Hindawi Publishing Corporation
Journal of Engineering
Volume 2014, Article ID 485361, 13 pages
http://dx.doi.org/10.1155/2014/485361

Research Article

Performance Evaluation of New Joint EDF-RM Scheduling Algorithm for Real Time Distributed System

Rashmi Sharma and Nitin

Department of Computer Science & Engineering and Information & Communication Technology, Jaypee University of Information Technology, Waknaghat, Solan, Himachal Pradesh 173234, India

Correspondence should be addressed to Nitin; delnitin@ieee.org

Received 27 May 2013; Accepted 25 November 2013; Published 22 January 2014

Academic Editor: WaiKeung Wong

In Real Time System, the achievement of deadline is the main target of every scheduling algorithm. Earliest Deadline First (EDF), Rate Monotonic (RM), and least Laxity First are some renowned algorithms that work well in their own context. As we know, there is a very common problem Domino's effect in EDF that is generated due to overloading condition (EDF is not working well in overloading situation). Similarly, performance of RM is degraded in underloading condition. We can say that both algorithms are complements of each other. Deadline missing in both events happens because of their utilization bounding strategy. Therefore, in this paper we are proposing a new scheduling algorithm that carries through the drawback of both existing algorithms. Joint EDF-RM scheduling algorithm is implemented in global scheduler that permits task migration mechanism in between processors in the system. In order to check the improved behavior of proposed algorithm we perform simulation. Results are achieved and evaluated in terms of Success Ratio (SR), Average CPU Utilization (ECU), Failure Ratio (FR), and Maximum Tardiness parameters. In the end, the results are compared with the existing (EDF, RM, and D_R_EDF) algorithms. It has been shown that the proposed algorithm performs better during overloading condition as well in underloading condition.

1. Introduction and Motivation

Real Time Distributed System is a distributed system with real time properties. We can say that RTDS is a combination of RTS and distributed system (Figure 1). Properties of real time tasks are applied on the distributed system or concept of a distributed system is implemented on RTS. The following is the first appearance of two main components of RTDS.

1.1. Real Time System (RTS). RTS is a system in which execution of tasks has some time restrictions (deadline) [1–3]. Based on the execution of real time task RTS falls into the following categories.

(i) *Hard RTS.* Task execution by assigning deadline is restrictive. Missing deadline will produce incurable results for the entire system [4].

(ii) *Soft RTS.* Although missing deadline is not enviable in RTS, but in soft RTS tasks could miss some deadline that will not affect the working of system [5].

(iii) *Firm RTS.* If tasks complete their execution before the deadline, they gain more rewards [4, 6]. It is a special type of soft RTS.

EDF, RM, and Least Laxity First are some basic scheduling algorithms that execute real time tasks on the basis of their deadline, interarrival period, tardiness, and so forth. Every scheduling algorithm is having some drawbacks, for example, EDF is not functioning well in overloading condition and RM in underloading condition [7, 8].

1.2. Distributed System (DS). Distributed system is an arrangement of several processors/nodes followed by some interconnection topologies. The distribution of load to various processors increases the performance of the entire system. In DS, on the basis of nodes utilization load is balanced in between processors [9–11]. Task migration and duplication are two methodologies that help in balancing the processor load.

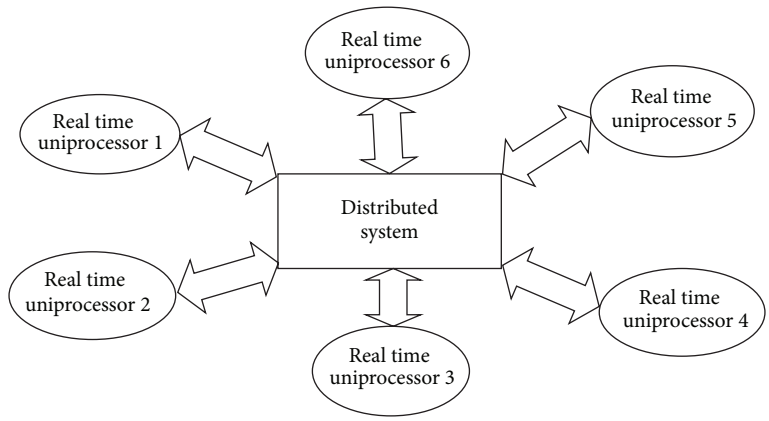

Figure 1: Real time distributed system.

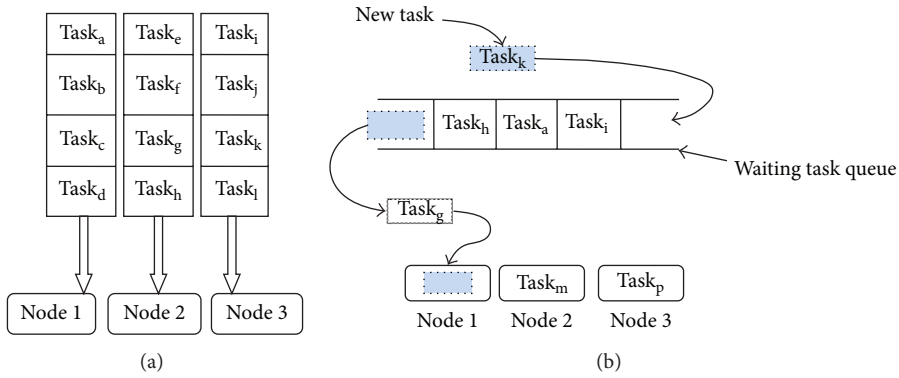

Figure 2: Distributed system scheduling.

(i) *Load Migration. In order to balance the load of DS task migration methodology relocates the victim task from one processor (source) to another processor (destination).* This technique can be applied to dependent (DAG) as well as independent tasks [12, 13].

(ii) *Task Duplication. Task duplication method is given on the basis of computation to communication ratio (CCR) in between processors.* This technique is mostly applicable on dependent tasks (DAG) in order to balance the load as well as minimize the execution cost of complete DAG [14, 15].

Moreover, Tasks in a distributed system are assigned to processors by using the following schedulers (Figure 2).

(i) *Partitioned Scheduler.* Tasks are mapped to processors deterministically for execution. Scheduler statically assigns tasks to processors and migration of task instance is prohibited here [16, 17].

(ii) *Global Scheduler. All tasks are kept in a global queue from where tasks to other processors will be assigned for the death penalty.* This scheduler dynamically assigns tasks to nodes and allows task migration in between processors [17, 18]. Global scheduler is appealing in such systems where average as well as the worst case response time (WCRT) is important. In queuing theory, single queue scheduling generates healthier average response time as compared to queue per processing scheduler [17, 19]. Hence, in RTDS this scheduler is superior to partitioned scheduler.

In this paper we have explained a new joint EDF-RM scheduling algorithm which behaves optimally in both circumstances (underloading as well as overloading). Here, the utilization bound of RM is used to boundary limit of the global queue instead of processors ρ_{UB} and tasks will be assigned to randomly selected processors. Moreover, for the execution of tasks the EDF scheduling is used on every processor. Our proposed algorithm is divided into the following two modules:

(a) real time job assignment in distributed system (using global scheduler);

(b) real time task execution on allotted processors of distributed system (including task migration).

Reason behind using RM utilization bound is to safeguard our system from overloading condition. Additionally, in order to deal dynamically with tasks the EDF scheduling algorithm is implemented. Load balancing in RTDS is availed by using the global scheduler because task migration is permissible in this scheduler.

Table 1: Symbols and definitions.

Symbols	Definition
T	Task-set
τ_{arrival}	Task arrival time
τ_{period}	Interarrival period of task
τ_{dline}	Task deadline
$\tau_{\text{utilization}}$	Per task utilization
τ_{priority}	Task priority
$\tau_{p,a}$	Task a of p processor
$\rho_{\text{utilization}}$	Per processor utilization
ρ_{UB}	Processor upper bound
$E\rho_{\text{utilization}}$	Efficient processor utilization
SR	Success Ratio
FR	Failure Ratio
MR	Migration ratio
A_preemption	Average number of preemptions
B_Q	Boundary limit of global task queue
TST	Total time of scheduling
tard_{τ_i}	Tardiness of given task

The remainder of the paper is organized as follows. Existing real time scheduling algorithms have been discussed in Section 2. Section 3 explains the proposed joint EDF-RM scheduling algorithm. In Section 4, performances of EDF, RM, D_O_EDF, and D_R_EDF [8] along with proposed algorithm have been analyzed with simulation results. Finally, the conclusion is presented in Section 5.

Before moving towards the next section, let us have a glance on the symbols that are utilized in the entire composition. Table 1 demonstrates these symbols and their denotation.

2. Preliminaries and Background

Before giving the elucidation of our proposed work we would like to enlighten scheduling algorithms associated with our work. This paper uses RM and EDF scheduling algorithms together and tries to resolve the downside of EDF (Domino's effect) and RM as well. As we know that in distributed system load balancing is an important issue. Hence, proposed algorithm also amplifies the working capability in overloading or underloading situations. These above stated two situations (loading conditions) are checked by recalculations of per task/processor utilization.

2.1. Real Time Schedulers. Tasks having real time properties and scheduled on RTS are real time tasks. Scheduling is done by two types of scheduler modules.

(i) *Dynamic Scheduler.* Dynamic scheduler assigns priorities to tasks at run time. This scheduler reevaluates the information like priorities, resource availability after the arrival of current task. Due to the dynamism of tasks priority, already running tasks have to be preempted by the arrival of new task [20].

(ii) *Static Scheduler.* Under this scheduler, priorities of tasks are predefined or we can say static. It cannot be altered by the arrival of new tasks. Priority of $\tau_m > \tau_n$, where $m < n$. Generally, this scheduler works on dependent tasks because communication cost in between tasks is reduced [20].

Let us consider that T is a set of n real time tasks $T = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \ldots, \tau_n\}$ arriving on processor ρ. Hence,

$$\tau_{\text{utilization}} = \frac{\tau_{\text{wcet}}}{\tau_{\text{period}}},$$
$$\rho_{\text{utilization}} = \sum_{i=1}^{n} \tau_{i\ \text{utilization}}. \tag{1}$$

RTS works with dynamic as well as static tasks. The EDF scheduling algorithm works well with dynamic and RM schedule static tasks efficiently. We are dealing with RTDS in this paper; therefore we consider the case of overloading in each case.

2.2. Earliest Deadline First Scheduling Algorithm. EDF is a dynamic scheduler that follows the principle: the nearer the deadline the higher the priority of task [21, 22]. It assigns priorities to tasks dynamically. Consider

$$\tau_{\text{priority}} \propto \frac{1}{\tau_{\text{dline}}}. \tag{2}$$

Before assigning the priorities to the task, scheduler first checks the acceptance test by inspecting the utilization value of task as well as processor. Here task-set T is schedulable

$$\text{if } \rho_{\text{utilization}} \leq 1. \tag{3}$$

Let us consider the following examples that explain the behavior of EDF in overloading case in both single and multiple processors (distributed computing).

2.2.1. Example of EDF in Uniprocessor. As we have mentioned in Algorithm 1 and Table 2 the arrival of new tasks preempts already running tasks. In Figure 3 the deadline of task τ_3 is shorter than the task τ_2 and therefore newly arrived task τ_3 preempts τ_2 task because τ_2 task has missed the deadline.

2.2.2. Example of EDF in Distributed System. In case of distributed system, overloaded processor can migrate the victim task to other processors (destination processor) [23].

According to Table 3 and Figure 4, $\tau_{\text{utilization}}$ is 0.45 which qualifies the acceptance test of schedulability but due to task utilization $\rho_{\text{utilization}}$ becomes 1.15 which is greater than 1. Due to the arrival of this third task ρ_1 becomes overloaded and future tasks will also miss their deadline. Therefore, by applying the migration terminology scheduler checks per processor utilization and migrate victim task to the processor having a time slot for its execution ($\rho_{\text{utilization}} < 1$). In the above example ρ_2 has available time slot and its utilization is also less than 1. In distributed system there is some migration cost α, that is, time taken by task to migrate from one processor (source) to another (destination).

Table 2: Arrival time, wcet, period, and deadline of tasks τ_1, τ_2, and τ_3.

Tasks	Estimate arrival time	Computation time (wcet)	Period	Deadline	$\tau_{\text{utilization}} = \frac{\tau_{\text{wcet}}}{\tau_{\text{period}}}$
τ_1	0	1	3	3	0.33
τ_2	1	2	5	5	0.4
τ_3	3	1.8	4	4	0.45

Table 3: Arrival time, wcet, period, deadline, and node for assignment of tasks τ_1, τ_2, τ_3, τ_4, τ_5, and τ_6.

Tasks	Arrival time	Computation time (wcet)	Period	Deadline	$\tau_{\text{utilization}} = \frac{\tau_{\text{wcet}}}{\tau_{\text{period}}}$	Processor
τ_1	0	1	3	3	0.33	ρ_1
τ_2	1	2	5	5	0.4	ρ_1
τ_3	3	1.8	4	4	0.45	ρ_1
τ_4	5	3	6	6	0.5	ρ_2
τ_5	1	0.5	2	2	0.25	ρ_3
τ_6	2	2	4	4	0.5	ρ_3

EDF scheduling algorithm in uniprocessor
BEGIN
(1) **If** $\tau_{\text{utilization}} \leq 1$
(2) **If** $\rho_{\text{utilization}} \leq 1$
(3) The task is schedulable and

(4) assign τ_{priority} of task on given processor
(5) **Else task is non-schedulable**
END

EDF scheduling algorithm in distributed system
BEGIN
(1) **If** $\tau_{\text{utilization}} \leq 1$
(2) **If** $\rho_{\text{utilization}} \leq 1$
(3) **then** task is schedulable and assign τ_{priority} of task on given processor
(4) **Else** migrate the task
(5) **Else task is non-schedulable**
END

Algorithm 1

2.3. Rate Monotonic Scheduling Algorithm. RM scheduling algorithm comes under static scheduler (static priority) and follows the principle: the shorter the interarrival period the higher the priority of task [23, 24]:

$$\tau_{\text{priority}} \propto \frac{1}{\tau_{\text{period}}}. \tag{4}$$

Before assigning the priorities to the task, scheduler first checks the acceptance test by inspecting the utilization value of task as well as processor. For this algorithm T task-set is schedulable on a given processor [25]

$$\text{iff } \rho_{\text{UB}} \leq n\left(2^{1/n} - 1\right), \tag{5}$$

where n is total number of tasks

2.3.1. Example of RM in Uniprocessor. Now next is the implementation of RM scheduling algorithm on the data of Table 2. In overloading condition neither EDF nor RM performs well [7].

The upper bound of RM represents schedulability bound of the number of tasks, and as n increases, it decreases with n (number of jobs). While we compute ρ_{UB} for any number of tasks (as the number of processors increases towards infinity), the value of ρ_{UB} remains 0.6931472035974151. This means that

$$\begin{gathered}
\lim_{n \to \infty} n\left(\sqrt[n]{2} - 1\right) = \ln 2 \approx 0.693, \\
\tau_{1\ \text{utilization}} = \frac{\tau_{\text{wcet}}}{\tau_{\text{period}}} = \frac{1}{3} = 0.33, \\
\tau_{2\ \text{utilization}} = 0.4, \qquad \tau_{3\ \text{utilization}} = 0.45, \\
\rho_{\text{utilization}} = \sum_{i=1}^{2} \tau_{i\ \text{utilization}} = 0.33 + 0.4 = 0.73,
\end{gathered} \tag{6}$$

$0.693 \leq \rho_{\text{utilization}} \leq 1$ but after the arrival of 3rd task $\rho_{\text{utilization}}$ becomes 1.15 which is greater than 1. Acceptance test of RM shows that any task-set is able to schedule through if $\rho_{\text{utilization}} \leq 0.693$, but not all tasks can be scheduled if $0.693 < \rho_{\text{UB}} \leq 1$ [7]. Therefore, we can say that all tasks may or may not be scheduled by RM.

In addition, in case of RM, tasks having the least priority miss the deadline due to preemption of higher priority tasks in overloading case, since RM assigns priorities statically (priority of task will not change throughout the execution process) on the basis of τ_{period}. In Figure 5 tasks priority is in order $\tau_1 > \tau_2 > \tau_3$ and also according to Table 4 the execution of third task τ_3 is doubtful. Hence, each new arrival of τ_1 after every periodic cycle preempts task τ_2 or τ_3. Similarly task τ_2 preempts τ_3. Also the arrival of 3rd task exceeds $\rho_{\text{utilization}} > 1$ and its priority is also third. As a result, τ_3 has missed its deadline. Therefore, we can say that tasks containing the east priority fail to meet the deadline in overloading situation.

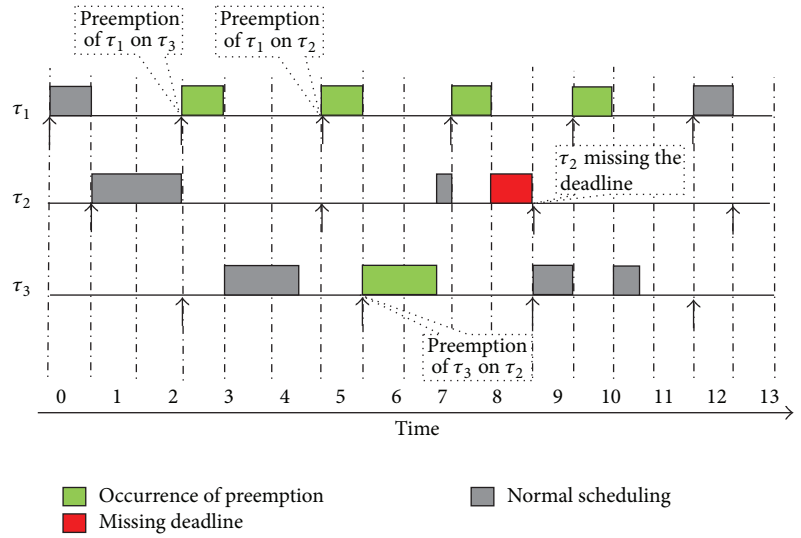

Figure 3: EDF scheduling on single processor.

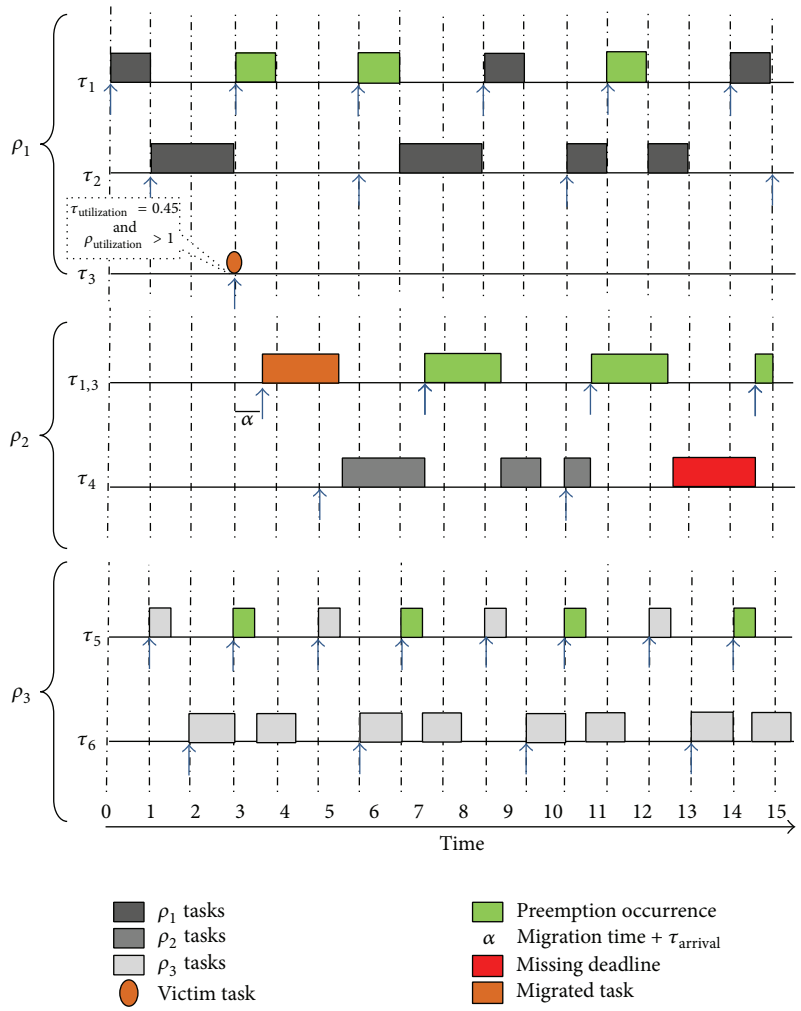

Figure 4: EDF scheduling algorithm in RTDS.

Table 4: Shows some values of ρ_{UB} on different number of tasks n.

Number of tasks (n)	Tasks	$\tau_{utilization} = \frac{\tau_{wcet}}{\tau_{period}}$	$\rho_{utilization} = \sum_{n=1}^{3} \tau_{n\,utilization}$	$\rho_{UB} = n \times (2^{1/n} - 1)$	Comparison of $\rho_{utilization}$ with ρ_{UB}	Conclusion
1	τ_1	0.33	0.33	1.0	$0.33 \leq 1.0$	τ_1 is schedulable
2	τ_2	0.4	0.43	0.8284	$0.43 \leq 0.82$	τ_2 is schedulable
3	τ_3	0.45	0.88	0.7797	$0.88 \geq 0.77$	τ_3 can be schedulable

RM scheduling algorithm in uniprocessor
BEGIN
(1) **If** $\tau_{utilization} \leq 1$
(2) **If** $\rho_{UB} \leq n\left(2^{1/n} - 1\right)$
(3) task is schedulable and
assign $\tau_{priority}$ of task on given processor
(4) **Else task is non-schedulable**
END

RM scheduling algorithm in Distributed system
BEGIN
(1) **If** $\tau_{utilization} \leq 1$
(2) **If** $\rho_{UB} \leq n\left(2^{1/n} - 1\right)$
(3) **then** task is schedulable and
assign $\tau_{priority}$ of task on given processor
(4) **else** migrate the task
(5) **Else task is non-schedulable**
END

Algorithm 2

2.3.2. Example of RM in Real Time Distributed System. As we have computed that arrival of 3rd task exceeds the range of $\rho_{utilization}$, therefore in ρ_1 task τ_3 is a victim task that initiates overloading situation in its source processor. Initially ρ_2 processor is idle and $\rho_{3\ utilization}$ is 0.75. Therefore, on the basis of utilization criteria ρ_2 becomes a destination processor for giving victim task and it will be executed on ρ_2 processor. After the arrival of migrated task $\tau_{1,3}$ in ρ_2, its utilization becomes 0.45 and its priority is 1, but after the arrival of τ_4 task $\rho_{2\ utilization}$ becomes 0.95 which is again greater than 0.693. In our given example $\tau_{period} = \tau_{dline}$; therefore RM behaves like EDF but if $\tau_{period} \neq \tau_{dline}$, then RM behaves different than EDF.

2.4. D_O_EDF and D_R_EDF Scheduling Algorithm. The main motive behind the derivation of these two scheduling algorithms is Domino's effect problem of EDF that is created only in overloading condition. We keep in our mind that we should not let the processor shoot in such a way that causes Domino's effect.

The D_O_EDF scheduling algorithm assigns static priorities 0 and 1 to jobs (Figure 6). Such static priorities will be used in overloading condition. In overloading condition, this scheduling algorithm discards those tasks that are expected to miss the deadline and their static priority is 0. On the other side, tasks with firm timing constraint and are expected to miss the deadline with static priority 1, are allowed to execute [8].

The D_R_EDF scheduling algorithm is an amalgamation of both dynamic and static scheduling algorithms, that is, EDF and RM (Figure 7). As we know, EDF is working well in underloaded situation, but its performance reduces exponentially in overloading condition. Similarly, RM behaves normally in underloaded condition but performs well in overloaded situation. Hence, according to given algorithm initially processor uses EDF for task execution, but when tasks start missing deadline due to overloading condition, the scheduler switches towards RM algorithm. As tasks continuously meet the deadline, then it means that the system is in underloaded condition now and then scheduler again switches towards EDF algorithm [8].

3. Proposed Joint EDF-RM Scheduling Algorithm

3.1. The Algorithm. This proposed algorithm is a combination of RM and EDF scheduling algorithms. As we knows in RM the upper bound of processor is computed by $n(2^{1/n} - 1)$, where n is a number of tasks. In our proposed algorithm this upper bound of RM will be a boundary limit of global task queue B_Q of global scheduler. If cumulative utilization of tasks is less than or equal to B_Q will distribute towards randomly selected processors of the system for execution, otherwise that task will be discarded.

The second scheduling algorithm EDF is working for the execution of assigned tasks on a particular processor. The positive side of using global scheduler is permitted to task migration in between processors. Figure 9 explains the task migration methodology in between processors.

3.2. Joint EDF-RM Scheduling Algorithm. Proposed Joint EDF-RM scheduling algorithm is divided into following three modules:

(1) maintenance of global task queue
(2) execution of assigned tasks on allotted processors
(3) migration of tasks in between processors if needed (if overloading alarm generates).

Following Algorithm 3 explains all above-stated three modules.

3.3. Significance of the Above-Stated Proposed Algorithm. System assumed here is loosely coupled distributed system in which all processors share identical architecture (homogeneous RTDS). Threshold limit of each processor is fixed and

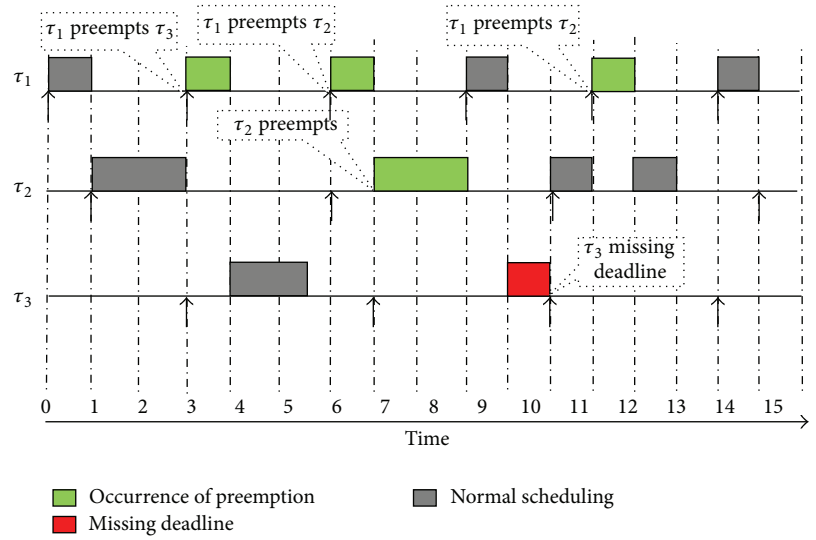

Figure 5: RM scheduling on single processor.

Input: Random arrival of tasks with $\tau_{arrival}, \tau_{wcet}, \tau_{period}, \tau_{dline}$
Output: Number of tasks meet/miss the deadline along with another parameters

BEGIN
GlobalScheduler() `// Global task Queue`
(1) `The aperiodic // Periodic arrival of tasks with arrival time, wcet, assigned deadline and period`
(2) `tasku = wcet/Period;`
(3) `UB=n*(Math.pow(2, 1.0/n)-1);`
(4) **IF** `tasku<=UB`
(5) `Generated task is schedulable`
(6) `pselection(task)`
(7) **Else**
(8) `The task is non-schedulable`

***pselection*(*task*)** `// Random Selection of Processors`
(1) `Random Selection of Processor`
(2) `PQueue(task);`

***PQueue*(*task*)** `// Processors local queue`
(1) `Assign priorities to tasks on the basis of deadline`
(2) $\text{task}_{\text{priority}} \propto \frac{1}{\text{task}_{\text{priority}}}$
(3) `TaskExecution(task)`

***TaskExecution*(*task*)** `// Task Execution by using EDF Scheduler`
(1) **If** `tasku<=1`
(2) `U= U+ tasku //Cumulative accumulation of task utilization`
(3) **If** `(U<=.810) //Processor utilization`
(4) `The task is ready for execution`
(5) **Else**
(6) `Task Migration (task, tasku)`

***Taskmigration*(*task, tasku*)** `// Task Migration on the basis of a processor utilization factor`
(1) `Sort all processor utilization`
(2) `Assign task to the processor having least a utilization factor`
(3) `PQueue(task);`
END

Algorithm 3: Joint EDF-RM scheduling algorithm.

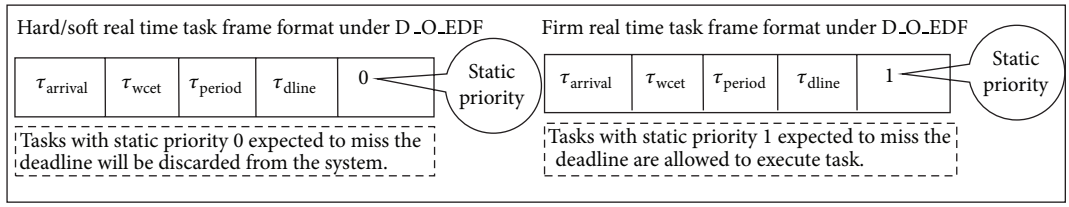

Figure 6: Real time tasks frame format according to D_O_EDF scheduling algorithm.

in order to execute the tasks, the Earliest Deadline First (EDF) scheduling is employed by every processor. In given system all tasks are independent and their $\tau_{\text{period}} \geq \tau_{\text{dline}}$. Based on the priority of task, higher priority job can preempt the lower priority task. One central scheduler (global scheduler) is used that maintains the global task queue for the entire system. As we recognize that global scheduler allows task migration in between processors, therefore for every processor we have set one utilization threshold value that generates alarms for migration of task. This entire system deals with soft as well as firm real time tasks.

In Figure 8, there is a global scheduler that maintains a waiting task queue globally. In order to avoid the problem of overloading, limited amount of tasks arrives on the queue and its boundary limit is determined by using a RM scheduling algorithm. Tasks having $\tau_{\text{utilization}} + \sum_{i=1}^{n-1} \tau_{i\ \text{utilization}} \leq i \times (2^{1/i} - 1)$ will be easily executable or if $\tau_{\text{utilization}} + \sum_{i=1}^{n-1} \tau_{i\ \text{utilization}} \leq i \times (2^{1/i} - 1) \leq 1$, then not all tasks will be executable.

As the queue behaves on the basis of first in first out, tasks are randomly assigned to processors. Each processor executes tasks by using an EDF scheduling algorithm. Small change we have done in this existing algorithm is migration threshold limit. This migration threshold limit decides the migration of task from given processor. In Figure 9 ρ_1 processor utilization is $\rho_{1\ \text{utilization}} = \tau_{i\ \text{utilization}} + \tau_{2\ \text{utilization}} + \tau_{3\ \text{utilizaion}} + \tau_{(n-2)\ \text{utilization}} = 0.61 \leq 0.810$, And after the arrival of τ_{n-1} processor utilization becomes $\rho_{1\ \text{utilization}} + \tau_{(n-1)\ \text{utilization}} = 0.70 \leq 0.810$, but arrival of τ_n reaches utilization towards $\rho_{1\ \text{utilization}} + \tau_{(n)\ \text{utilization}} \geq 0.810 < 1$. Arrival of further tasks will increase $\rho_{1\ \text{utilization}}$ by 1; after that, all tasks start missing deadline which causes a Domino effect. After migration threshold alarm, processor checks the utilization values of other processors. And then migrate the task to the processor having the least utilization. ρ_2 processor is a destination node given in Figure 9.

Theorem 1. *If the upper bound of global task queue is* $n \times (2^{1/n} - 1)$*, then overloading of processor is reduced.*

Proof. Given set of n aperiodic tasks $\tau_1, \tau_2, \tau_3, \tau_4, \ldots, \tau_n$ arrives in a global task queue, whose periods and execution times are $\tau_{1\ \text{period}}, \tau_{2\ \text{period}}, \tau_{3\ \text{period}}, \tau_{4\ \text{period}}, \ldots, \tau_{n\ \text{period}}$ and $\tau_{1\ \text{wcet}}, \tau_{2\ \text{wcet}}, \tau_{3\ \text{wcet}}, \tau_{4\ \text{wcet}}, \ldots, \tau_{n\ \text{wcet}}$, respectively. $\tau_{1\ \text{utilization}}, \tau_{2\ \text{utilization}}, \tau_{3\ \text{utilization}}, \tau_{4\ \text{utilization}}, \ldots, \tau_{n\ \text{utilization}}$ are per task utilization. We are considering here $\tau_{\text{deadline}} \leq \tau_{\text{period}}$. There are 4 processors present in our RTDS with $\rho_{1\ \text{utilization}}$, $\rho_{2\ \text{utilization}}$, $\rho_{3\ \text{utilization}}$, and $\rho_{4\ \text{utilization}}$ being their respective utilizations. Global scheduler randomly selects processors for the allocation of tasks but tasks, follow FCFS discipline for allocation.

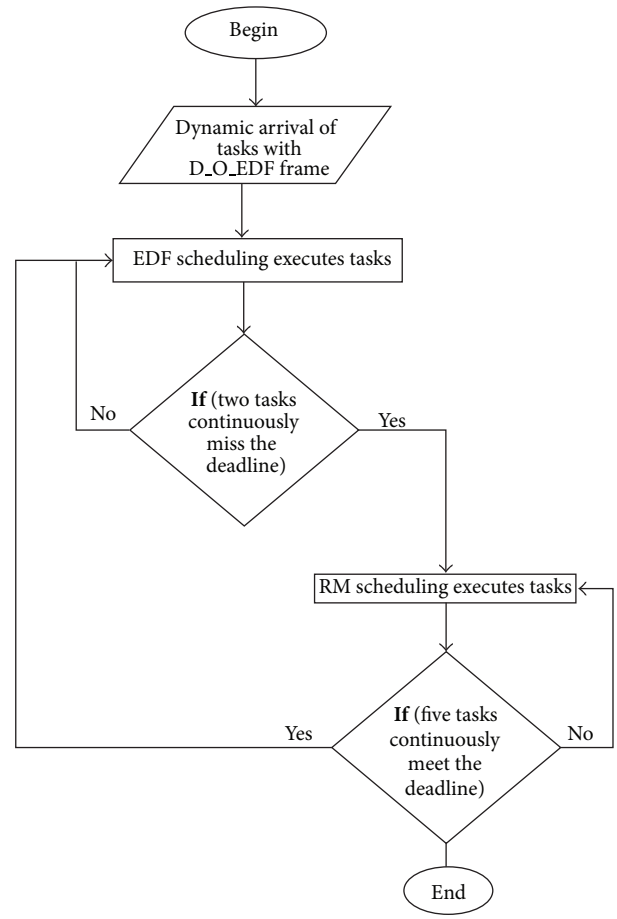

Figure 7: D_R_EDF scheduling algorithm flowchart.

We are taking three cases in order to proof given theorem

Case I. Global queue has infinite limit as the global queue is containing no acceptance test of task. Without checking its utilization, based on FCFS τ_1 task assigned to the randomly

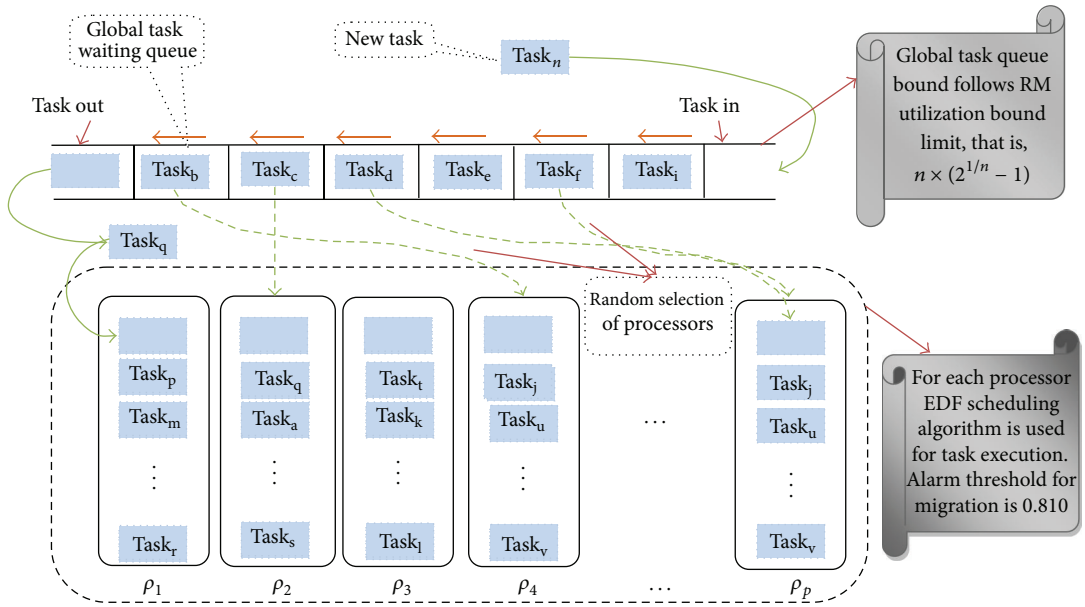

FIGURE 8: Architecture of proposed algorithm.

selected processor ρ_2 whose $\rho_{2\,\text{utilization}} < 1$, and after the assignment of τ_1, two conditions can occur:

$$\rho_{2\,\text{utilization}} = \begin{cases} \rho_{2\,\text{utilization}} + \tau_{1\,\text{utilization}}, & \text{if } \tau_{1\,\text{utilization}} \leq 1, \\ \rho_{2\,\text{utilization}}, & \text{otherwise,} \end{cases} \tag{7}$$

$$\rho_{2\,\text{utilization}} + \tau_{1\,\text{utilization}} = \begin{cases} \text{schedule } \tau_1, & \text{if } \leq 1, \\ \text{miss the deadline or overload,} & \text{otherwise,} \end{cases} \tag{8}$$

$$\text{overload} = \begin{cases} \text{migration,} & \text{if } \rho_{n\,\text{processor}} < 1, \\ \text{wait for execution,} & \text{otherwise.} \end{cases} \tag{9}$$

In this case there are more chances of overloading on every processor.

Case II. Boundary limit of global queue is 1, that is, $\sum_{i=1}^{n} \tau_{i\,\text{utilization}} \leq 1$.

Here only those tasks are allowed to enter in a global task queue whose $\tau_{\text{utilization}} \leq 1$.

If tasks in a queue are waiting for an assignment and the arrival of new task increases the boundary limit by 1, then all upcoming tasks will not be allowed for admission:

$$B_Q = \begin{cases} \text{allowed for execution,} & \text{if } \leq 1, \\ \text{Queue is full,} & \text{otherwise.} \end{cases} \tag{10}$$

In this case (7) is satisfied only on 1st condition; that is,

$$\rho_{\text{utilization}} = \rho_{\text{utilization}} + \tau_{\text{utilization}}. \tag{11}$$

This case gives a guarantee of schedulability of every task. Equations (8) and (9) behave similar to 1st case. The limitation of the EDF scheduling algorithm is that if one task starts missing deadline, then upcoming tasks also miss deadline continuously (Domino's effect).

Case III. Global queue has boundary limit $n \times (2^{1/n} - 1)$.

Tasks having $\sum_{i=1}^{n} \tau_{n\,\text{utilization}} \leq n \times (2^{1/n} - 1)$ are allowed to execute on assigned processors. As we know the value of

$$n \times \left(2^{1/n} - 1\right) = \begin{cases} 1, & \text{for } n = 1, \\ < 1, & \text{for } n \text{ tends } \infty. \end{cases} \tag{12}$$

According to RM scheduling, every task is schedulable if its $\tau_{\text{utilization}} \leq n \times (2^{1/n} - 1)$, but its execution is doubtful if it is in between $n \times (2^{1/n} - 1)$ and 1. Therefore, here queue allows only those tasks for further execution whose $\tau_{\text{utilization}} \leq n \times (2^{1/n} - 1)$. Consider

$$\sum_{i=1}^{n} \tau_{i\,\text{utilization}} \leq n \times \left(2^{1/n} - 1\right) \quad \forall n,$$

$$B_Q = \begin{cases} \text{allowed for execution,} & \text{if} \leq n \times \left(2^{1/n} - 1\right), \\ \text{tasks are non-schedulable,} & \text{otherwise.} \end{cases} \tag{13}$$

After the allocation of tasks on the processor, it will execute tasks by using EDF:

$$\rho_{\text{utilization}} = \begin{cases} \text{schedule tasks,} & \text{if } \leq 1, \\ \text{overloading occur,} & \text{if } > 1. \end{cases} \tag{14}$$

But in 3rd case very rare tasks utilization reaches 1 but not beyond 1. Hence, we can say that if the upper bound of global task queue is $n \times (2^{1/n} - 1)$, then overloading of processor is reduced. □

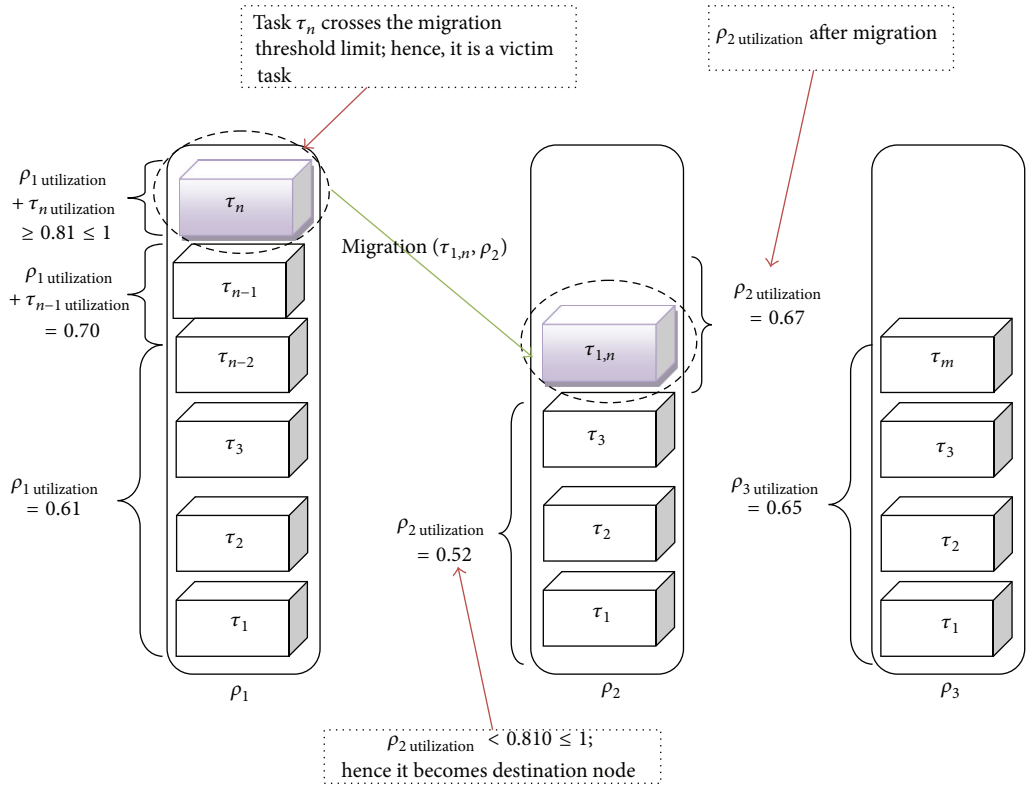

Figure 9: Migration scenario in proposed algorithm.

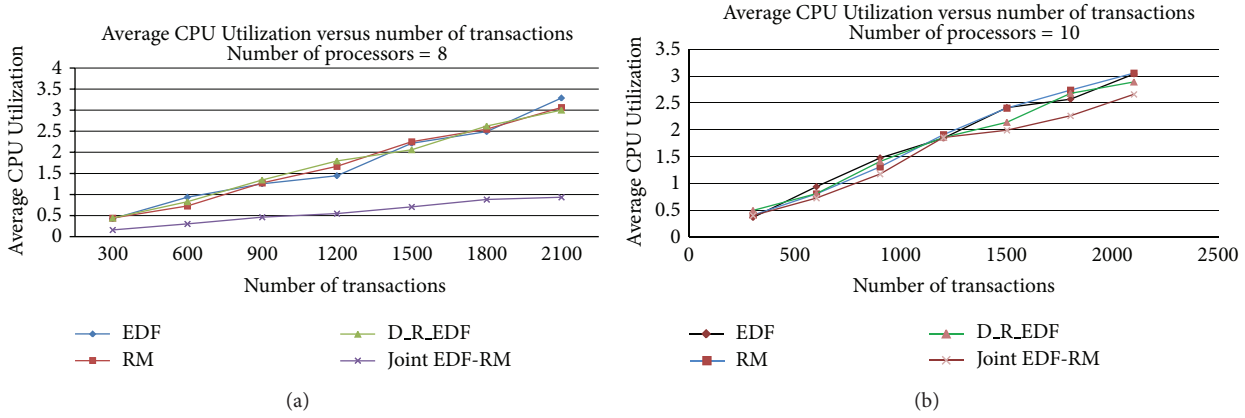

Figure 10: Average CPU Utilization versus number of transactions on 8 and 10 processors.

4. Performance Analysis and Simulation Results

In order to evaluate our proposed scheduling algorithm we have used Eclipse Java EE IDE. The operation of the proposed study is measured by calculating the Average CPU Utilization, Success Ratio, Failure Ratio, and Maximum Tardiness. The simulation is done with more than 26000 transactions on 10 processors of RTDS, but in simulation results we have mentioned transactions up to 2100. Before making the demonstration of calculating simulation results, let us discuss those parameters that determine the performance of joint EDF-RM scheduling algorithm with some existing algorithms (EDF, RM, D_O_EDF, and D_R_EDF).

4.1. Average CPU Utilization ($A_{\rho\ \text{utilization}}$). It is defined as

$$A_{\rho\ \text{utilization}} = \sum_{i=1}^{n} \frac{\rho_{i\ \text{utilization}}}{n}, \tag{15}$$

where the number of processors (ρ) varies from 1 to n.

In Figure 10, the CPU utilization in proposed joint EDF-RM scheduling algorithm is lesser than the other three algorithms. As (1) explains, processor utilization is dependent

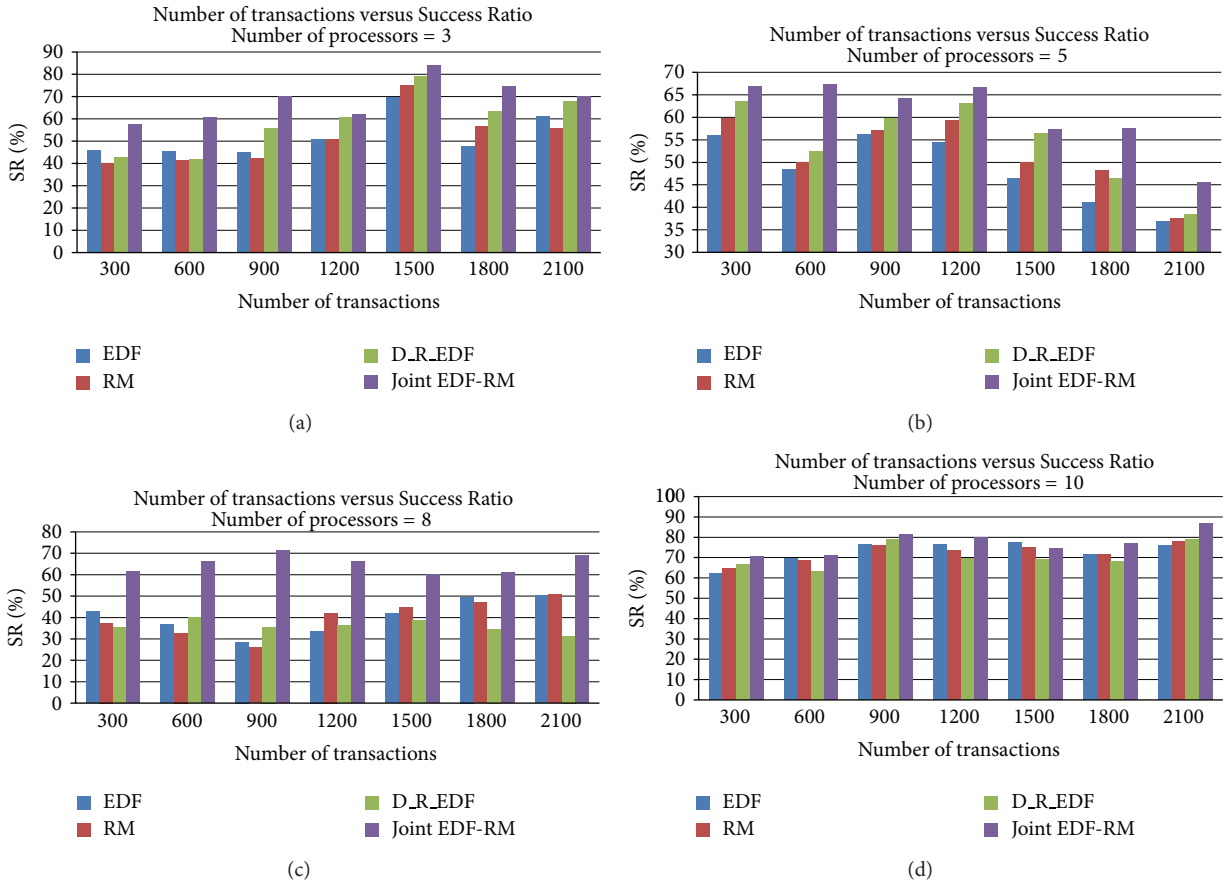

FIGURE 11: Number of transactions versus Success Ratio on 3, 5, 8, and 10 Processors.

on particular tasks utilization and further tasks utilization is dependent on the worst case execution time (τ_{wcet}) and interarrival period (τ_{period}) of tasks as well. Additionally, the τ_{wcet} and τ_{period} are generated randomly in given simulation. Due to all these factors of real time tasks the Average CPU Utilization can vary from time to time. For example, in Figure 10 $A_{\rho\,\text{utilization}}$ of 8 processors for 300 tasks is 0.16338866 and on the other side for 10 processors it is 0.4004137. Similarly, In EDF $A_{\rho\,\text{utilization}}$ of 8 processors for 2100 tasks is 3.287974 whereas for 10 processors it is 3.0410550. Hence, the value of Average CPU Utilization is uncertain, but as compared to previous existing algorithm, proposed joint EDF-RM algorithm gives better $A_{\rho\,\text{utilization}}$ because of migration threshold limit of all processors along with the boundary limit of global task queue.

4.2. Success Ratio (SR). Consider

$$\text{SR} = \frac{\text{Successfully scheduled tasks}}{\text{Total number of tasks arrival}}. \tag{16}$$

Meeting with the deadline is very imperative for all real time tasks; therefore we have computed success ratio that tells the percentage of successful implementation of tasks from total transactions.

After simulating thousands of tasks, we find that EDF, RM and D_R_EDF scheduling algorithms' Success ratio can vary however we sometimes find that our proposed algorithm has a higher Success Ratio than the existing algorithms (Figure 11). Reason behind its best performance is a threshold value of task migration.

4.3. Failure Ratio (FR). Consider

$$\text{FR} = \frac{\text{Tasks miss the deadline}}{\text{Total number of tasks arrival}}. \tag{17}$$

This parameter computes the other side of coin, that is, percentage of those tasks which are unable to meet the deadline. Missing deadline is also a big task in front of all algorithms (Figure 12). Therefore, we also compute the Failure Ratio that tells us the occurrence of missing deadline.

4.4. Maximum Tardiness. As we know, tardiness is the lateness occurring in tasks execution; that is,

$$\begin{aligned} \text{tardiness} &= \text{TST} - \tau_{\text{deadline}}, \\ \text{Max Tardiness} &= \max\left(\text{tard}_{\tau_i}\right), \quad \text{where } i \in T. \end{aligned} \tag{18}$$

While missing a deadline, we have computed the time after which task successfully executes. Figure 13 explains that the proposed algorithm has Minimum Tardiness compared to existing algorithms.

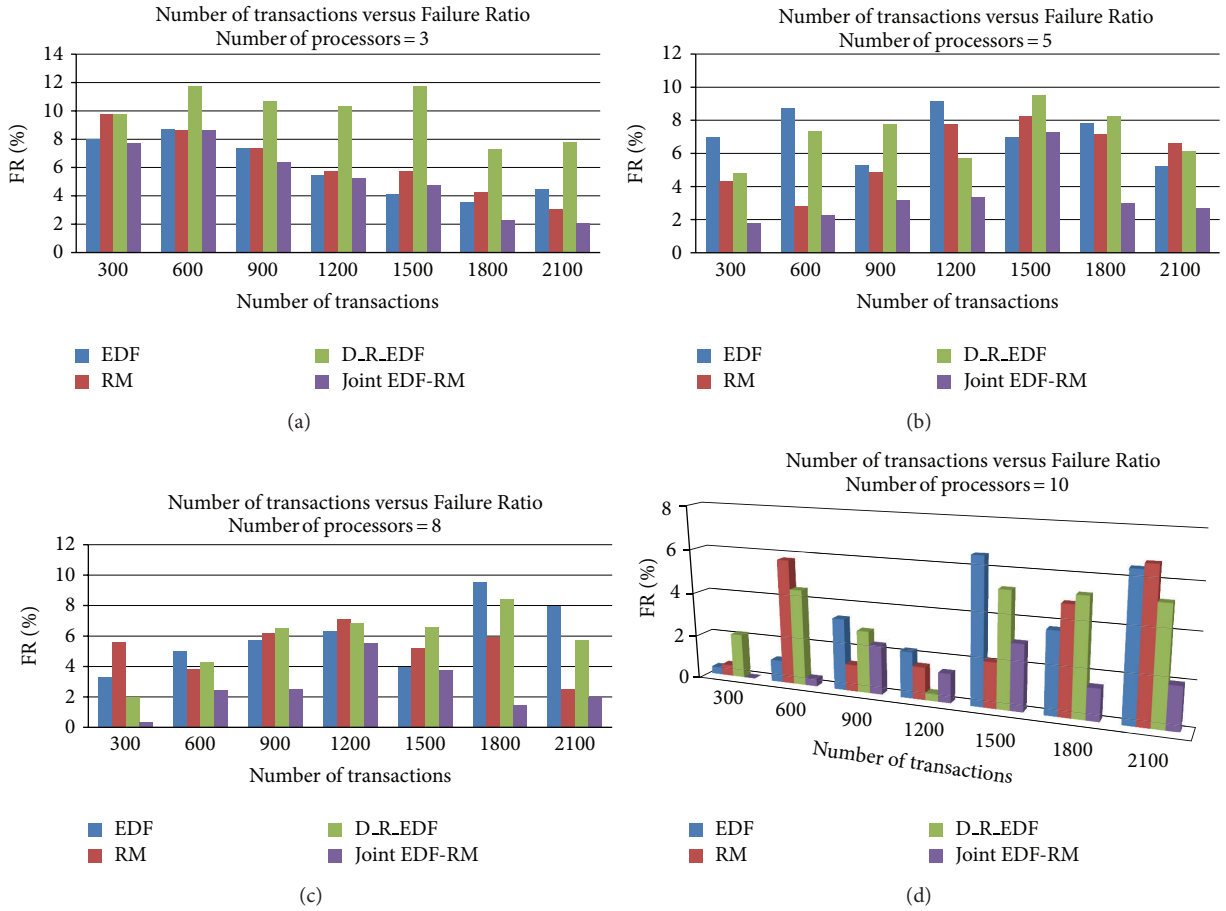

FIGURE 12: Number of transactions versus Failure Ratio on 3, 5, 8 and 10 processors.

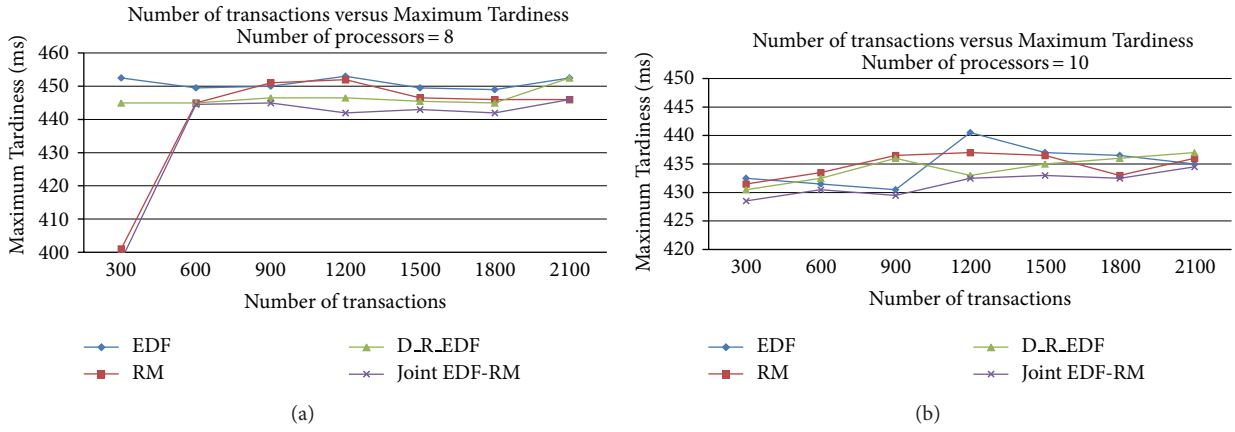

FIGURE 13: Number of transactions versus Maximum Tardiness on 8 and 10 processors.

5. Conclusion and Future Work

Proposed scheduling algorithm is the combination of EDF and RM scheduling algorithms that rise above the overloading problem of any processor. We have set the threshold limit 0.81 that generates alarm for the migration of upcoming tasks because overloading on task is restricted. We simulate this work for homogeneous system; in the future we will implement it on heterogeneous arrangement. One main problem occurs when running tasks are preempted by higher priority new tasks because running tasks miss the deadline. Hence, in future we will work on preemption technique of scheduling algorithms with fault tolerant techniques.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] J. W. S. Liu, *Real-Time Systems*, Pearson Education, New Delhi, India, 2003.

[2] P. Li and B. Ravindran, "Fast, best-effort real-time scheduling algorithms," *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1159–1175, 2004.

[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[4] J. A. Stankovic and K. Ramamritham, *Tutorial Hard Real-Time Systems*, IEEE Computer Society Press, 1988.

[5] M. Joseph, *Real-Time Systems: Specification, Verification and Analysis*, Prentice Hall, New York, NY, USA, 1996.

[6] P. A. Laplante, *Real-Time Systems Design and Analysis: An Engineer Handbook*, IEEE Computer Society, IEEE Press, 1993.

[7] G. C. Buttazzo, "Rate monotonic versus EDF: judgment day," *Real-Time Systems*, vol. 29, no. 1, pp. 5–26, 2005.

[8] A. Shah and K. Kotecha, "Efficient scheduling algorithms for real-time distributed systems," in *Proceedings of the 1st International Conference on Parallel, Distributed and Grid Computing (PDGC '10)*, pp. 44–48, Solan, India, October 2010.

[9] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, Addison-Wesley, New York, NY, USA, 5th edition, 2011.

[10] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Bader, "Dynamic load balancing in distributed systems in the presence of delays: a regeneration-theory approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 485–497, 2007.

[11] R. Sharma and Nitin, "Optimal method for migration of tasks with duplication," in *Proceedings of the 14th International Conference on Modelling and Simulation*, pp. 510–515, 2012.

[12] T. T. Y. Suen and J. S. K. Wong, "Efficient task migration algorithm for distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 4, pp. 488–499, 1992.

[13] R. Sharma and Nitin, "Task migration with EDF-RM scheduling algorithms in distributed systems," in *Proceedings of the Internation Conference on Advances in Computing and Communications*, pp. 182–185, 2012.

[14] S. Darbha and D. P. Agrawal, "A task duplication based scalable scheduling algorithm for distributed memory systems," *Journal of Parallel and Distributed Computing*, vol. 46, no. 1, pp. 15–27, 1997.

[15] R. Sharma and Nitin, "Duplication with task assignment in mesh distributed system," in *Proceedings of the World Congress on Information and Communication Technologies (WICT '11)*, pp. 672–676, Mumbai, India, December 2011.

[16] N. Fisher, S. Baruah, and T. P. Baker, "The partitioned scheduling of sporadic tasks according to static-priorities," in *Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS '06)*, pp. 118–127, Dresden, Germany, July 2006.

[17] T. P. Baker, "A comparison of global and partitioned EDF schedulability tests for multiprocessor," TR-051101, 2005.

[18] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 4, pp. 553–566, 2009.

[19] L. Kleinrock, *Queueing Systems—Volume 2: Computer Applications*, Wiley-Interscience, New York, NY, USA, 1976.

[20] A. Gantman, P. N. Guo, J. Lewis, and F. Rashid, "Scheduling real-time tasks in distributed systems: a survey," 1998.

[21] O. U. P. Zapata and P. M. Alvarez, "Edf and RM multiprocessor scheduling algorithms: survey and performance evaluation," Seccion de Computacion Av. IPN, 2508, 2005.

[22] K. Ramamritham, J. A. Stankovic, and P. F. Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 184–194, 1990.

[23] J. H. Anderson, V. Bud, and U. C. Devi, "An EDF-based restricted-migration scheduling algorithm for multiprocessor soft real-time systems," in *Proceedings of the 17th Euromicro Conference on Real-Time Systems (ECRTS '05)*, pp. 199–208, July 2005.

[24] J. Lehoczky, L. Sha, and Y. Ding, "Rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Proceedings of the Real-Time Systems Symposium*, pp. 166–171, December 1989.

[25] V. N. Darera and L. Jenkins, "Utilization bounds for RM scheduling on uniform multiprocessors," in *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '06)*, pp. 315–321, Sydney, Australia, August 2006.