International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

Delayed Rate Monotonic Algorithm with First-fit Partitioning

Saeed Senobary[1*], Mahmoud Naghibzadeh[2]

[1] Imam Reza International University, Mashhad, Iran

[2] Dept. of computer engineering, Ferdowsi University of Mashhad, Mashhad, Iran

*Corresponding Author's E-mail: s.snobary@imamreza.ac.ir

Abstract

The problem of scheduling periodic tasks has been studied extensively since its first introduction by Liu and Layland. They proposed utilization bound, called L&L bound, to guaranty feasibility of task set under rate monotonic policy on a single processor. Recently, some papers have used semi-partitioned scheduling to increase overall system utilization. In semi-partitioned scheduling, most tasks are statically assigned to processors, while a few tasks are split into several subtasks and each assigned to a different processor. In this paper a new semi-partitioned scheduling algorithm called FFDRM, which is based on delayed rate monotonic, is proposed for multiprocessor systems by which the system utilization of most of processors is raised up to L&L bound. Delayed rate monotonic is an improved version of rate monotonic. It is proved that the lower bound of feasibility of a system which includes two tasks under delayed rate monotonic is equal to one. FFDRM uses this ability of delayed rate monotonic to achieve higher system utilization than previous works. The feasibility of tasks which are partitioned by FFDRM is formally proved.

Keywords: real-time embedded systems, semi-partitioned scheduling, first-fit allocation, delayed rate monotonic algorithm.

1. Introduction

Today real-time systems are found in various application areas such as space systems and medical imaging. One of the significant parts of real-time systems is scheduling algorithm which should assure the real-time performance while making the most effective

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

use of the available processing capacity [1]. Real-time scheduling algorithms have been studied since 1960s. Liu and Layland proposed an optimal scheduling algorithm, called rate monotonic, for uniprocessor systems [2]. They also proposed an admission control based on system utilization. The scheduling algorithms on multiprocessor systems are classified as partitioned scheduling and global scheduling, with each category having its own problems. In partitioned scheduling, each task is statically assigned to one processor and always executed on that processor. In global scheduling, there is one queue for entire system and tasks may execute on different processors. The upper bound for partitioned scheduling algorithms on multiprocessor systems is known to be 50% [3]. Recent papers have made another class called semi-partitioned scheduling by which the system utilization is enhanced [4-9]. In semi-partitioned scheduling, most tasks are statically assigned to processors, while a few tasks are split into several subtasks and each assigned to a different processor. The main challenge is how to split tasks while the feasibility of tasks and subtasks should be assured.

The system utilization, in some papers, is raised up to L&L bound [6, 8, 9]. An approach called SPA is proposed to schedule periodic tasks on multiprocessor systems based on rate monotonic algorithm [9]. This approach can safely schedule any system in which the system utilization is no larger than L&L bound. Tasks are sorted in descending order of their periods and a processor with the least workload is selected for assigning current task. This approach splits the highest priority task on each processor. The number of created subtasks is at most equal to $M-1$, where M is the number of processors. Another approach, called HSP, is proposed in paper [6] to schedule sporadic tasks on multi-core systems. The scheduling policy is based on rate monotonic algorithm. This approach uses the harmonic relation among tasks to achieve higher system utilization. The same as SPA, this approach splits the highest priority task. RMLS is another approach in which the system utilization is raised up to L&L bound [8]. First, tasks are sorted in ascending order of their periods and

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, ISSN: 2305-0543
Available online at: http://www.aeuso.org

then assigned to the first processor. If the system utilization of this processor is elevated to L&L bound, then tasks are assigned to the next processor. If current task is not completely accommodated by a processor, it is split into two subtasks. The first subtask is assigned to the processor and the second subtask is put back in front of the queue. Based on the sorting method, the lowest priority task on each processor is split. The proposed approach of this paper, FFDRM, is another approach by which the system utilization of most of processors is raised up to L&L bound. The scheduling policy is based on delayed rate monotonic [10]. Delayed rate monotonic is an improved version of rate monotonic. It is proved that a system which includes two tasks is feasible under delayed rate monotonic, if and only if the system utilization is lower than or equal to one [10]. It means that the lower bound of feasibility of a processor consisting of two tasks and under delayed rate monotonic is equal to one. In this paper, the feasibility of tasks under delayed rate monotonic, the same as rate monotonic, is formally proved in order to use L&L bound as admission control of FFDRM.

First-fit is one of the heuristic methods used in partitioned scheduling algorithms. This method allocates compatible tasks together so that the system utilization is improved [11]. FFDRM uses first-fit method for assigning tasks to the processors. It is formally proved that any system which is successfully partitioned by FFDRM will be safely scheduled under delayed rate monotonic. The simulation results demonstrate that FFDRM achieves significantly higher system utilization in comparison with previous works. The rest of this paper is organized as follows: Section 2 introduces the basic concepts, Section 3 describes the new approach, Section 4 considers the simulation results and finally the conclusion is presented in Section 5.

2. Basic Concepts

In the proposed system, tasks are periodic and their deadline parameters (i.e., relative deadline) are assumed to be equal to their periods. A request of task τ_i, $i=\{1,\ldots, n\}$, is called a job. Every task τ_i is modelled by two parameters:

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

- C_i: the worst case execution time required by task τ_i on each job.
- T_i: the time interval between two jobs of task τ_i.

Every job of task τ_i should be completed before the next job of the same task arrives. Response time of a job is the time span from job arriving up to its execution completion. Liu and Layland proved that the worst response time of task τ_i occurs when it simultaneously requests with all the higher priority tasks [2]. Task τ_i is feasible if its worst response time, R_i, is lower than or equal to its period ($R_i \leq T_i$). Utilization of task τ_i is defined by $U_i = \frac{C_i}{T_i}$. In this system, we want to assign task set $\Gamma = \{(C_1, T_1), (C_2, T_2), \ldots, (C_n, T_n)\}$ to m processors, $\{M_1, \ldots, M_m\}$. Tasks are sorted in descending order of their periods, i.e., $i < j \rightarrow T_i \geq T_j$.

Definition 1. The utilization of task set Γ is defined by:

$$U(\Gamma) = \sum_{i=1}^{n} U_i \tag{1}$$

Definition 2. The system utilization of M_i, U^{M_i} $i=\{1, \ldots, m\}$ is the utilization of task set Γ' which is assigned to M_i.

Definition 3. The average system utilization of task set Γ with m processors is:

$$U_m(\Gamma) = \frac{U(\Gamma)}{m} \tag{2}$$

We use $\Theta(n)$ to symbolize the L&L bound, where n is the number of tasks.

$$\Theta(n) = n * (2^{\frac{1}{n}} - 1) \tag{3}$$

FFDRM uses semi-partitioned scheduling to raise up the system utilization of each processor to $\Theta(n)$.

Definition 4. Suppose task set $\Gamma' = \{\tau_1, \ldots, \tau_{n'}\}$ is assigned to processor M_i, $i=\{1, \ldots, m\}$. Now, the remaining system utilization of M_i is defined by:

$$\lambda(M_i) = \Theta(n'+1) - U(\Gamma) \tag{4}$$

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

A semi-partitioned scheduling includes two phases: partitioning and scheduling. In the partitioning phase, it is defined how to assign and split each task to a fixed processor. Most tasks are statically assigned to the processors; such tasks are called non-split task while other tasks are called split-task which split into two subtasks. The highest priority task on each processor is split, therefore a task is probably split into several subtasks and each assigned to a different processor. It is necessary to synchronize subtasks of a split-task. Suppose task τ_i is split into several subtasks, $\{\tau_{i1}, \tau_{i2}, \ldots, \tau_{ip}\}$. Now, subtask τ_{ij} starts its execution when the prior subtasks, $\tau_{i1}, \ldots, \tau_{ij-1}$, are finished. It means release time of subtask τ_{ij} is equal to the total worst response time of prior subtasks. Release time of a subtask, i.e., τ_{ij}, is considered as

$$\Phi(\tau_{ij}) = \sum_{k=1}^{j-1} R_{ik} \tag{5}$$

Where R_{ik} is the worst response time of subtask τ_{ik} in its own processor. As mentioned earlier, the highest priority task is split, therefore the worst response time of subtask τ_{ik} is equal to its worst case execution time. The second phase of a semi-partitioned scheduling is the policy to determine how to schedule assigned tasks on each processor. The Scheduler used within each processor is based on delayed rate monotonic, which is an improved version of rate monotonic. In rate monotonic algorithm every task with lower period has higher priority and this algorithm is pre-emptive. So, the lowest priority task has the highest probability for overrun. In delayed rate monotonic all tasks have a secure delay except the task with the lowest priority. Therefore, in this scheduler, these two queues are defined for tasks:

- Ready: task with the lowest priority directly enters this queue and other tasks enter when their delays are over.
- Delay: all tasks except the lowest priority one enter this queue as they make a request.

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

A task in the ready queue has higher priority over all tasks in the delay queue. If no task exists in the ready queue, then a task is selected from the delay queue to run next. Tasks in both ready and delay queues are scheduled based on rate monotonic algorithm. To prove the feasibility of tasks under delayed rate monotonic, the same as rate monotonic, the definition of delay in FFDRM, has been changed. As mentioned, two types of tasks exist in our proposed system, therefore we define two kinds of delay for tasks.

- For non-split tasks, i.e., τ_i, delay is:
  - If τ_i is the lowest priority task, then delay is equal to zero.
  - Otherwise, delay is equal to $T_i - R_i$.
- For split-tasks, i.e., τ_i, delay is equal to zero.

Theorem 1. To use L&L bound as admission control of FFDRM, any task which is feasible under rate monotonic, will be feasible under delayed rate monotonic as well.

Proof. Suppose task τ_i is feasible under rate monotonic, therefore its worst response time, R_i, is lower than or equal to its period. As mentioned above, scheduling in both ready and delay queues is based on rate monotonic. So:

If task τ_i is the lowest priority task, it is feasible under delayed rate monotonic, because it directly enters the ready queue. For other non-split tasks, i.e., τ_i, delay is equal to $T_i - R_i$. the worst response time of task τ_i in the ready queue is equal to R_i, therefore the maximum response time of task τ_i under delayed rate monotonic, is equal to $T_i - R_i + R_i = T_i$. The maximum response time of task τ_i under delayed rate monotonic at most is equal to its period. So, this task is feasible under delayed rate monotonic, as well.

Delay of split-tasks is equal to zero, therefore they are feasible under delayed rate monotonic, because they directly enter the ready queue.□

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

3. FFDRM Algorithm

Partitioning phase of FFDRM includes two steps. Algorithm 1 demonstrates the pseudo code of the first step of partitioning phase of FFDRM. As mentioned, tasks are sorted in descending order of their periods. Therefore, the lowest priority task is in front of the queue of unassigned tasks. In the first step, tasks which entirely be accommodated into a processor are assigned. Processors are selected by a first-fit method. The first-fit method is used in order to collect the compatible tasks together. In this method the first processor, M_1, is filled until the current task, i.e., τ_i, which is in front of the queue of unassigned tasks, cannot entirely be accommodated into the processor M_1, Line 32 of Algorithm 1. It means that:

$$U^{M_1} + U_i > \Theta(n'+1)$$

Where n' is the number of tasks which are assigned to processor M_1. If the current task cannot entirely be accommodated into processor M_1, then it is assigned to the second processor. This process repeats for other processors, $M_2, \ldots, M_m$.

It is easy to derive the following property from Algorithm1.

Lemma 1. If task τ_i is in front of the queue of unassigned tasks and its utilization is larger than $\Theta(2)$ and an empty processor exist in the queue of processors, then task τ_i is entirely assigned to one processor and this processor is removed from the queue of processors.

Proof. Suppose processors M_a and M_{a+1} is in the queue of processors. Processor M_a is in front of queue and it has some tasks which are assigned earlier, $\tau_1, \ldots, \tau_{n'}$. Now, we try to assign task τ_i to this processor. In Line 24 of Algorithm 1, it is checked whether total utilization of task τ_i and system utilization of processor M_a are less than $\Theta(n'+1)$. If it is correct, then task τ_i is assigned to processor M_a. But, this condition is not correct, because $U_i \geq \Theta(2)$ and $\Theta(2) > \Theta(n'+1)$. Therefore, the next processor, M_{a+1}, is selected, Line 33 of Algorithm 1. In the next iteration, task τ_i is assigned to empty processor M_{a+1} and this processor is removed from queue of processors, Lines 7 to 14 from Algorithm 1.□

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

As mentioned, it is proved that the lower bound of feasibility of a processor which includes two tasks under delayed rate monotonic is equal to one [10]. Therefore, when we decide to add the current task, i.e., τ_i, to a processor, i.e., M_a, it is checked whether

$$\text{Processor } M_a \text{ has only one task and } \Theta(2) \le U_i + U^{M_a} \le 1 \qquad (6)$$

If this condition is correct, then, task τ_i is assigned to the processor M_a and no other task or subtask is assigned to this processor.

Algorithm 1. Step one of partitioning phase of FFDRM

Input: task set $\Gamma=\{\tau_1 \ldots,\tau_n\}$ and the list of processors $\{M_1, \ldots, M_m\}$.
// Sort all tasks in descending order of their periods

```
M_a = 1
n'=0
For each task τ_i in Γ do
  If M_a is equal to m then
    Break for.
  End if
  If n' is equal to zero then
    Assign τ_i to processor M_a.
    n' = n' +1.
    If U_i ≥ Θ(2)
      Remove M_a from queue of processors.
      M_a = M_a + 1.
      n' = 0.
    End if
    Continue.
  End if
  If n' is equal to one and Θ(2) ≤ U_i + U^{M_a} ≤ 1 then
    Assign τ_i to processor M_a.
    Remove M_a from queue of processors.
    M_a = M_a + 1.
    n' = 0.
    Continue.
  End if
  If U^{M_a} + U_i ≤ Θ(n'+1) then
    Assign τ_i to processor M_a.
    n' = n' +1.
    If U^{M_a} is equal to Θ(n'+1) then
      Remove M_a from queue of processors.
      M_a = M_a + 1.
      n' = 0.
    End if
    Else
      M_a = M_a + 1.
      n' = 0.
    End if
End for
```

In this situation, the system utilization of processor M_a is probably raised up to 100%. This processor has two non-split tasks which are safely scheduled under delayed rate monotonic policy.

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

In the first step of partitioning phase of FFDRM, most tasks are statically assigned to the processors. It is easy to derive the following property form Algorithm 1.

Lemma 2. At least one task is entirely assigned to each processor.

Proof. Proof follows the assigning approach which is used by FFDRM.□

Considering Lemma 2, the lowest priority task in each processor is a non-split task. After Algorithm 1, some processors are removed from queue of processors, Lines 11, 19, or 28. These processors are sufficiently filled. On the other hand, there are some processors in the queue of processors which still have some unfilled capacity.

Now, in the next step of partitioning phase of FFDRM, if there are some unassigned tasks in the queue, i.e., τ_j, τ_{j+1}..., τ_n, then, in each iteration, a processor, i.e., M_a, which has the largest remaining system utilization, $\lambda(M_a)$ is maximum, is selected for assigning task τ_j. Algorithm 2 demonstrates the pseudo code of step two of partitioning phase of FFDRM.

If task τ_j cannot entirely be assigned to processor M_a, then it is split into two subtasks, τ_{j1} (C_{j1}, T_j) and τ_{j2} $(C_j - C_{j1}, T_j)$. The first subtask τ_{j1} is assigned to processor M_a.

Algorithm 2. Step two of partitioning phase of FFDRM

// step 2: assign the remaining tasks	11. *Split τ_i into two subtasks τ_{i1} and τ_{i2}, in which $\frac{C_{i1}}{T_i} + U^{M_a}$ is equal to $\Theta(n'+1)$.*
1. ***For each** task τ_i in the remaining tasks **do***	12. *Assign subtask τ_{i1} to processor M_a.*
2. ***If exist** a processor in queue of processors **then***	13. *Remove M_a from queue of processors.*
3. *Select a processor, i.e., M_a, which $\lambda(M_a)$ is maximum*	14. *Put back τ_{i2} in front of queue of unassigned tasks.*
4. *n' = number of tasks in M_a.*	15. ***End if***
5. ***If** $U^{M_a} + U_i \le \Theta(n'+1)$ **then***	16. ***Else***
6. *Assign τ_i to processor M_a.*	17. *Return 'cannot assign task set Γ to m processors'*
7. ***If** τ_i is the last subtask of a task **or** U^{M_a} is equal to $\Theta(n'+1)$ **then***	18. ***End for***
8. *Remove M_a from queue of processors.*	19. *Return 'task set Γ are successfully assigned to m processors'*
9. ***End if***	
10. ***Else***	

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

The worst case execution time of τ_{j1} is obtained by which the system utilization of processor M_a with n' tasks, is equal to L&L bound.

$$\frac{C_{j1}}{T_j} + U^{M_a} = \Theta(n'+1)$$

After assigning the first subtask to processor M_a, this processor is removed from queue of processors since no other task or subtask is assigned to this processor. The second subtask τ_{j2} is put back in front of the queue. Based on the sorting method, the highest priority task in each processor is split. If the subtask τ_{j2} cannot entirely be assigned to the host processor, then it is again split. Therefore, in the partitioning phase of FFDRM, a task, i.e., τ_k, is probably split into several subtasks $\{\tau_{k1}, \tau_{k2}, \ldots, \tau_{kp}\}$. After assigning the last subtask of a split task, τ_{kp}, to a processor, this processor is removed from the queue of processors, Lines 7 to 9 from Algorithm 2. Feasibility of subtasks is proved by Lemma 3.

Lemma 3. If task τ_i, by partitioning phase of FFDRM, is split into several subtasks $\{\tau_{i1}, \ldots, \tau_{ip}\}$, then, respect to the release time of each subtask, all subtasks of τ_i can meet their deadline parameters.

Proof. Considering partitioning phase of FFDRM, the highest priority task on each processor is split and after that these processors are removed from the queue of processors since no other task or subtask is assigned to them. Scheduling policy is based on delayed rate monotonic and it is mentioned earlier that delay of subtasks are assumed to be zero. Therefore, any job of these subtasks enter the ready queue. So, each subtask τ_{ij}, $j = \{1,..,p-1\}$ can meet its deadline parameters, because the system utilization of each processor which includes each subtask τ_{ij}, is equal to L&L bound and it guaranties the feasibility of task set under delayed rate monotonic, the same as rate monotonic, according to Theorem 1.

The release time of last subtask, $\Phi(\tau_{ip})$, considering the relation (5), is equal to

$$\Phi(\tau_{ip}) = \sum_{j=1}^{p-1} R_{ij}$$

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

Scheduling in the ready queue is based on rate monotonic, so the worst response time of each subtask is equal to its worst case execution time. Therefore, we have

$$\Phi(\tau_{ip}) = \sum_{j=1}^{p-1} C_{ij}$$

Deadline parameters of last subtask is equal to

$$T_i - \Phi(\tau_{ip})$$

It is larger than the worst case execution time of τ_{ip}, because

$$T_i \geq C_i$$

$$C_i = C_{i1} + C_{i2} + \ldots + C_{ip}$$

$$T_i - (C_{i1} + C_{i2} + \ldots + C_{ip-1}) \geq C_{ip}$$

As mentioned earlier, after assigning the last subtask to a processor, no other task or subtask is assigned to this processor. Therefore, the last subtask τ_{ip} is the highest priority task in its own processor and it can meet its deadline parameters, because the system utilization of processor which includes subtask τ_{ip}, is probably lower than L&L bound and it guaranties the feasibility of task set under delayed rate monotonic, the same as rate monotonic, according to Theorem 1.□

If queue of tasks is empty, then, we can say that all tasks are successfully assigned to the processors. With the partitioning phase of FFDRM, the system utilization of most of processors is raised up to L&L bound, while, considering Condition 6, a few processors probably exist in which the system utilization of them is equal to 100%. After assigning tasks to the processors, it is time for scheduling phase. As mentioned, delayed rate monotonic is used within each processor. Now, by Lemma 4, the feasibility of non-split tasks which are partitioned by FFDRM, is proved.

Lemma 4. After partitioning phase of FFDRM, non-split tasks can be safely scheduled under delayed rate monotonic.

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

Proof. The system utilization of most of processors is equal to L&L bound, therefore these processors are safely scheduled under rate monotonic algorithm. We proved in Theorem 1 that delayed rate monotonic can safely schedule any system which is feasible under rate monotonic, as well. So, non-split tasks which are assigned to these processors are safely scheduled under delayed rate monotonic.

Any other processors which the system utilization of them is raised up to 100%, have two non-split tasks, i.e., τ_i and τ_j, while $U_i + U_j \leq 1$. The feasibility of them is presented in paper [10].

Therefore, we formally proved that all non-split tasks which are partitioned by FFDRM can be safely scheduled under delayed rate monotonic.□

Now, we can derive the following lemma:

Lemma 5. If task set Γ is successfully assigned to m processors by partitioning phase of FFDRM and scheduled under delayed rate monotonic, then all tasks can meet their deadline parameters.

Proof. The feasibility of split-tasks and non-split tasks are assured considering Lemma 3 and 4, respectively.□

4. Simulation Results

In this section, the performance of the proposed approach in comparison with three prior works is investigated. In our simulations, 5000 task sets with random utilization between $[\Theta(n)_{n\to\infty} * L, L]$ are randomly generated. L determines the maximum system utilization of each task set and its value will be one of the values of the following set.

$$L \in \{4, 8, 16, 32, 64\}$$

The number of processors is assumed to be variable so that each approach can successfully partition these tasks sets. Indeed, by assigning these task sets to the lower

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

processors, higher system utilization can be achieved. The utilization of each task is also randomly generated between [0.01, 1]. Distribution of our random function is uniform.

We compare FFDRM with SPA [9], HSP [6] and RMLS [8]. The percentage of the number of extra processors under each of the four aforementioned approaches is demonstrated in Table 1. This value is obtained by Formula 7.

$$percentage\ \ of\ number\ of\ extra\ processors = \frac{NP_{extra} - NP_{opt}}{NP_{opt}} * 100 \qquad (7)$$

NP_{extra} is the number of processors which are used by an approach and NP_{opt} is the number of required processors under an optimal approach for assigning task sets. We assumed that the number of required processors under an optimal approach is equal to $L * 5000$.

For instance, when L is equal to 4, we generated 5000 task sets in which system utilization of each task set did not exceed 4. Therefore, the optimal state occurs when an approach assigns such task sets to 20000 processors. In this situation, the system utilization of each processor is almost equal to 100%.

Table 1. The percentage of the number of extra processors.

L	4	8	16	32	64
FFDRM	13%	9.3%	7.3%	6.2%	5.6%
SPA	14.8%	11.5%	9.7%	8.75%	8.2%
HSP	14%	10%	9%	8.6%	7.7%
RMLS	14.5%	11.4%	9.1%	8.7%	7.8%

As shown in Table 1, when L is equal to 4, the percentage of the number of extra processors under FFDRM is 13%. It means that the number of required processors for assigning 5000 task sets in which system utilization of each task set does not exceed 4, under FFDRM, is equal to 22600. SPA, HSP, and RMLS assign such task sets on 22960,

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

22800, and 22900 processors, respectively. As illustrated in Table 1, the number of required processors under FFDRM compared to other prior works is less.

We define another parameter called average system utilization U_{ave}, to evaluate each approach.

$$U_{ave} = \frac{\sum_{i=1}^{5000} (U(\Gamma))_i}{number\ of\ required\ processors} \tag{8}$$

Figure 1(a) illustrates the average system utilization of each approach. As shown in Figure 1(a), the average system utilization of FFDRM is more than other approaches. This achievement occurs because FFDRM uses first-fit method to allocate compatible tasks together. Another motivation which makes FFDRM to achieve higher average system utilization is the lower bound of feasibility of a system includes two tasks. As mentioned, the lower bound of feasibility of a system includes two tasks is 100% under delayed rate monotonic; therefore, after partitioning phase of FFDRM, some processors exist in which the system utilization of any of them is between $[\Theta(2),1]$. Table 2 demonstrates the percentage of the number of such processors. These processors have two tasks and can be safely scheduled under delayed rate monotonic. For instance, when L is equal to 8, then, 15% of 40000 processors includes two tasks in which the system utilization of any of them is between $[\Theta(2),1]$.

Table 2. The Percentage of the number of processors with system utilization between $[\Theta(2),1]$.

Maximum system utilization	*4*	*8*	*16*	*32*	*64*
% number of processors in each test	*15.6%*	*15%*	*14.5%*	*14.6%*	*14.4%*

Another benefit of two motivations of partitioning phase of FFDRM is in the number of created subtasks. As mentioned, these motivations are the usage of first-fit method and lower bound of feasibility of a system includes two tasks. Figure 1(b) demonstrates the

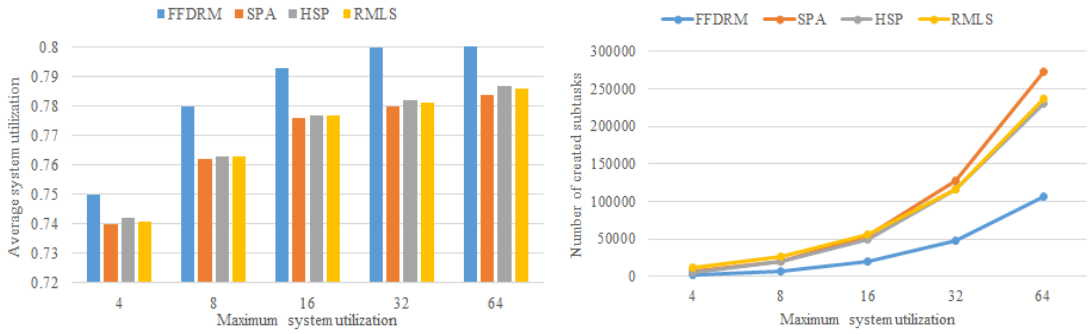
International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

number of created subtasks by all approaches. As shown in Figure 1(b), the number of created subtasks by FFDRM is lower than that of the three prior works.

5. Conclusion

In this paper a new semi-partitioned scheduling algorithm, called FFDRM, is proposed by which the system utilization of most of processor is raised up to L&L bound. After partitioning phase of FFRDM, some processors exist in which the system utilization of any of them is between $[\Theta(2),1]$. Feasibility of tasks which are partitioned by FFDRM is formally proved. The simulation results demonstrate that FFDRM significantly improves the scheduling performance compared with prior works, in terms of the average system utilization and the number of created subtasks. First-fit method and the lower bound of feasibility of a system under delayed rate monotonic are the motivations which make FFDRM achieve these improvements. Proposing a special admission control for delayed rate monotonic is one of our future purposes.



(a) (b)

Figure 1. Simulation results based on variable number of processors for investigating (a) the average system utilization (b) the number of created subtasks.

References

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

[1]. Davis R. I., Burns A., "*A Survey of Hard Real-Time Scheduling for Multiprocessor Systems",* ACM Computing Surveys, (2011), vol. 43, no. 4.

[2]. Liu and J.W. Layland, "*Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment",* Journal of the ACM, (1973) pp. 46-61.

[3]. Andersson, B. and J. Jonsson, "*The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%",* in Proceedings of 15th Euromicro Conference on Real-Time Systems, (2003), pp. 33-40.

[4]. Arvind Kandhalu, K.L., Junsung Kim, Ragunathan (Raj) Rajkumar, "*pCOMPATS: Period-Compatible Task Allocation and Splitting On Multi-Core Processors",* IEEE 18th Real Time and Embedded Technology and Applications Symposium, (2012).

[5]. Bhatti, M.K., C. Belleudy, and M. Auguin. "*A Semi-partitioned Real-time Scheduling Approach for Periodic Task Systems on Multicore Platforms",* in SAC '12 Proceedings of the 27th Annual ACM Symposium on Applied Computing, (2012), Trento.

[6]. Fan, M. and G. Quan, "*Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform"*, in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, (2012), Dresden, pp. 503 - 508

[7]. George, L., P. Courbin, and Y. Sorel, "*Job vs. portioned partitioning for the earliest deadline first semi-partitioned scheduling",* Journal of Systems Architecture, (2011), vol. 57 pp. 518-535.

[8]. Naghibzadeh, M., et al., "*Efficient Semi-Partitioning and Rate-Monotonic Scheduling Hard Real-Time Tasks on Multi-Core Systems"*, in 8th IEEE International Symposium on Industrial Embedded Systems (SIES), (2013) Porto, pp. 85-88.

[9]. Nan Guan, M.S., Wang Yi, Ge Yu, "*Fixed-Priority Multiprocessor Scheduling with Liu and Layland's Utilization Bound"*, Real-Time and Embedded Technology and Applications Symposium (RTAS), 16th IEEE, (2010) pp. 165-174.

[10]. Naghibzadeh, M. and K.H.K. Kim, "*The yielding-first rate-monotonic scheduling approach and its efficiency assessment"*, Computer Systems Science & Engineering, (2003), vol. 18 pp. 173-180.

[11]. Sylvain Lauzac, R.M., Daniel Mosse, "*An Improved Rate-Monotonic Admission Control and Its Applications"*, IEEE Transactions on Computers, (2003), vol. 52, no.3 pp. 337-350.

International Journal of Mechatronics, Electrical and Computer Technology

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

Authors

Saeed Senobary received his BS degree in computer engineering from Institute of Darolfonun Bojnourd, Iran, in 2011 and MS degree in computer software engineering from Imam Reza International University, Mashhad, Iran, in 2013. He has been recognized as first rank student in Institute of Darolfonun Bojnourd. His research interests are included real-time systems, especially real-time scheduling algorithms, parallel programing, and recommendation systems.

Mahmoud Naghibzadeh received his BS degree in statistics and computer science from Ferdowsi University of Mashad, Iran and his MS and PhD degrees in computer science and Computer Engineering, respectively, from University of Southern California, USA. He is now a full professor at the Deparment of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Prof. Naghibzadeh is currently teaching graduate courses and supervising both MS and PhD students. In addition, he is the director of Knowledge Engineering Reasearch Group (KERG) laboratory. His research interests include the schedulig aspects of real-time systems, Grid, Cloud, Multiprocessor, and Multicore. Besides, another subject of his interest is Bioinfomatics computer algorithms. He has published numerous papers in international journals and conference proceedings as well as eight books in the field of computer science and engineering.

Prof Naghibzadeh was the general charir of an international computer conference and technical chair of two others. He is the reviewer of many journals and a member of many computer societies as well as a senior

Vol. 4(10), Jan, 2014, Special Number, pp. 1045-1062, **ISSN**: 2305-0543
Available online at: http://www.aeuso.org

member of IEEE. He is the recipiant of many awards including MS and PhD study award and outstanding professor award.