

When creating the busy period, the other transactions need to be classified according to $P_{\min(i)}$ in order to find their maximum contribution to the busy period. In fact, because $P_{\min(i)}$ is equal to P_{i1}', classifying the types of transactions according to $P_{\min(i)}$ is equal to that according to P_{i1}'. To use the notations of transaction types in a uniform manner, in the following discussion about the busy period, unless otherwise stated, we use P_{i1}' instead of $P_{\min(i)}$, τ_{i1}' instead of Γ_i. If the priority of P_{i1}' is 3, a transaction with the priority sequence (6, 5, 1, 2, 9) is classified to be task segments (H, L, H), where H denotes one or more tasks whose priorities are larger than or equal to P_{i1}', L denotes one or more tasks whose priorities are less than P_{i1}'. A transaction other than Γ_i can be classified into one of the following five types:

(1) Type-1 transaction, i.e., (H) transaction. All tasks' priorities of a type-1 transaction are equal to or higher than P_{i1}', and may preempt τ_{i1}' more than once during the busy period.

(2) Type-2 transaction, i.e., $((HL)^+)$ transaction (+ denotes equal to or more than one). A type-2 transaction is composed of H segments followed by L segments. Usually, every type-2 transaction uses its initial H segment to preempt τ_{i1}' once during the busy period. In some special cases, an internal H segment can also block τ_{i1}' once. A type-2 transaction can exhibit singly preemptive effect or blocking effect.

(3) Type-3 transaction, i.e., $((HL)^+H)$ transaction. A type-3 transaction differs from a type-2 transaction in that it ends with an H segment. Like type-2 transactions, it usually exhibits singly preemptive effect. In some special cases, it can exhibit blocking effect.

(4) Type-4 transaction, i.e., $((LH)^+L^0)$ transaction (0 denotes equal to zero or one). A type-4 transaction is composed of L segments followed by H segments. No or one L segment lies at the end of a

type-4 transaction. It can only exhibit blocking effect.

(5) Type-5 transaction, i.e., (L) transaction. All tasks' priorities of a type-5 transaction are lower than P'_{i1}, and it has no effect on the completion time of τ'_{i1}.

Note that type-2 and type-3 transactions contribute singly preemptive effect under the default condition. In some special cases, a type-2 or type-3 transaction may contribute the blocking effect. The blocking effect contributed by type-2, type-3 and type-4 transactions is caused by H segments, and we call it task blocking. Under hybrid scheduling, non-preemptive tasks can also contribute the blocking effect, which is called the non-preemptive blocking.

After transactions are classified, in order to deal with the non-preemptive blocking, we perform the following process. Let's use τ_{jp} as an example. If τ_{jp} ($j\neq i$) is a non-preemptive task with $P_{jp}\geq P'_{i1}$, τ_{jp} will be classified into an H segment, so we need not consider its non-preemptive blocking effect. When $P_{jp}<P'_{i1}$, if τ_{jp} is the immediate predecessor of H_{jk}, an internal H segment or final H segment of Γ_j, τ_{jp} should be merged with H_{jk}, i.e., transformed into a new H segment H'_{jk} which consists of τ_{jp} and H_{jk}; otherwise, transform τ_{jp} into an H segment. Note that if τ_{jq}, …, τ_{jp} are consecutive non-preemptive tasks with $P_{jr}<P'_{i1}$ ($p\geq r\geq q$) before H_{jk}, only τ_{jp} can be merged with H_{jk} because preemption can occur between non-preemptive tasks. Because of non-preemptive blocking, type-2 transactions may add a final H segment and/or internal H segments. With respect to a type-4 or type-5 transaction, it may add internal H segments wherever non-preemptive tasks lie. After the above process, all non-preemptive blockings are merged into internal H segments or final H segments of transactions.

When calculating the blocked time of Γ_i, we should consider the maximum blocking effect from the internal H segments and final H segments of type-2 and type-3 transactions, together with the internal H segments of type-4 and type-5 transactions. By extending the method of blocking time calculation in (Gonzalez Harbour *et al.*, 1994), we can derive the blocked time of a transaction. Assuming the H segment with the longest WCET in all type-4 and type-5 transactions is $B'_{4,5}$, and Γ_p is a type-2 or type-3 transaction with the initial H segment's WCET F_p, the maximum internal H segment's WCET M_p, and the final H segment's WCET L_p. If there is no internal H segment or final H segment, M_p or L_p is equal to zero. The maximum blocking time caused by type-2 and type-3 transactions is

$$B_{2,3}=\max_{\Gamma_p\in T23_{i1}}\left(\max\left(M_p-F_p-B'_{4,5},L_p-B'_{4,5}\right)\right), \quad (1)$$

where $T23_{i1}$ denotes the set of type-2 and type-3 transactions of τ'_{i1}. If $B_{2,3}$ is no more than zero, the blocked time of τ'_{i1}, i.e. B_i, is equal to $B'_{4,5}$. Otherwise, the maximum blocking time is caused by a type-2 or type-3 transaction. Assuming that the transaction Γ_b in $T23_{i1}$ causes the maximum blocking effect, if $M_b-F_b>L_b$, the blocking time is caused by an internal H segment. Set B_i to be M_b and change the transaction type of Γ_b to be a type-5 transaction. Otherwise, set B_i to be L_b and keep the transaction type of Γ_b unchanged.

In the HKL algorithm, the critical instant of Γ_i is defined as the instant when Γ_i is released with its multiply preemptive tasks and singly preemptive tasks simultaneously, and suffers from the maximum blocked time. In this paper, because of release jitters of transactions, the definition of critical instant should be redefined to make the other transactions have the maximum interference in Γ_i.

Assuming that Γ_p is a type-1 transaction of Γ_i, and t_c is the instant when a job of Γ_i is released. Γ_p has the capability of multiple preempting Γ_i. According to the interference time analysis of (Audsley *et al.*, 1993) for tasks with release jitters, we can derive that Γ_p has the maximum preemption time for Γ_i during the busy period when the jobs of Γ_i and Γ_p before t_c undergo the maximum release jitters and are released at t_c simultaneously, and the jobs of Γ_i and Γ_p after t_c have the release jitters of zero. This scenario is shown in Fig.1, where $a_{m,n}$ and $r_{m,n}$ denote the arrival instant and the release instant of the mth job of Γ_n, respectively.

If Γ_p is a type-2, type-3, type-4, or type-5 transaction, it may be a singly preemptive transaction or a blocking transaction, i.e., Γ_p can be executed once at most in the busy period. Γ_p will contribute the maximum time to the busy period if it is released at t_c when it is a singly preemptive transaction, or its maximum blocking H segment is released at t_c when it is a blocking transaction. The t_c constructed as above is the critical instant of Γ_i.

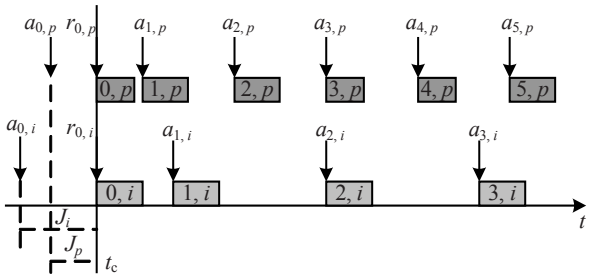

Fig.1 Maximum preemption effect of Γ_p on Γ_i

Because of the release jitters, there will be more jobs of transactions in the busy period of Γ_i. Considering the release jitters, we extend the definitions of the length of the busy period in (Gonzalez Harbour *et al.*, 1994) and the definitions of the job number of Γ_i in the busy period.

The length of the busy period is defined as

$$L_i=\min\left\{t>0 \mid t=B_i+\sum_{\Gamma_p\in MP_{i1}}\left\lceil (t+J_p)/T_p\right\rceil C_p + \sum_{\Gamma_p\in SP_{i1}}\left(C_p^{\mathrm{h}}+\left\lceil (t+J_p)/T_i\right\rceil C_i\right)\right\}, \tag{2}$$

where B_i is the blocked time that Γ_i suffers; MP_{i1} is the set of transactions which exhibits multiply preemptive effect on τ'_{i1}; the term in which MP_{i1} lies denotes the multiply preemptive time that Γ_i suffers in the busy period; SP_{i1} is the set of transactions which exhibits singly preemptive effect on τ'_{i1}; C_p^{h} is the WCET of the initial H segment of Γ_p; the term in which SP_{i1} lies denotes the singly preemptive time that Γ_i suffers in the busy period; the last term in the right of Eq.(2) denotes the total execution time of all the jobs of Γ_i in the busy period. Eq.(2) can be solved by iteration.

The job number of Γ_i in the busy period is

$$N_i=\left\lceil (L_i+J_i)/T_i\right\rceil. \tag{3}$$

WCRT of a transaction

The WCRT of Γ_i is equal to the maximum response time of all jobs of Γ_i in the busy period. When calculating the response time of the kth job of Γ_i, first transform Γ_i into its canonical form Γ_i', and then calculate the completion time of each canonical form task τ'_{ij} sequentially. According to the completion time of the last canonical form task of the kth job of Γ_i, we can obtain the response time of the kth job of Γ_i.

Assuming that τ'_{ij} consists of one or more task(s), we first prove a lemma before calculating the completion time of τ'_{ij}.

Lemma 1 If the last task of τ'_{ij} is preemptive, the completion time of τ'_{ij} will not be affected by the preemption properties of the other task(s) it includes.

Proof In the response time of τ'_{ij}, if j=1, the interference time from other transactions includes three parts: multiply preemptive time, singly preemptive time and blocked time; if j>1, the interference time from other transactions only includes multiply preemptive time and singly preemptive time. Because blocking always occurs before τ'_{ij} is executed, the blocked time that τ'_{ij} suffers is not affected by the preemption properties of tasks that τ'_{ij} includes. Assume that the last task τ'_{ij} includes is τ_e. Γ_p is a multiply preemptive or singly preemptive transaction, and it is released before τ_e. From the transformation algorithm of canonical form tasks of (Gonzalez Harbour *et al.*, 1994), we know that the priority of τ_e is equal to P'_{ij}, i.e., τ_e has the least priority in all tasks included in τ'_{ij}. Whether the task(s) included in τ'_{ij} is/are preemptive or not, even in the worst case, Γ_p will be executed by preempting τ_e after τ_e is released. If Γ_p is released between the release instant and completion instant of τ_e, it will be executed by preempting τ_e. From the above discussion, we can conclude that all tasks which have interference effect on τ'_{ij} can exhibit their interference effect between the release instant and completion instant of τ'_{ij}. Therefore if τ_e is preemptive, the response time of τ'_{ij} will not be affected by the preemption properties of the other task(s) it includes.

We number the first job of Γ_i in the busy period to be 1, and the subsequent jobs to be 2, 3, and so on. Assume the instant at which the critical instant occurs to be zero, and the completion time is the time span from the critical instant on. For simplicity, in the following parts, we assume C_{ij} and C_p refer to the WCET of τ'_{ij} and Γ_p', respectively.

According to the computational model, we extend the algorithm presented by Gonzalez Harbour *et al.*(1994) by incorporating release jitters and non-preemptive tasks, and develop the WCHS algorithm to analyze the completion time of transactions under hybrid scheduling.

We first calculate the completion time of τ'_{i1} in the kth job of Γ_i, $\tau'_{k,i1}$.

If the last task of $\tau'_{k,i1}$ is a preemptive task, the completion time of $\tau'_{k,i1}$ is

$$E_{k,i1}=\min\left\{t>0 \mid t=B_i+\sum_{\Gamma_p\in MP_{i1}}\left\lceil (t+J_p)/T_p\right\rceil C_p + \sum_{\Gamma_p\in SP_{i1}} C_p^{\mathrm{h}}+C_{i1}+(k-1)C_i\right\}. \tag{4}$$

If the last task of $\tau'_{k,i1}$, $\tau_{\mathrm{e}1}$, is a non-preemptive task with the WCET of $C_{\mathrm{e}1}$, the completion time of $\tau'_{k,i1}$ is

$$E_{k,i1}=W_{k,i1}+C_{\mathrm{e}1}, \tag{5}$$

where

$$W_{k,i1}=\min\left\{t>0 \mid t=B_i+\sum_{\Gamma_p\in MP_{i1}}\left\lceil (t+J_p)/T_p\right\rceil C_p + \sum_{\Gamma_p\in SP_{i1}} C_p^{\mathrm{h}}+C_{i1}-C_{\mathrm{e}1}+(k-1)C_i\right\}, \tag{6}$$

where $W_{k,i1}$ is the waiting time of $\tau'_{k,i1}$.

After obtaining the completion time of the first task, we should adjust the types of transactions because $\tau'_{k,i2}$ has a higher priority. The multiply preemptive tasks of $\tau'_{k,i2}$ can be classified according to the priority level of $\tau'_{k,i2}$. The singly preemptive task of $\tau'_{k,i2}$ is a subset of MP_{i1}, denoted as

$$SP_{i2}=\left\{\Gamma_p \mid \Gamma_p\in(MP_{i1}-MP_{i2})\wedge\left(\exists k,(P_{p1},\cdots,P_{pk}\geq P'_{i2})\wedge(P_{p(k+1)}<P'_{i2})\right)\right\}. \tag{7}$$

If the last task of $\tau'_{k,i2}$ is a preemptive task, the completion time of $\tau'_{k,i2}$ is

$$E_{k,i2}=\min\left\{t>0 \mid t=E_{k,i1}+C_{i2}+\sum_{\Gamma_p\in MP_{i2}}\left[\left\lceil (t+J_p)/T_p\right\rceil-\left\lceil (E_{k,i1}+J_p)/T_p\right\rceil\right]C_p+\sum_{\Gamma_p\in SP_{i2}}\min\left(1,\left[\left\lceil (t+J_p)/T_p\right\rceil-\left\lceil (E_{k,i1}+J_p)/T_p\right\rceil\right]C_p^{\mathrm{h}}\right)\right\}. \tag{8}$$

If the last task of $\tau'_{k,i2}$, $\tau_{\mathrm{e}2}$, is a non-preemptive task with the WCET of $C_{\mathrm{e}2}$, the completion time of $\tau'_{k,i2}$ is

$$E_{k,i2}=W_{k,i2}+C_{\mathrm{e}2}, \tag{9}$$

where

$$W_{k,i2}=\min\left\{t>0 \mid t=E_{k,i1}+C_{i2}-C_{\mathrm{e}2}+\sum_{\Gamma_p\in MP_{i2}}\left[\left\lceil (t+J_p)/T_p\right\rceil-\left\lceil (E_{k,i1}+J_p)/T_p\right\rceil\right]C_p+\sum_{\Gamma_p\in SP_{i2}}\min\left(1,\left[\left\lceil (t+J_p)/T_p\right\rceil-\left\lceil (E_{k,i1}+J_p)/T_p\right\rceil\right]C_p^{\mathrm{h}}\right)\right\}. \tag{10}$$

After obtaining the completion time of $\tau'_{k,i2}$, the completion time of the subsequent canonical form tasks can be calculated sequentially. When calculating the completion time of $\tau'_{k,ij}$, we should obtain MP_{ij} and SP_{ij}. MP_{ij} can be obtained by classifying all transactions according to the priority level of $\tau'_{k,ij}$. By considering the release jitters of transactions in task type classification of the algorithm presented by Gonzalez Harbour *et al.*(1994), SP_{ij} is denoted as follows:

$$SP_{ij}=SP'_{ij}\cup SP''_{ij}. \tag{11}$$

$$SP'_{ij}=\left\{\Gamma_p \mid \Gamma_p\in SP_{i(j-1)}\wedge\left(\left\lceil (E_{k,i(j-1)}+J_p)/T_p\right\rceil-\left\lceil (E_{k,i(j-2)}+J_p)/T_p\right\rceil\right)=0\wedge\left(\exists l,(P_{p1},\cdots,P_{pl}\geq P'_{ij})\wedge(P_{p(l+1)}<P'_{ij})\right)\right\},$$

$$SP''_{ij}=\left\{\Gamma_p \mid \Gamma_p\in(MP_{i(j-1)}-MP_{ij})\wedge\left(\exists l,(P_{p1},\cdots,P_{pl}\geq P'_{ij})\wedge(P_{p(l+1)}<P'_{ij})\right)\right\}.$$

After that, using equations similar to Eqs.(8) and (9), we can obtain the completion time of $\tau'_{k,ij}$. The response time of the kth job of Γ_i is

$$R_{k,i}=E_{k,im}+J_i-(k-1)T_i. \tag{12}$$

EXAMPLE AND EXPERIMENTS

Example

In this subsection, we use an engine electronic control system as an example to illustrate the use of the WCHS algorithm. The transactions are shown in Fig.2.

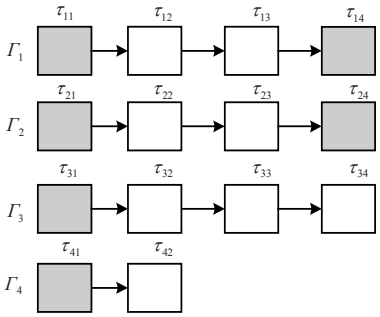

Fig.2 Engine electronic control system. Each transaction consists of multiple tasks, and the arrows between tasks denote the execution sequence of tasks. The tasks filled with gray color denote non-preemptive tasks

In Fig.2, Γ_1 is an electronic ignition transaction. It works as follows. First, τ_{11} collects shaft and load signals and regulates these signals. Then τ_{12} collects the rotation speed signals. After that, τ_{13} calculates the ignition time. Finally, τ_{14} sends the ignition command to the execution component. Γ_2 is an electronic fuel-injecting transaction. First, τ_{21} receives timer signals and initializes all the parameters about fuel injection. Then τ_{22} obtains the rotation speed signals. After that, τ_{23} calculates the optimal position for fuel injection. Finally, τ_{24} drives the injection component to inject fuels. Γ_3 is an electronic throttle control transaction. First, τ_{31} collects the signals of the pedal and the other related sensors. Then τ_{32} calculates the optimal position of the throttle. After that, τ_{33} calculates the adjustment angles according to the current position and the optimal position. Finally, τ_{34} sends commands to adjust the position of the throttle. Γ_4 is a transaction responsible for collecting the water temperature of the engine. τ_{41} first collects the water temperature information of the engine, and then sends it to τ_{42}. τ_{42} stores the water temperature information to a specific position. Γ_1, Γ_2, Γ_3 and Γ_4 are all periodic transactions with the periods of 20, 20, 500 and 2000 ms, and release jitters of 2, 3, 60 and 400 ms, respectively. All transactions' deadlines are equal to their periods. The parameters of all tasks are shown in Table 1.

Table 1 Task parameters of the transactions in Fig.2

Task	Priority	WCET (ms)	Preemption property
τ_{11}	9	0.5	non-preemptive
τ_{12}	6	2	preemptive
τ_{13}	11	3	preemptive
τ_{14}	10	1	non-preemptive
τ_{21}	11	0.5	non-preemptive
τ_{22}	8	2	preemptive
τ_{23}	6	3	preemptive
τ_{24}	7	2	non-preemptive
τ_{31}	5	1	non-preemptive
τ_{32}	3	40	preemptive
τ_{33}	4	15	preemptive
τ_{34}	5	20	preemptive
τ_{41}	2	2	non-preemptive
τ_{42}	1	40	preemptive

First, calculate the WCRT of Γ_1. Γ_1' consists of two tasks, τ_{11}' and τ_{12}'. τ_{11}' includes τ_{11} and τ_{12}; τ_{12}' includes τ_{13} and τ_{14}.

After classifying the types of transactions with $P_{11}'=6$, we can conclude that Γ_2 is a multiply preemptive transaction, and Γ_3 and Γ_4 are blocking transactions. τ_{31} and τ_{41} are non-preemptive transactions with priorities less than 6, so they have blocking effect on Γ_1'. From Eq.(1), we can derive that the blocked time Γ_1' suffers is contributed by τ_{41}, and is equal to 2 ms. The length of the Γ_1'-busy period is

$$L_1=\min\left(t>0\,|\,t=2+\lceil(t+3)/20\rceil\times 7.5+\lceil(t+2)/20\rceil\times 6.5\right)=16,$$

where the first item, 2, is the blocked time that Γ_1' suffers; the second term is the multiply preemptive time that Γ_2 contributes to Γ_1'; the third term is the total execution time of all Γ_1''s jobs in the busy period. Because Γ_1 has the release jitter of 2 ms and period of 20 ms, there is only one job of Γ_1 in the busy period.

Because the last task that $\tau_{1,11}'$ includes, τ_{12}, is preemptive, we can obtain the completion time of $\tau_{1,11}'$ using Eq.(4):

$$E_{1,11}=\min\left(t>0\,|\,t=2+\lceil(t+3)/20\rceil\times 7.5+2.5\right)=12,$$

where the first item, 2, is the blocking time that τ_{41} contributes to $\tau_{1,11}'$; the second term is the multiply

preemptive time that Γ_2 contributes to $\tau'_{1,11}$; the third item 2.5 is the WCET of $\tau'_{1,11}$.

After that, classify the types of all transactions in MP_{11} according to P'_{12}. We can know that SP_{12} has only the transaction Γ_2, and MP_{12} is empty. τ'_{12} is a non-preemptive task. From Eq.(10), we can derive its waiting time:

$$W_{1,12}=\min\Big(t>0\,|\,t=12+\min\big(1,\ \lceil\lceil(t+3)/20\rceil -\lceil 15/20\rceil\rceil\big)\times 0.5+3\Big)=15,$$

where 12 is the completion time of $\tau'_{1,11}$; 0.5 is the singly preemptive time coming from the initial H segment of Γ_2; 3 is the WCET of τ_{13}. Because the singly preemptive time is zero during the waiting time of $\tau'_{1,12}$, the waiting time of $\tau'_{1,12}$ is 12+3=15 (ms).

The completion time of $\tau'_{1,12}$ is

$$E_{1,12}=15+1=16.$$

There is only one job of Γ_1' in the busy period. The WCRT of Γ_1' is equal to that of Γ_1, i.e., 16+2=18 (ms).

Similarly, we can derive that the WCRTs of Γ_2, Γ_3 and Γ_4 are 19, 334 and 812 ms, respectively. Because the WCRT of each transaction is less than its deadline, the system is schedulable.

Experiments and discussion

In order to evaluate the performance of the WCHS algorithm, we compared the WCHS algorithm with the WCDOPS+ algorithm, an algorithm working well in transaction models. The experiment was performed on a PC with CPU AMD AthlonXP 2500+, 512 MB memory, and running Windows XP. We made use of the fact that transactions were executed sequentially to reduce the computational complexity of WCDOPS+. We investigated the WCRT of transactions and the computation time of the schedulability analysis with the average task numbers of transactions being 3, 5, 7, 9, 11 and 13. For a specific average task number, we created 20 transactions randomly. We used the method presented in (Pillai and Shin, 2001) to generate transaction sets. The periods of transactions can be short (1~10 ms), medium (10~100 ms), or long (100~1000 ms) periods to simulate different kinds of applications. The purpose of using transaction sets generated randomly is to avoid the limitation of transaction types and preemptive properties in real-world systems. Transactions are uniformly distributed into these three kinds of periods. Transactions are all preemptive with deadlines larger than or equal to their periods. The release jitters of transactions are generated randomly, and they are less than 20% of the transactions' deadlines. We compared their computational results and computing speed. Experiments showed that there are no differences in WCRT between WCHS and WCDOPS+ algorithms. The computation time of the schedulability analysis is shown in Fig.3.

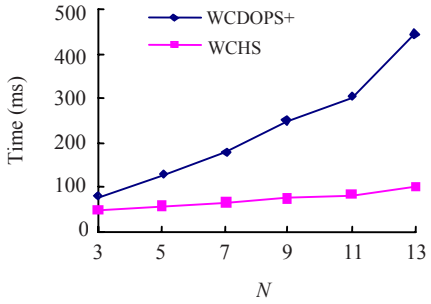

Fig.3 Computation time of the schedulability analysis vs. the average task number (*N*) of transactions

Fig.3 shows that the computation time of WCDOPS+ is always larger than that of WCHS. The larger the average task number of transactions, the more significant their time difference. WCDOPS+ calculates the best-case response time and the WCRT of tasks one by one to obtain the WCRT of transactions, which incurs more computation overheads. WCHS calculates the completion time of canonical form tasks to obtain the WCRT of transactions. The number of the canonical form tasks of a transaction is always less than that of the tasks of the transaction, so WCHS has less computation overheads. With the increment of average task numbers, difference between the computed task numbers in the two algorithms increases, which leads to more significant difference between their computation time.

Because there are many transactions triggered by the user input in automotive electronic systems, a steady average response time is desirable for the driver. Besides that, non-preemptive tasks may influence the schedulability of transactions, which re-

quires the developer to arrange the tasks' granularities (their WCET) and assign their priorities and preemption properties properly to develop correct software. Therefore, we investigated the influence of hybrid scheduling on the average WCRT and schedulability of transactions. Because there is no other existing algorithm for schedulability analysis of transactions under hybrid scheduling, we used the WCHS algorithm to perform the scheduling analysis. We used two transaction sets in the experiments. The first transaction set, TS1, was generated by using the above method and had 20 transactions. The average task number of transactions in TS1 was 7. After that, adjust the CPU utilization to be 10%, …, 80% by scaling the tasks' WCET proportionally. Under every utilization, generate the transaction set TS2 by changing the tasks' preemption property with the probability of 10% to be non-preemptive. The average WCRT is shown in Fig.4, and the average unschedulable transaction number is shown in Fig.5. Note that every datum is the average value of 10 measurements.

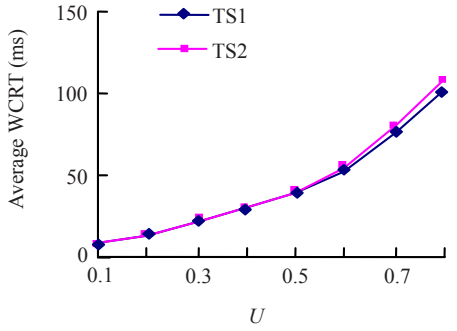

Fig.4 Average WCRT vs. utilization *U*

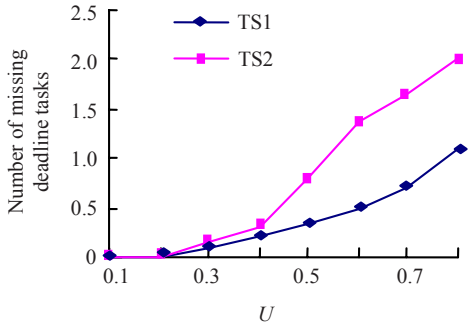

Fig.5 Number of missing deadline tasks vs. utilization *U*

Fig.4 shows that the average WCRT of transactions increases with the increment of utilization, and the average WCRT of TS2 is usually larger than that of TS1. Under higher utilization, more interference among transactions makes the average WCRT of transactions longer. Although the non-preemptive task may reduce the WCRT of the transaction it belongs to, it may increase the WCRT of transactions with higher task priorities, which increases the average response time of systems. However, in some cases, the average WCRT of TS2 is smaller than that of TS1. It is because some transactions with lower task priorities can improve schedulability greatly after changing tasks' properties into non-preemption. In a word, there is little difference of the average WCRT of transactions between TS1 and TS2.

Fig.5 shows that some transactions become unschedulable after some tasks become non-preemptive, and the non-preemptive task number increases with the increment of utilization. Usually, short and medium period transactions consist of tasks with high priorities. In some cases, non-preemptive tasks may contribute blocking effect to short and medium period transactions, making them unschedulable. With the increment of utilization, more interference among transactions leads to more unschedulable transactions.

It is well known that non-preemptive tasks have the advantage of preventing tasks' execution from being interrupted, reducing system overheads and making resource synchronization easy to realize. From the above experiments, we can draw the conclusion that non-preemptive tasks have little influence on the average WCRT of transactions. Sometimes we can make use of non-preemptive tasks to reduce the WCRT of some transactions. However, if the WCET of a non-preemptive task is too long, it is necessary to take some measures, for example, dividing a non-preemptive task into multiple tasks with smaller WCET, or assigning tasks' priorities and preemption properties properly to reduce the blocking effect on other transactions.

CONCLUSION

In this paper, we present the worst-case hybrid scheduling (WCHS) algorithm, a schedulability analysis method for transactions with release jitters

under a hybrid scheduling model. Experiments show that this algorithm has lower computation overheads while keeping correctness. Moreover, we analyze the influence of hybrid scheduling on the average worst-case response time (WCRT) and schedulability through experiments. We propose the method on how to limit the negative influence of hybrid scheduling. Our future work will focus on how to extend the WCHS algorithm to more hard transaction models, such as the tree-shaped transaction model.

References

ARTIST2, 2005. ARTIST2 Roadmap on Real-Time Techniques in Control. Http://www.artist-embedded.org/artist/ARTIST-2-Roadmap-on-Real-Time.html

Audsley, N., Burns, A., Richardson, M., Tindell, K., Wellings, A.J., 1993. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Eng. J.*, **8**(5): 284-292.

Baruah, S.K., Chakraborty, S., 2006. Schedulability Analysis of Non-Preemptive Recurring Real-Time Tasks. Proc. 20th Int. Parallel and Distributed Processing Symp., p.1-8. [doi:10.1109/IPDPS.2006.1639406]

Buttazzo, G.C., 1995. Hard Real-Time Computing Systems. Kluwer, Boston, MA.

Damm, A., Reisinger, W., Schwabl, W., Kopetz, H., 1989. The real-time operating system of MARS. *ACM SIGOPS Oper. Syst. Rev.*, **23**(3):141-151. [doi:10.1145/71021.71029]

Dolev, S., Keizelman, A., 1999. Non-preemptive real-time scheduling of multimedia tasks. *Real-Time Syst.*, **17**(1): 23-39. [doi:10.1023/A:1008033411290]

Fisher, N., Nguyen, T.H.C., Goossens, J., Richard, P., 2007. Parametric Polynomial-Time Algorithms for Computing Response-Time Bounds for Static-Priority Tasks with Release Jitters. Proc. 13th IEEE Int. Conf. on Embedded and Real-Time Computing Systems and Applications, p.377-385. [doi:10.1109/RTCSA.2007.54]

Gonzalez Harbour, M., Klein, M.H., Lehoczky, J.P., 1994. Timing analysis for fixed-priority scheduling of hard real-time systems. *IEEE Trans. on Software Eng.*, **20**(1):13-28. [doi:10.1109/32.263752]

Henia, R., Ernst, R., 2005. Context-Aware Scheduling Analysis of Distributed Systems with Tree-Shaped Task-Dependencies. Proc. Design, Automation and Test in Europe, **1**:480-485. [doi:10.1109/DATE.2005.104]

Jeffay, K., Stanat, D.F., Martel, C.U., 1991. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. Proc. IEEE Real-Time Systems Symp., p.129-139. [doi:10.1109/REAL.1991.160366]

Jiang, S., 2006. A Decoupled Scheduling Approach for Distributed Real-Time Embedded Automotive Systems. Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symp. p.191-198. [doi:10.1109/RTAS.2006.5]

Khil, A., Maeng, S., Cho, J., 1997. Non-preemptive scheduling of real-time periodic tasks with specied release times. *IEICE Trans. on Inf. Syst.*, **E80-D**(5):562-572.

Lehoczky, J.P., 1990. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadline. Proc. 11th Real-Time Systems Symp., p.201-209. [doi:10.1109/REAL.1990.128748]

Liu, C.L., Layland, J., 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, **20**(1):46-61. [doi:10.1145/321738.321743]

OSEK, 2003. OSEK/VDX Operating System, Version 2.2.1. Http://www.osek-vdx.org/mirror/os221.pdf

Palencia, J.C., Gonzalez Harbour, M., 1998. Schedulability Analysis for Tasks with Static and Dynamic Offsets. Proc. 19th IEEE Real-Time Systems Symp., p.26-37. [doi:10.1109/REAL.1998.739728]

Palencia, J.C., Gonzalez Harbour, M., 1999. Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems. Proc. 20th IEEE Real-Time Systems Symp., p.328-339. [doi:10.1109/REAL.1999.818860]

Pillai, P., Shin, K.G., 2001. Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. Proc. 18th ACM Symp. on Operating Systems Principles, p.89-102. [doi:10.1145/502034.502044]

Redell, O., 2004. Analysis of Tree-Shaped Transactions in Distributed Real Time Systems. Proc. 16th Euromicro Conf. on Real-Time Systems, p.239-248. [doi:10.1109/EMRTS.2004.1311026]

Tindell, K.W., 1994. Adding Time-Offsets to Schedulability Analysis. Technical Report YCS 221, Univ. of York.

Wang, L., Wu, Z.H., 2004. Schedulability Test for Fault-Tolerant Hybrid Real-Time Systems with Preemptive and Non-Preemptive Tasks. Proc. 4th Int. Conf. on Computer and Information Technology, p.1169-1174. [doi:10.1109/CIT.2004.10001]

Yomsi, P.M., Sorel, Y., 2007. Extending Rate Monotonic Analysis with Exact Cost of Preemptions for Hard Real-Time Systems. Proc. 19th Euromicro Conf. on Real-Time Systems, p.280-290. [doi:10.1109/CRTS.2007.15]