Response Time Analysis of Execution Right Delegation Scheduling

Takaharu Suzuki[1,a)] Kiyofumi Tanaka[1,b)]

Abstract: In scheduling algorithms based on Rate Monotonic (**RM**) method widely used in development of real-time systems, tasks with shorter periods have higher priorities. In contrast, ones with longer periods are likely to suffer from increased response times and jitters due to their lower priorities. Execution Right Delegation (**ERD**) we proposed is a method based on RM where a high-priority server for particular (or important) tasks is introduced to shorten response time and jitter of the tasks. Our previous work showed ERD improves real-time processing of the target tasks through simulations. However, Response Time Analysis (**RTA**), which assures worst case response time of the task, was left to future work. This paper shows RTA for ERD and evaluates it by comparing with a Deadline Monotonic method and ERD simulation results.

Keywords: Real-time scheduling, Response Time Analysis, Rate Monotonic

1. Introduction

The main purpose of real-time scheduling algorithms is to achieve optimal scheduling for tasks which have a period, execution time, and/or deadline. It is important for the algorithms to not only meet the deadline of each task but also shorten its response time and jitter.

In scheduling algorithms based on Rate Monotonic (**RM**) [1] method widely used in development of real-time systems, tasks with shorter periods have higher priorities. In contrast, ones with longer periods are likely to suffer from increased response times and jitters due to their lower priorities.

Execution Right Delegation (**ERD**) [2] [3] we proposed is a method based on RM where a high-priority server for a particular (or important) task[i] is introduced to shorten response time and jitter. Our previous work showed ERD improves real-time processing of the target tasks through simulations. ERD method assures that worst case response time (**WCRT**) of the important task is at least equal to or less than that by the conventional RM method. However, obtaining Response Time Analysis (**RTA**), which gives the WCRT, was left as a future work in the previous study. The aim of this paper is to show RTA of ERD.

This paper consists of five sections. Section 2 describes related work in terms of response time analysis in real-time systems. Section 3 reviews ERD method which was proposed in the authors' previous study. Section 4 proposes RTA of ERD. Evaluation of the proposed method is shown in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

2.1 Scheduling Algorithms

In the RM model, task τ_i releases an infinite sequence of Jobs. Once Job is released, it runs during the defined time C_i. Job is released every period T_i.

Notation of a task is $\tau_i = (C_i, T_i)$. A set of tasks is denoted as $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where the smaller subscript figure a task has, the shorter period and higher priority it has (i.e. $T_1 \leq T_2 \leq ... \leq T_n$). A deadline miss occurs if Job does not finish by the next task release.

Deadline Monotonic (**DM**) [4], whose model includes relative deadline D_i in addition to period T_i and execution time C_i, is another scheduling algorithm. The shorter the relative deadline of a task is, the higher its priority is. As priority is determined by deadline independent of a period, response time and jitter of a task with a longer period can be shortened by assigning a short deadline.

There are several fixed-priority server algorithms for shortening response times of particular aperiodic tasks. Deferrable Server (**DS**) [5] and Priority Exchange (**PE**) [5] are representatives. Servers in these algorithms have a period and a capacity which corresponds to execution time. The servers are scheduled along with a set of periodic tasks while their capacity is consumed for execution of aperiodic tasks. In these algorithms, the server period and capacity are decided according to processor utilization of the whole periodic task set, where the importance of a particular periodic task is not considered. On the other hand, the method proposed in our previous paper obtains and uses the period and the

[1] Japan Advanced Institute of Science and Technology, Tokyo/Ishikawa, Japan
[a)] hal-suzuki@jaist.ac.jp
[b)] kiyofumi@jaist.ac.jp

[i] We assume the particular task has relatively lower priority due to its longer period. For example, for CAN messages in integrated ECUs, control messages have a shorter period, but notification messages have a longer period. However, some notification messages are urgent and must receive a higher priority (e.g. warning lamp, low fuel).

capacity in favor of a particular (important) periodic task, although the server algorithm is basically PE.

2.2 Priority Assignment

The aim of ERD is to shorten response time of an important task regardless of its period. In order to assign priority to the task regardless of the period, Leung and Whitehead generalized that DM priority assignment is optimal for synchronous periodic task sets (without phases) with constrained deadlines [6]. In different system model (e.g. tasks with phases, tasks with arbitrary deadlines, non preemptive scheduling), it is known that DM is not optimal [7].

2.3 Response Time Analysis

There is a classical method in order to judge whether a task set is schedulable or not. Liu et al. [1] proposed a schedulability test by calculating processor utilization by all tasks and comparing it with the least upper bound for the task set. This test provides only a sufficient condition, which leaves the estimate pessimistic.

Response Time Analysis (**RTA**) [8] provides a necessary and sufficient condition of schedulability for RM, which is defined as follows:

Definition 2.1 (Response Time Analysis (RTA) [8] of RM): *In a fixed-priority scheduling, the longest response time*[ii]*, R_i, of task τ_i is computed as*[iii]*:*

$$R_i = C_i + \sum_{j=1}^{i-1} \lceil \frac{R_i}{T_j} \rceil Cj. \tag{1}$$

R_i appears on both sides. Thus, R_i is calculated by setting an appropriate value (e.g. C_i) as the initial value to R_i and increasing R_i until both sides become equal.

Several RTA methods for multiprocessor are proposed. Bertogna et al. proposed RTA for global fixed-priority multiprocessor scheduling [9]. Compared with single processor RTA, multiprocessor RTA provides only a sufficient condition with its pessimism. In the method, interference time, which is a total amount of time that higher priority tasks prevent a certain task from working, is introduced. Guan et al. improved Bertogna's method by limiting carry-in load, a part of interference time, of higher priority tasks (**RTA_LC**) [10]. Sun et al. revised RTA_LC to reduce its pessimism. The method provides more accuracy result [11]. They applied schedulability test for multiprocessor partitioned schedule by the RTA [12]. Zhou et al. showed RTA for a model that a task has a non-preemptive point under multiprocessor global scheduling [13]. As mentioned above, there are various RTAs dedicated to specific models or different systems.

ii If the first jobs of all tasks are released simultaneously at the instant $t = 0$, response time of each task becomes the worst case for the corresponding task (Critical Instant [1]).

iii The smaller subscript a task has, the shorter period and higher priority it has. That is, for τ_i and τ_j, if $i < j$, τ_i has higher priority than τ_j.

3. Execution Right Delegation (ERD)

3.1 System Model and Definitions

ERD is a method to shorten response time and jitter of a particular (or important) task, τ_p, in a task set by using a high-priority virtual server, VS, which has capacity of C_s and period of T_s while keeping whole deadline of the task set. We assume τ_p has a relatively lower priority due to its longer period. By making the priority of VS high with short T_s, τ_p can be executed at the high priority while consuming C_s. The basic behavior of VS is from PE [5].

The target system model of this study is single processor fixed task priority. ERD assumes that each task has execution time and period, and that the deadline is equal to its period. A task does not have a phase, which means that the first job is released at $t = 0$. The scheduling rule follows RM except for a particular (target) task which VS is applied to.

First, we describe a theorem and a lemma as well as definitions underlying ERD[iv]. Based on these, a virtual server VS is introduced into a task set.

Theorem 3.1 (Changing Priorities of Tasks): *Assuming that a schedulable task set Γ includes τ_p and τ_h and that the priority of τ_h is higher than that of τ_p, a task set Γ', in which the priority of τ_p is changed to be higher than τ_h while the others are in the same priority as in Γ, is schedulable if R_p is equal to or less than T_h.*

Definition 3.2 (Idle Time): *An Idle Time is a period in which any task's job is not executed. idle(t) gives the total amount of idle times between 0 and the instant t.*

Lemma 3.3 (Adding an Idle Task): *If there exists Idle Time between 0 and τ_h's period T_h, adding a new task τ_{idle}, whose execution time is idle(T_h)*[v] *and whose period is T_h, to Γ leaves the new task set schedulable.*

Definition 3.4 (Delegation of Execution Right): *In ERD method, VS is scheduled based on RM rule. When VS is given the execution right, a particular periodic task, τ_p, is executed instead of VS. This situation is called Delegation of Execution Right. If Job of τ_p has already finished when the execution right is available, the behavior follows PE rule [5] where jobs of the other tasks are executed while the server capacity is accumulated at the priority level of the running job.*

Next, the following definition gives the algorithm for finding C_s and T_s.

Definition 3.5 (Candidates of VS): *Let Γ be a schedulable task set and τ_p in Γ be a particular task whose response time and jitter should be shortened. Then, RTA in Definition 2.1 is applied to $\tau_1, ...,$ and τ_p. From the relation between R_p and $T_1, ..., T_{p-1}$, C_s and T_s for VS are obtained as follows.*

iv Proofs of the theorem and lemma are found in [2].

v In this case, a relative time duration, e.g., T_h, is regarded as a time instant ($t = T_h$).

$$C_s = \begin{cases} C_p, & if\ R_p \le T_{p-1} \quad (2) \\ idle'(T_s), & otherwise \quad (3) \end{cases}$$

$$T_s = \begin{cases} T_h, & if\ R_p \le T_{p-1} \quad (4) \\ t \in \Psi, & otherwise \quad (5) \end{cases}$$

where

$$\Psi = \{T_1, T_2, ..., T_{p-1}\}$$
$$T_h = min(\ \{t \mid t \in \Psi,\ R_p \le t\}\ )$$
$$idle'(t) = t - \sum_{j=1}^{p-1} \lceil \frac{t}{T_j} \rceil Cj \quad (t \in \Psi)$$

The following gives a scheduling example with VS derived from the equations (2) and (4) .

Example 3.1 (ERD method - 1)**:** With $\Gamma = \{(2,4),(3,12),(3,14)\}$ and $\tau_p = \tau_3$, RTA gives response times of the three tasks as $R_1 = 2, R_2 = 7$, and $R_3 = 12$. Since $R_3 \le T_2(= 12)$, equations (2) and (4) offer VS $= (3,12)$.

Scheduling results by RM and ERD are shown in Figure 1 and Figure 2, respectively. At the instants $t = 2, 3$, and 6, delegation of execution right can be confirmed. In this example, ERD improves response time of τ_3 from 12 to 7.

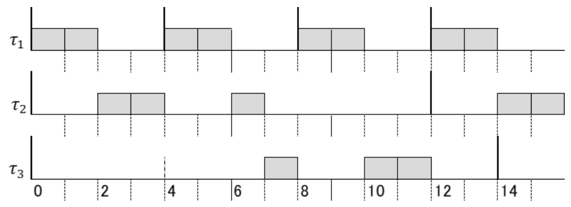

Fig. 1 Scheduling example of RM.

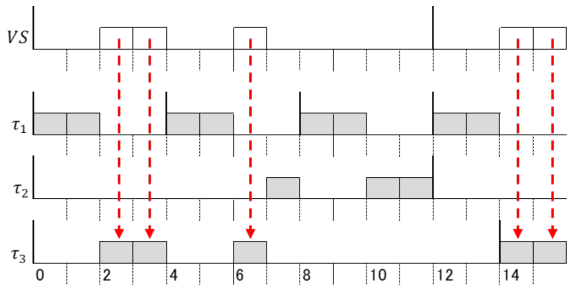

Fig. 2 Scheduling example by ERD.

As for VS provided by the equations (3) and (5) , more than one candidate can exist. $\Psi = \{T_1, T_2, ...T_{p-1}\}$ is a set of periods which are shorter than T_p, and $idle'(T_i)$ gives the length during which τ_p is executed. (Note that $idle()$ in Definition 3.2 and $idle'()$ in Definition 3.5 are different. While the former gives the sum of all idle times before any instant t, the latter means, in each period in Ψ, the sum of time slots during which any task in $\tau_1, ..., \tau_{p-1}$ is not executed but τ_p is executed due to $R_p > T_{p-1}$.)

Example 3.2 (ERD method - 2)**:** With $\Gamma = \{(1,5),(1,6),(2,8),(4,14)\}$ and $\tau_p = \tau_4$, RTA gives

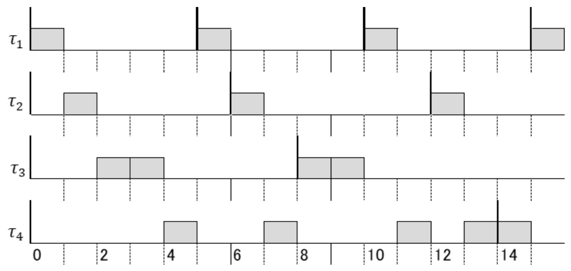

Fig. 3 Schedule by RM for $\Gamma = \{(1,5),(1,6),(2,8),(4,14)\}$.

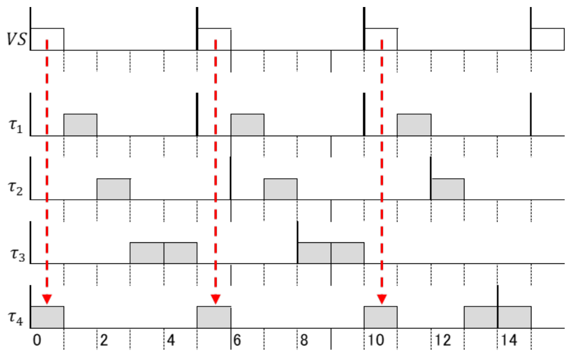

Fig. 4 Schedule by ERD with VS_1 for Γ.

$R_1 = 1, R_2 = 2, R_3 = 4$, and $R_4 = 14$ (Figure 3).

VS is given by the equations (3) and (5) due to $R_4 > T_3$. With $\Psi = \{T_1, T_2, T_3\}$, $idle'(T_i)$ is calculated for each period. Since $idle'(T_1) = 1$, $VS_1 = (1,5)$ is derived. With VS_1, $R'_4 = 14$, which does not shorten response time (Figure 4). Similarly, $VS_2 = (1,6)$ is obtained from $idle'(T_2) = 1$. However, VS_2 still gives $R'_4 = 14$.

Then, from $idle'(T_3) = 2$, $VS_3 = (2,8)$ is derived and found to give $R'_4 = 10$ (Figure 5). Finally, the results of VS_1, VS_2, and VS_3 are compared and VS_3 is appointed as VS because of the shortest response time for τ_4.

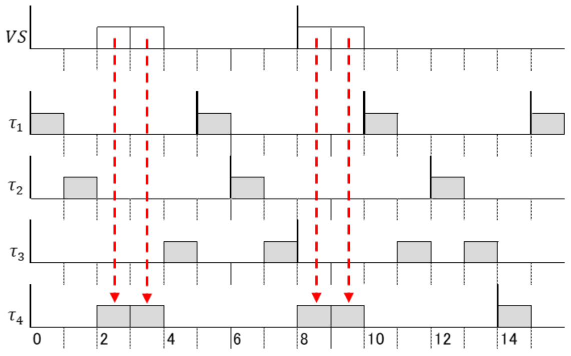

Fig. 5 Schedule by ERD with VS_3 for Γ.

It is worth noting that Deadline Monotonic scheduling[vi] does not shorten the response time of τ_4 with the above Γ. DM is effective only when the relative deadline of τ_4 can be made less than or equal to the period of higher-priority tasks. However, for Γ in this example, if the deadline is set to be less than or equal to T_3, τ_3's job results in missing its deadline.

[vi] In DM model, each task has relative deadline independent of its period and execution time. The shorter the deadline is, the higher its priority is.

4. RTA of ERD

4.1 Interference Time

Our previous paper [2] showed through simulation that ERD can reduce the response time of specific task τ_p but it left a future work of obtaining the WCRT from mathematical perspective. In the model of single processor RM, WCRT of a task is equal to the response time of its first released job. On the other hand, as in multiprocessor scheduling, the instance of τ_p's WCRT is unknown in ERD. In some cases, τ_p's first job's response time becomes its WCRT like examples in the previous section. However, Figure 6 shows the second job's response time is longer than the first job's one.

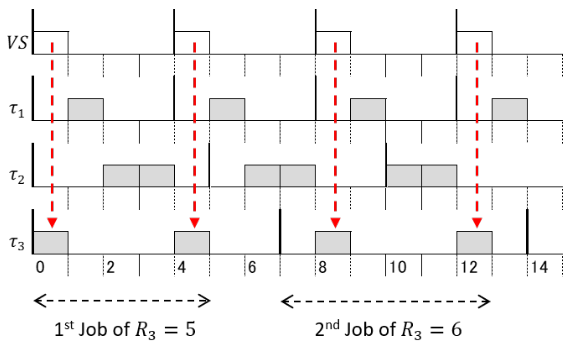

Fig. 6 Second Job has longer response time.

In this section, we present RTA of ERD. Let R_i^{RM} be a WCRT of τ_i in RM scheduling, and R_i^{ERD} be a WCRT of τ_i in ERD scheduling.

Definition 4.1 (τ_{vs} and τ_h): *τ_{vs} is a task whose period is the same as VS and whose priority is treated as being lower then VS. τ_h is a task whose priority is higher than τ_p but excludes τ_{vs}.*

τ_2 in Figure 2, τ_1 in Figure 4, and τ_3 in Figure 5 are τ_{vs}.

Lemma 4.2 (Upper bound of WCRT in ERD): *For τ_p, $R_p^{ERD} \leq R_p^{RM}$.*

Proof. In ERD method, τ_p is executed if VS whose priority is higher than τ_p has execution right. As a result, τ_p's execution is advanced. Otherwise, τ_p is scheduled according to the same rule as the RM method. Therefore, the response time can be no longer than the RM method. □

Theorem 4.3 (WCRT of High Priority Task): *A task subset Γ_{hp} consists of tasks whose priority is higher than τ_p. For any $\tau_i \in \Gamma_{hp}(i < p)$, a response time of τ_i's first job exhibits R_i^{ERD}.*

Proof. For τ_i, VS can be regarded as a task with execution time C_s and period T_s. Let VS be appended as a periodic task to Γ_{hp}, and make a task set $\Gamma_{hp'}$. After scheduling $\Gamma_{hp'}$ with RM rule, WCRT of τ_i can be retrieved from Definition 2.1. □

Definition 4.4 (Interference Time): *Interference time I_i^{ERD} is the maximum time that some higher-priority task τ_i affects response time of τ_p under ERD scheduling. I_i^{RM} is the case under RM scheduling and calculated as $I_i^{RM} = \lceil \frac{R_p^{RM}}{T_i} \rceil C_i$.*

Lemma 4.5 (Upper bound of Interference Time of τ_h): *For $\tau_h \in \Gamma_{hp}(h < p, h \neq vs)$, $I_h^{ERD} \leq I_h^{RM}$.*

Proof. Due to a contribution of VS, $I_h^{ERD} \leq \lceil \frac{R_p^{ERD}}{T_h} \rceil C_h$. From Lemma 4.2, $\lceil \frac{R_p^{ERD}}{T_h} \rceil C_h \leq \lceil \frac{R_p^{RM}}{T_h} \rceil C_h = I_h^{RM}$. Thus, $I_h^{ERD} \leq I_h^{RM}$. □

Now that we obtained upper bound of interference time of τ_h, with regards to the estimate of interference time of τ_{vs}, we start a method based on Bertogna's way [9]. We focus on the interference time while a task τ_{vs} with higher priority than τ_p operates in an interval L. L is divided into 3 sub intervals (Figure 7): L_{body} is the interval consisting of τ_{vs}'s successive periods which are totally included in L, L_{carry_in}/L_{cin} is the interval between the beginning of L and the beginning of L_{body}, and L_{carry_out}/L_{co} is the interval between the end of L_{body} and the end of L. Similar to the single processor RTA in Definition 2.1, ERD RTA repeats the calculations so that the right and left sides gets equal. Let the initial value of L be the response time of τ_p in the RM method.

$$L = R_p^{RM} \tag{6}$$

Now we consider L_{cin}, L_{body}, and L_{co} that maximize an interference time of τ_{vs}. There are two cases in which the interference time is maximized: L starts at the time when the job of τ_{vs} starts executing, and the job finishes at the end of L. The same discussion can be made for the both cases. We choose the former case here. In the interval L_{cin},

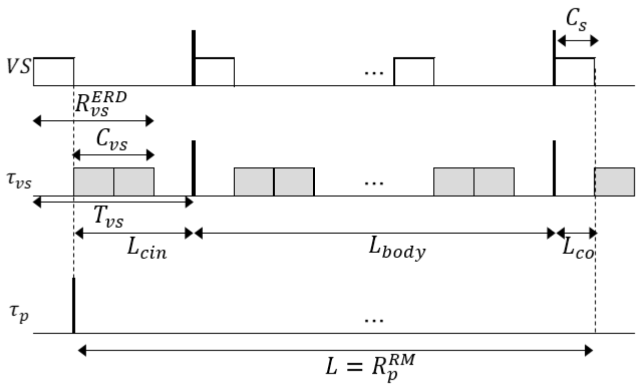

Fig. 7 Body, Carried-in and Carried-out jobs.

in order to maximize interference time of τ_{vs}, the whole job is required to be executed. On the other hand, in order to maximize the length of L_{body} plus L_{co}, L_{cin} must be minimized. Such L_{cin} is calculated as:

$$L_{cin} = T_{vs} - R_{vs}^{ERD} + C_{vs}. \tag{7}$$

where R_{vs}^{ERD} is derived from Theorem 4.3. With L_{cin}, L_{body} and L_{co} are:

$$\begin{aligned} nbody &= \lfloor \frac{L - L_{cin}}{T_{vs}} \rfloor \\ L_{body} &= nbody \times T_{vs} \\ L_{co} &= L - (L_{cin} + L_{body}). \end{aligned}$$

Thus, the execution times, *cin* and *body*, of τ_{vs}'s jobs in the intervals L_{cin} and L_{body}, respectively, are calculated as follows.

$$\begin{aligned} cin &= C_{vs} \\ body &= nbody \times C_{vs} \end{aligned}$$

Lemma 4.6 (Carried-out job of τ_{vs}): *Upper bound of the execution time, co, of τ_{vs}'s job in the interval L_{co} is calculated as follows.*

$$co = min(max(L_{co} - C_s, 0), C_{vs})$$

Proof. From Definition 4.1, τ_{vs}'s priority is evidently lower than VS and its period is the same as VS. That is, at the release instant of τ_{vs}'s job, VS is ready to execute. If there is no higher priority task's job than VS, τ_p's job is executed. In the case that higher priority task's job is released, that job is executed. In any case, τ_{vs}'s job must wait for C_s, which is the capacity of VS. □

Lemma 4.7 (Interference Time of ERD): *Interference Time I_i^{ERD} of τ_i during an interval L is calculated as follows.*

$$I_i^{ERD} = \begin{cases} cin + body + co & \text{if } \tau_i = \tau_{vs} \quad (8) \\ I_i^{RM} & otherwise \quad (9) \end{cases}$$

Proof. By Lemmas 4.5, 4.6 and the discussion in this section. □

4.2 Reducing Carry-in

The discussion in the previous subsection supposes that, in an interval L, the interference time to τ_p is the maximum one. Thus, the response time of τ_p is to be pessimistic. In regard to τ_{vs}'s carry-in job, we consider whether it is possible to be shortened or not. First, focus on the interval L' which is τ_p's period immediately before L (Figure 8). Each

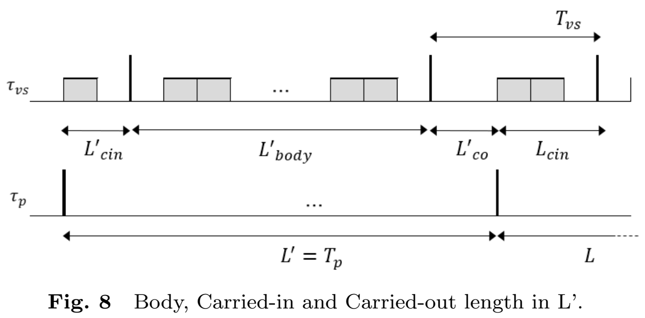

Fig. 8 Body, Carried-in and Carried-out length in L'.

sub interval of L' is calculated as follows.

$$\begin{aligned} L' &= T_p \\ L'_{co} &= T_{vs} - L_{cin} \\ nbody' &= \lfloor \frac{L' - L'_{co}}{T_{vs}} \rfloor \\ L'_{body} &= nbody' \times T_{vs} \\ L'_{cin} &= L' - (L'_{co} + L'_{body}) \end{aligned}$$

Thus, $co', body'$, and cin' are as follows.

$$\begin{aligned} co' &= 0 \\ body' &= nbody' \times C_{vs} \\ cin' &= min(max(L'_{cin} - T_{vs} + R_{vs}^{ERD}, 0), C_{vs}) \end{aligned}$$

Let $empty(l)$ be an idle/empty time in the interval l. Intuitively, l minus the sum of C_p and the interference time of all of higher priority tasks than τ_p is idle/empty time. If there is idle/empty time, it means there is room to advance the carry-in job of τ_{vs} in L_{cin}. Then, we calculate empty time in the interval L' as follows.

$$\begin{aligned} empty(L') = max(L' - (cin' + body' + co') - \\ C_p - \sum_{j=1, j \neq vs}^{p-1} I_j^{ERD}, 0) \end{aligned}$$

$empty(L')$ is the amount of time by which the carry-in job is advanced. That is, a part of the carry-in job in L_{cin} is moved to L'_{co}. Then, we update the execution time of the carry-in job as $ncin$.

$$ncin = cin - empty(L')$$

Compared to the formula (7) , L_{cin} of interval L can be reduced as follows.

$$L_{cin} = T_{vs} - R_{vs}^{ERD} + ncin$$

We can replace *cin* and L_{cin} in the formulas in the previous subsection by *ncin* and the reduced L_{cin}, respectively, and therefore can alleviate the pessimism of the interference time.

4.3 Updating *L*

With the updated I_i^{ERD}, the initial value of R_p^{ERD} is calculated as follows.

$$R_p^{ERD} = \sum_{i<p} I_i^{ERD} + C_p$$

Recalling (6) , L can be updated by R_p^{ERD}. In addition, the formula (9) in Lemma 4.7 can be updated since the response time of τ_p is now shortened compared with RM (see Definition 4.4). Consequently, the formula of Lemma 4.7 is updated as follows.

$$I_i^{ERD} = \begin{cases} cin + body + co & \text{if } \tau_i = \tau_{vs} \quad (10) \\ \lceil \frac{R_p^{ERD}}{T_i} \rceil C_i & \text{otherwise} \quad (11) \end{cases}$$

Theorem 4.8 (RTA of ERD): *In ERD, τ_p's WCRT* *is calculated as :*

$$R_p^{ERD} = \sum_{i<p} I_i^{ERD} + C_p. \quad (12)$$

where the initial value of R_p^{ERD} is R_p^{RM}. Calculation is repeated until R_p^{ERD} is unchanged.

Proof. By the discussion in this section. □

5. Evaluation

5.1 Task Sets

In this section, we evaluate the proposed method by showing how much the analyzed WCRT is shortened compared with RM RTA. For comparison, we show the results of ERD RTA and DM RTA. In addition, the longest response times in the task scheduling simulation with ERD (ERD simulation) are shown for reference[vii]. For evaluation, 10,000 task sets are synthetically generated. Each task set is schedulable (with no deadline misses) under RM and has processor utilization of 80% to 96%. When generating task sets, tasks' periods are decided by random numbers based on the exponential distribution, and their execution times are decide by random numbers based on the uniform distribution where the upper bound is 50% of the corresponding periods.

In each task set, the target task τ_p whose response time should be shortened is a task with the lowest priority or a task with a medium priority. Simulation period is 100,000 ticks. The same task sets are used for ERD RTA and DM RTA as well as ERD simulation.

5.2 Results

The ERD method shortens the response time by using a high-priority virtual server, while the DM method can shorten the response time when the priority of τ_p can be raised by regulating its relative deadline. Figure 9 shows the ratio of task sets which can reduce WCRT when the priority of τ_p is the lowest in the task sets. The horizontal axis indicates the processor utilization of the task sets, and the vertical axis indicates the ratio of the task sets that can have WCRT reduced. In the ERD simulation, the longest response times can be reduced for almost all task sets whose processor utilization is up to 84%. On the other hand, in the ERD RTA method, fewer task sets have shorter WCRT for τ_p. This means there is a possibility that ERD RTA is still pessimistic. In the DM method, it is confirmed that even fewer task sets that in ERD RTA can have shortened WCRT for τ_p.

Figure 10 shows the average WCRT normalized to RM RTA. While ERD RTA exhibits shorter WCRT than DM RTA, it is longer than ERD simulation due to its pessimism, which implies there is room for further improving the analysis.

Figure 11 and Figure 12 are the results when the priority of τ_p is a medium among the tasks in a set. Similar trends

[vii] In ERD simulation, the obtained longest response times might not be WCRTs, since the simulation period is limited.

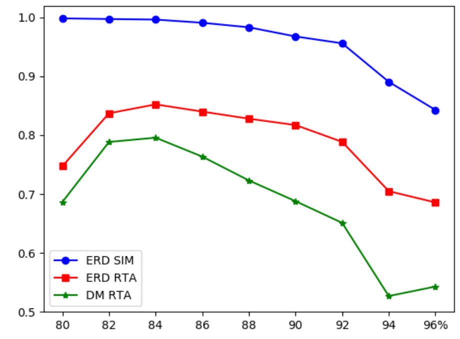

Fig. 9 Ratio of task sets with shortened WCRT (τ_p's priority is the lowest).

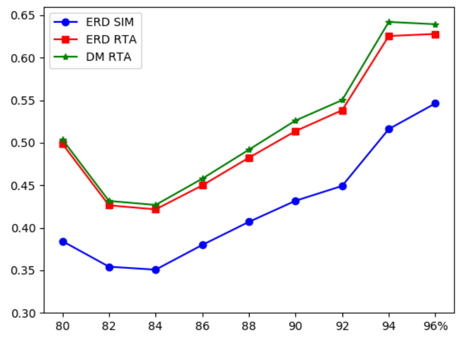

Fig. 10 Average WCRT normalized to RM (τ_p's priority is the lowest).

to Figure 9 and Figure 10 can be seen except that more task sets have shorter WCRTs than RM RTA (Figure 11).

5.3 Discussion

From the results in the previous subsection, we can confirm that the proposed RTA method for ERD obtains shorter WCRTs than DM method for the same task set, which means ERD can give higher schedulability than DM. However, the obtained WCRT is still pessimistic compared to the actual longest response times exhibited by ERD simulation. This pessimism is mainly due to the calculation where the interference time of a higher priority task τ_h (not τ_{vs}) is supposed to be the same as in the case of RM method (see the formula (11)). Further improvement in WCRT would be made by reducing the interference time of τ_h which has higher priority than τ_p but lower than VS. This problem is a combinatorial optimization problem under the model that tasks have phases and is left to a future work.

6. Conclusion

Execution Right Delegation (ERD) is a real-time scheduling technique that improves real-time performance of tasks which have long periods but are of high importance, while RM necessarily gives low priority to tasks with long periods. In this paper, after reviewing ERD, we showed a response time analysis (RTA) method of ERD and evaluated it in comparison with DM RTA and the simulated behavior of ERD. The results showed that ERD RTA can give shorter

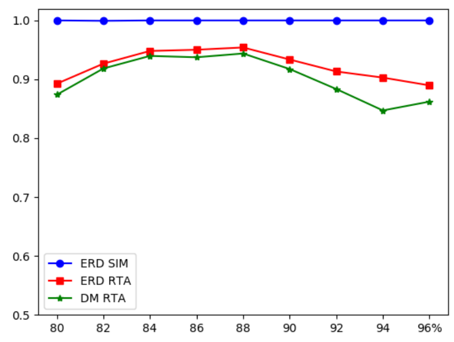

Fig. 11 Ratio of task sets with shortened WCRT (τ_p's priority is a medium).

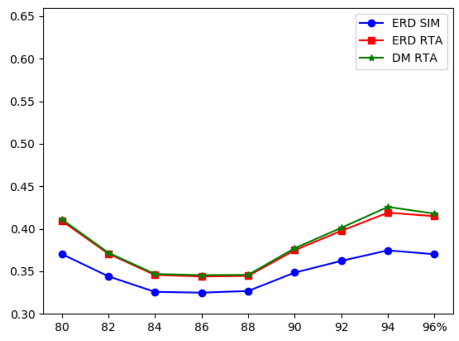

Fig. 12 Average WCRT normalized to RM (τ_p's priority is a medium).

worst case response times (WCRTs) than DM RTA but is still pessimistic analysis compared with the simulation results. There is room to improve the analysis for further reduced WCRT.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number JP 19K11873.

References

[1] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment," Journal of the ACM, Vo.20, No.1, pp.40–61, 1973.
[2] T. Suzuki and K. Tanaka, "A Virtual Server for Shortening Task Response Time," Proc. of IPSJ Embedded Systems Symposium, pp.37–46, 2016. (in Japanese)
[3] T. Suzuki and K. Tanaka, "Execution Right Delegation: Beyond the Rate Monotonic," Proc. of the 32nd International Conference on Computers and Their Applications, 2017.
[4] N. C. Audsley, A. Burns, M. F. Richardson, and A J. Wellings, "Hard Real-Time Scheduling: The Deadline Monotonic Approach," Proc. of IEEE Workshop on Real-Time Operating Systems and Software, pp.133–137, 1991.
[5] J. P. Lehoczky, L. Sha, and J. K. Strosnider, "Enhanced Aperiodic Responsiveness in Hard Real-Time Environments," Proc. of IEEE Real-Time Systems Symposium, pp.261–270, 1987.
[6] Leung, Joseph Y-T., and Jennifer Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks," Performance evaluation, Vol.2, No.4, pp.237–250, 1982.
[7] Davis, Robert I., et al., "A review of priority assignment in real-time systems," Journal of Systems Architecture, Vol.65, pp.64–82, 2016.
[8] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," The Computer Journal, Vol.29, No.5, pp.390–395, 1986.
[9] Bertogna, Marko, and Michele Cirinei, "Response-Time Analysis for globally scheduled Symmetric Multiprocessor Platforms," Proc. of IEEE Real-Time Systems Symposium, pp.149–158, 2007.
[10] Guan, Nan, et al., "New Response Time Bounds for Fixed Priority Multiprocessor Scheduling," Proc. of IEEE Real-Time Systems Symposium, pp.387–397, 2009.
[11] Sun, Youcheng, et al., "Improving the Response Time Analysis of Global Fixed-Priority Multiprocessor Scheduling," Proc. of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE Xplore, 2014.
[12] Sun, Youcheng, and Marco Di Natale, "Pessimism in multicore global schedulability analysis," Journal of Systems Architecture, Vol.97, pp.142–152, 2019.
[13] Zhou, Quan, et al., "Response Time Analysis for Tasks with Fixed Preemption Points under Global Scheduling," ACM Transactions on Embedded Computing Systems (TECS), Vol.18, No.5, pp.1–23, 2019.