International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 4, August 2016, pp. 1489~1498
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i4.9781

Performance Analysis of Preemptive Based Uniprocessor Scheduling

M Shanmugasundaram[1], R. Kumar[2], Harish M Kittur[1]
[1]School of Electronics Engineering, VIT University, India
[2]Wipro Technologies, Chennai, India

Article Info

Article history:

Received Dec 25, 2015
Revised Mar 11, 2016
Accepted Mar 28, 2016

Keyword:

EDF
Real Time System
RM
Task
Uniprocessor Scheduling

ABSTRACT

All the real-time systems are bound with response time constraints, or else, there is a risk of severe consequences, which includes failure. The System will fail when not able to meet the requirements according to the specifications. The problem of real-time scheduling is very vast, ranging from uni-processor to complicated-multiprocessor. In this paper, we have compared the performance of real-time tasks that should be scheduled properly, to get optimum performance. Analysis methodology and the concept of optimization leads to the design of appropriate scheduling. We have done the analysis among RM and EDF algorithm that are important for scheduling in uni-processor.

Corresponding Author:

M Shanmugasundaram,
School of Electronics Engineering,
VIT University,
Old Madras Road, Vellore, 632014, Tamilnadu, India.
Email: phdsundaram@gmail.com

1. INTRODUCTION

The motivation behind a continuous framework is to have a physical impact in a picked time-limit. A continuous framework is made up of a controlled and controlling framework. The computer speaks with its surroundings on the premise of data accessible about the surroundings. A continuous system that controls a gadget / methodology/ sensors, gives the readings at intermittent interim and the computer reacts to the actuators with the help of signals. There may be unknowing occasions and they should likewise get a reaction [1].

In all cases, there is a periodic bound in which the reaction ought to be conveyed. The potential of the system is to meet those bound requests relies on upon its capability to perform the fundamental calculations inside the limited time. In the event that various occasions all the while happening inevitably, the computer will need to timetable the processing so that every reaction is recorded in the obliged time limits. It may happen that, the framework is not able to satisfy all the conceivable sudden requests. In the critical circumstances, the framework needs sufficient assets and fit for handling at unbounded pace can fulfill any such time limitation. Inability to meet these requirements for a reaction can come about into diverse results. There may be no impact by any stretch of the imagination or the impact may be little or correctable. On the other hand, the outcomes may be disastrous. Every assignment happening in a continuous framework has some timing properties. These properties ought to be considered when facing assignments on a continuous framework [2],[3].

- *Release time:* Time needed for handling errand.
- *Deadline:* maximum allowable time to complete the given assignment.
- *Minimum deferral:* Minimum obliged interval before starting the execution of the assignment.

- *Maximum deferral:* Maximum measure of permeable time before starting the execution of the errand is begun, after discharging the assignment.
- *Worst-case execution time:* Maximum time taken to complete the errand, after the assignment is discharged.
- *Run time:* Time taken to finish the undertaking without a break, after the assignment is discharged.
- *Weight (or need):* Relative critics of the task.

The target of a computer based controller may be to summon the robots by moving the parts from machines to transport in some obliged design without impacting different articles. In the event that the processor controlling a robot does not summon it to do the desired work in time, the robot may impact an alternate question on the industrial facility floor. A constant framework will normally need to meet numerous requests inside a bound time. The essentials of the requests may differ in nature (example a security based interest might be more critical than basic information processing request). So the portion of the framework assets requirements to be arranged with the goal that all requests are met within the due dates.

The planning is carried out to use a scheduler which executes a booking arrangement that characterizes how the assets of the framework are allotted to the project. Planning arrangements are uncovered numerically so this exactness of the formal particular and system advancement can be supplemented by a scientific timing investigation of the system properly.

2. DISCUSSION OF DIFFERENT METHODOLOGIES

In real-time, correctness of the system also depends on the time of delivery apart from logical solution. If the system is not handled in the best way with time limit, then system failure is surely going to happen. Therefore, it is very essential to assure the time constraints of the system. Predictability is defined as the possibility of determining the completion time of activated job. While meeting the time constraints the system will meet the maximum utilization [4]. It is essential that the controlling-system should be reliable with the real-time constraints. Otherwise, this system's response may be terrible. Hence, monitoring the status in the regular basis and processing the information in and timely manner are necessary [5]. In real-time, the application is a combination of a variety of jobs with crucial at different levels. We can categorize them as shown in Figure 1. Soft real-time tasks are that in which the system will work even though the missing of some deadlines. But, sometimes it may lead to pay consequences [6]. In hard real-time, missing of task's deadline leads to catastrophic effect. There is one more set of tasks, called firm real-time tasks, which are those will finish their executions before the deadlines [7].

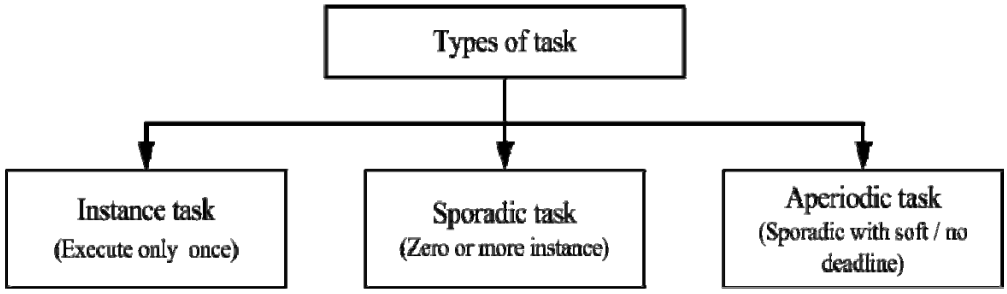


Figure 1. Types of Tasks

2.1. Uniprocessor Real-time Scheduling

It decides the time and the environment to execute tasks such that time requirements are met and performance is optimized. Scheduling-analysis defines the study of the properties of scheduling policies [8]. Feasible Schedule defines that each and every task must be executed within the deadline without violating the constraints, whereas optimal schedule tells about optimal criteria related to the feasible schedules.

Real-time systems give the preference for preemptive based priority scheduling, which can be further categorized as shown in Figure 2.

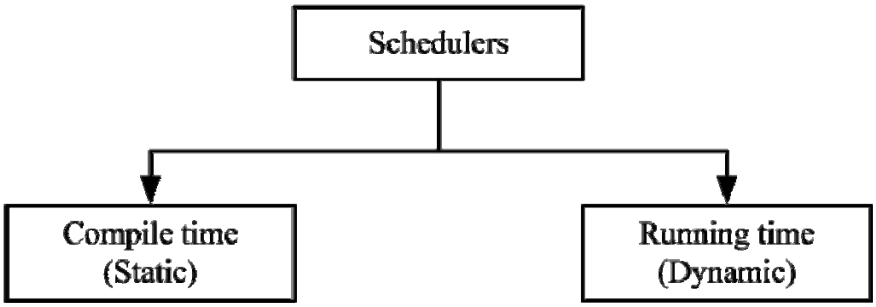


Figure 2. Types of Schedulers

Two most important priorities driven preemption scheduling techniques are:

- Rate Monotonic Scheduling (RMS - Static)
- Earliest Deadline First (EDF - Dynamic)

Scheduling Test: - A scheduling test is a method to test the particular application, whether it is completed within the given deadline when the jobs or tasks are scheduled as per specific algorithm. Schedulability utilization is the maximum allowable usage of resources by the set of tasks to guarantee the optimality.

Optimal Scheduler: - A scheduler is referred as optimal, if suppose it can able to increase the average response time or to reduce the average waiting time [9].

Parameters related to the task are given as follows:

- r → ready time
- c → maximum computational time
- d → relative deadline
- D→ absolute deadline

If the task T has more than one instance, we have 'T' period (for periodic tasks). Additional constraints are periodic request of service of the task, dependency between the tasks, resource sharing and preemption details. A task can be represented with the parameters C and P. Where C is the worst case execution time/compile time such that $(C \leq T)$ .Also T is the period of the task. A task set can be represented as(C_i, T_i).

2.1.1. A Simple Model

Consider a simple real-time application consists of a priority based periodic hard real-time tasks which do not need any extra resources.

We define a simple code in real-time as, H receives the signal from S (sensor) periodically in the inter-arrival time T. The processing of an event is defined as a task. Let us consider C as the worst-case computation time. The task should be completed by D unit time. Assume the effective deadline bounded by T such that $C \leq D \leq T$ [7]. Consider the application which receives the signals from two sensors. SensorA send the signal at every P_1 time unit and each of them need C_1 units of computation time and SensorB send at every T_{S1} time units and need C_2 units. Let us assume absolute deadline (D) = period (T) such that it is T_{S1} units of time for SensorA and Ts_2 units of time for SensorB. Now the question is: when the code periodically collect the data from sensors, how can it determines the deadline of each device? Before doing the analysis of the above problem, first we must know the assumptions. Let us assume that code is the combination of periodic independent tasks. Assume the system has one processor which periodically receives unbuffered events from the external environment [10].

2.1.2. Scheduling Strategy

Let us consider the set of tasks to be $T_1, T_2, T_3, \ldots\ldots, T_n$ one way of the scheduling is, by analyzing the set of task statically and determines their time limitations. This can be used for creating a fixed scheduling table by which the tasks will be dispatched for execution during run-time. Hence, the order of execution will be fixed.

In the scheduling analysis, a positive number is assigned as priority. By convention, the lowest numeric value has the highest importance. Consider the following two simple priority scheduling disciplines as shown Figure 3.

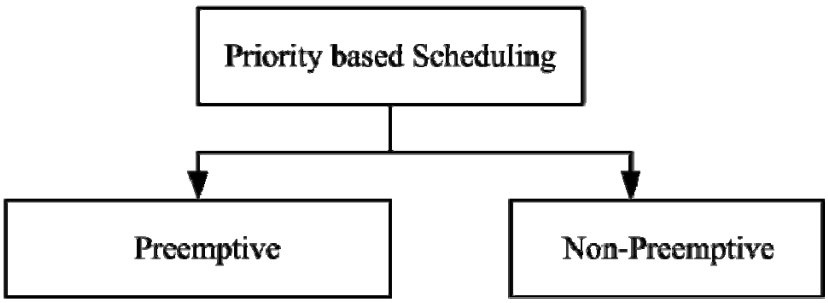


Figure 3. Classification of priority based scheduling

2.2. Rate Monotonic Scheduling

It is the method; the priorities are assigned to the set of set of tasks in a task-set as a monotonic function of the periodic task rate. Rate monotonic scheduling gives the inequality between total utilization of processor and a theoretically calculated bound which is the sufficient condition that ensures completion of set of tasks in a taskset within the end of their periods [11],[12].

$$\sum \frac{C_j}{T_j} \leq n\left(2^{\frac{1}{n}} - 1\right) \quad (1)$$

Where C_j is the execution time, T_j is the interarrival time of task.

Let us assume the model of task as given below

1. Periodic taskset: (C_j ,T_j)
2. $D_j = T_j$
3. Release time = Start of Period.
4. Task is independent.

Rate Static priority scheduling

- RM static priority are those which are higher priorities given to tasks with small periods.
- Run-time Scheduling are those preemptive with highest priority first
- RMS is optimal, if the set of tasks is schedulable with any static scheduling algorithm, it is schedulable with RMS.

RMS: Schedulability Test:

- $U < 1$ does not mean schedulable with RMS.
- Utilization bound: for a given task-set τ_S, find $UB(\tau_S)$ such that $U \leq UB(\tau_S)$ if and only if τ_S is schedulable by RMS (necessary and sufficient test).The bound $UB(\tau_S)$ in EDF is 1.

Assume a set of n independent tasks: $\tau_S = \{(1, C_1), (T_2, C_2) \ldots . (T_n, C_n)\}$ *And* $U = 1$

- If $U \leq n\left(2^{\frac{1}{n}} - 1\right)$, then τ_S is schedulable by RMS.
- Here the bound depends completely on the size of the task set

RMS Algorithm

```
Procedure feasible RTI (task-set)
    int x ;
    y= size(task-set,1);
    n_Feasible = 1;
    int  T = 0, T_old = 0;
    for all i=1:n do
        T = T+ task-set( i, 1);
        while (T > T_old)
                T_old = T;
                T = task-set(i,1);
                for all j=1:i-1 do
                        T = T + ceil( R_old / task-ser(j,2)) × task-set(j,1);
                end_for
                if (T> task-set(i,2)) then
                        n_Feasible = 0;
```

```
                    break;
                end_if
                end_for
                if (n_Feasible == 0) then
                    break;
                end_if
            en_while
        end_function
```

Consider the set of tasks as shown in Table 1.

Table 1. Example Taskset

Task	Ci	Di	Ti
τ_1	2	2	3
τ_2	2	3	5
τ_3	2	4	9

Figure 4 shows the RM schedulability the given tasks.

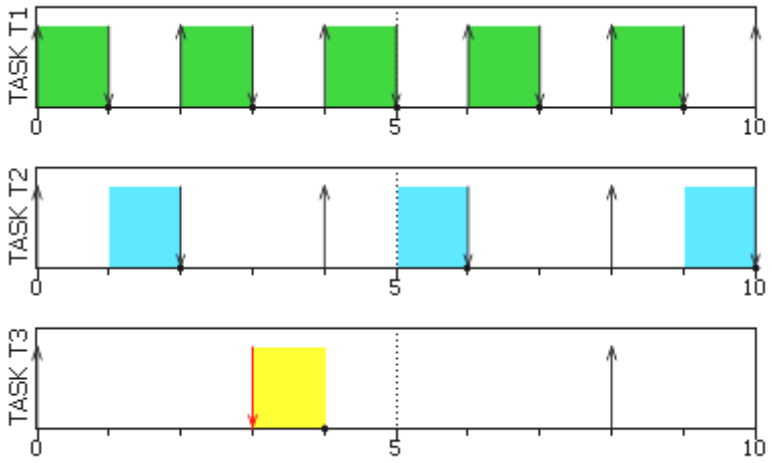


Figure 4. RM Scheduling

2.3. Earliest Deadline First Algorithm

EDF algorithm is a method that can be used to add the time redundancy to tolerate the transient faults [13]. For real-time processes time redundancy should be maintained to meet the fault tolerance. Mostly Transient faults will be handled by EDF [14]. If the transient fault occurs, then the task is repeated in intervals when the fault occurs until the system works correctly [15]. In general, the occurrence of transient faults is more frequent when compared to other faults [16].

In EDF the highest priority is assigned to the task which meets deadline first in the presence of faults. Hence, this is called as Earliest Deadline First scheduling. If the task, which is running does not have the minimum deadline then that task is stopped and the higher priority task is placed and executed. EDF does not manage the time redundancy. It does not guarantee that in presence of faults all the tasks will be completed within the specified time. It is used to just add the time redundancy in order to make the system fault tolerant [17].

2.4. Task Model

Assume the tasks to be periodic, independent and preemptive. Consider a set of n tasks $\{\tau_1, \tau_2, \tau_3 \ldots \tau_n\}$ such that $\tau_j = (C_j\ R_j,\ D_j,\ T_j)$ where $j=1, 2, \ldots..n$ and C_j R_j, D_j, T_j corresponds to time to compute, time to release, deadline and task period.

Task utilization is $U_j = {}^{C_j}/_{T_j}$

Total utilization of the system is $U = \sum_{j=1}^{n} U_j$

As the faults are transient it affects only one task at a time which can be corrected by re-execution. The detection of these faults can be done in different ways. One of such method is acceptance test.

2.5. Schedulability test

$$U = \sum \frac{C_j}{T_j} \leq 1 \quad (2)$$

Consider the taskset as given in Table 4. While doing the scheduling analysis for the set of given tasks under EDF Scheduling, we have got the scheduling stratergy as shown in Figure 5.

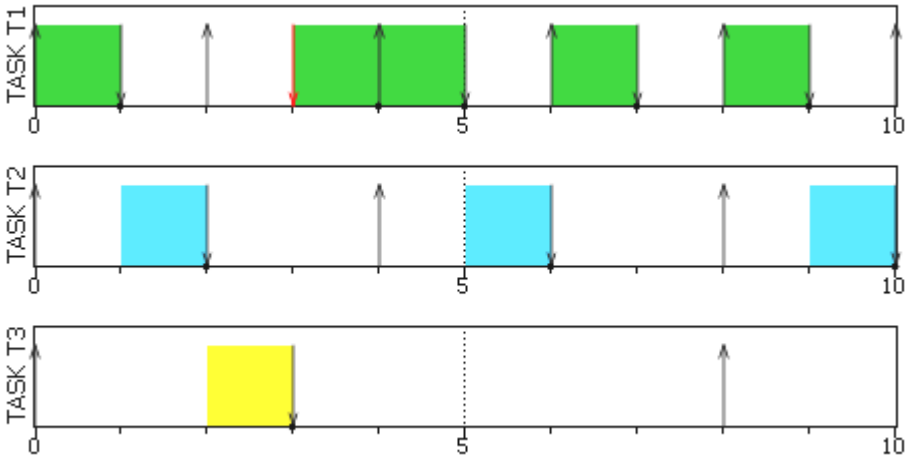


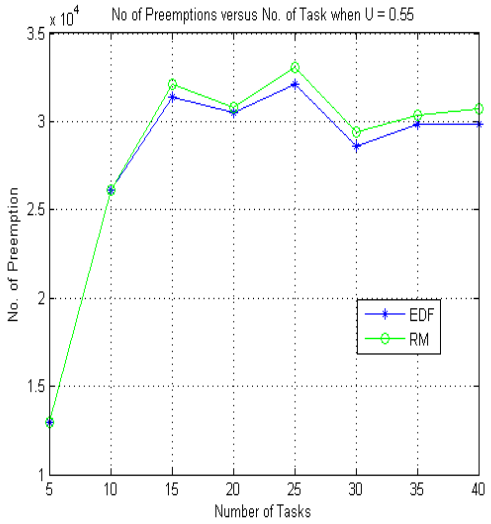
Figure 5. EDF Scheduling

2.6. Comparative Analysis of RM and EDF

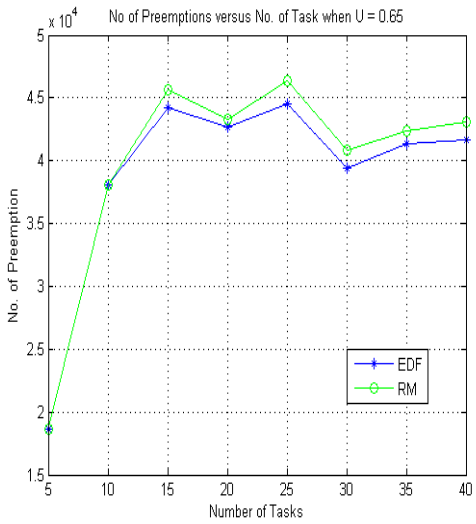
For the analysis, we have assumed preemptable periodic tasks with maximum computation time C_j, period T_j and relative deadline D_j ($D_j \leq T_j$). Generated the taskset with different combination of tasks under uniform distribution [18].

We have conducted two sets of experiments. In the first set, we have analyzed the occurrence of number of preemptions, while scheduling the set of tasks by RM and EDF Scheduling algorithms. To make it fair, we have considered only preemptable feasible task with zero preemption cost.

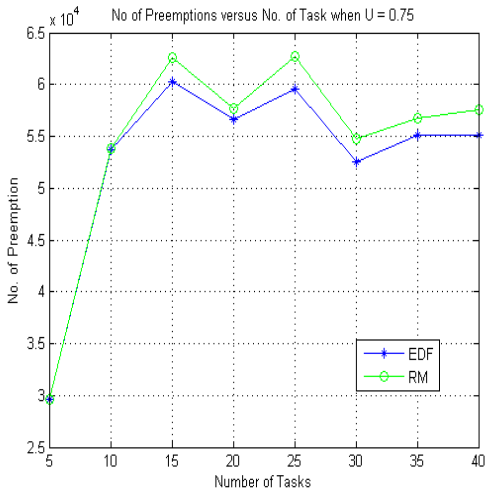
Figure 6 shows the performance of RM and EDF scheduling in terms of preemption as the function of number of tasks by keeping total utilization as constant.



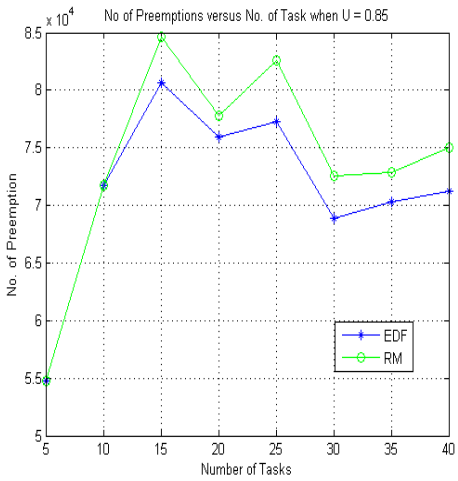
(a)



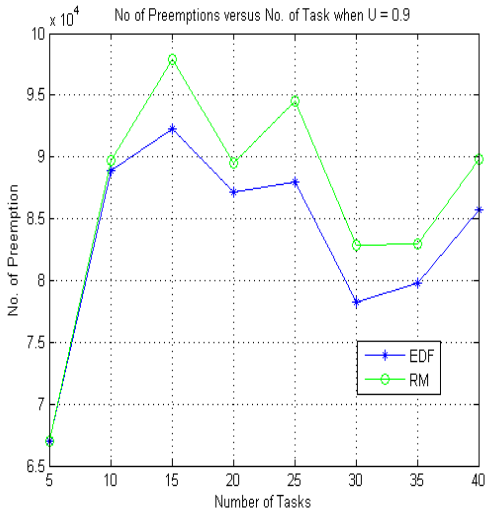
(b)



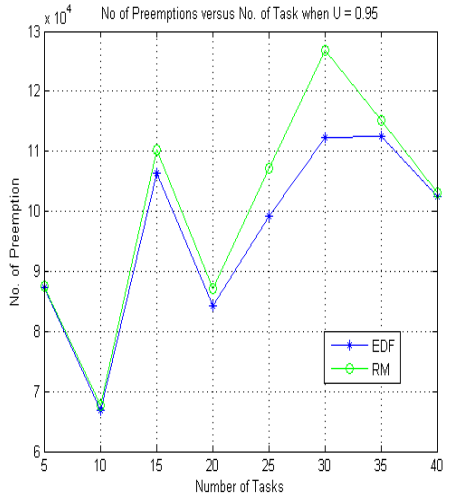
(c)



(d)



(e)



(f)

Figure 6. Preemption vs. No. or tasks when U = Constant (55, 65, 75, 85, 90 and 95)

Figure 7 shows the No. of preemptions as the function of total utilization for constant number of tasks. From the graph, we have found that the no. of preemptions decreases when there is an increase in number of tasks and total utilization in both the algorithms.

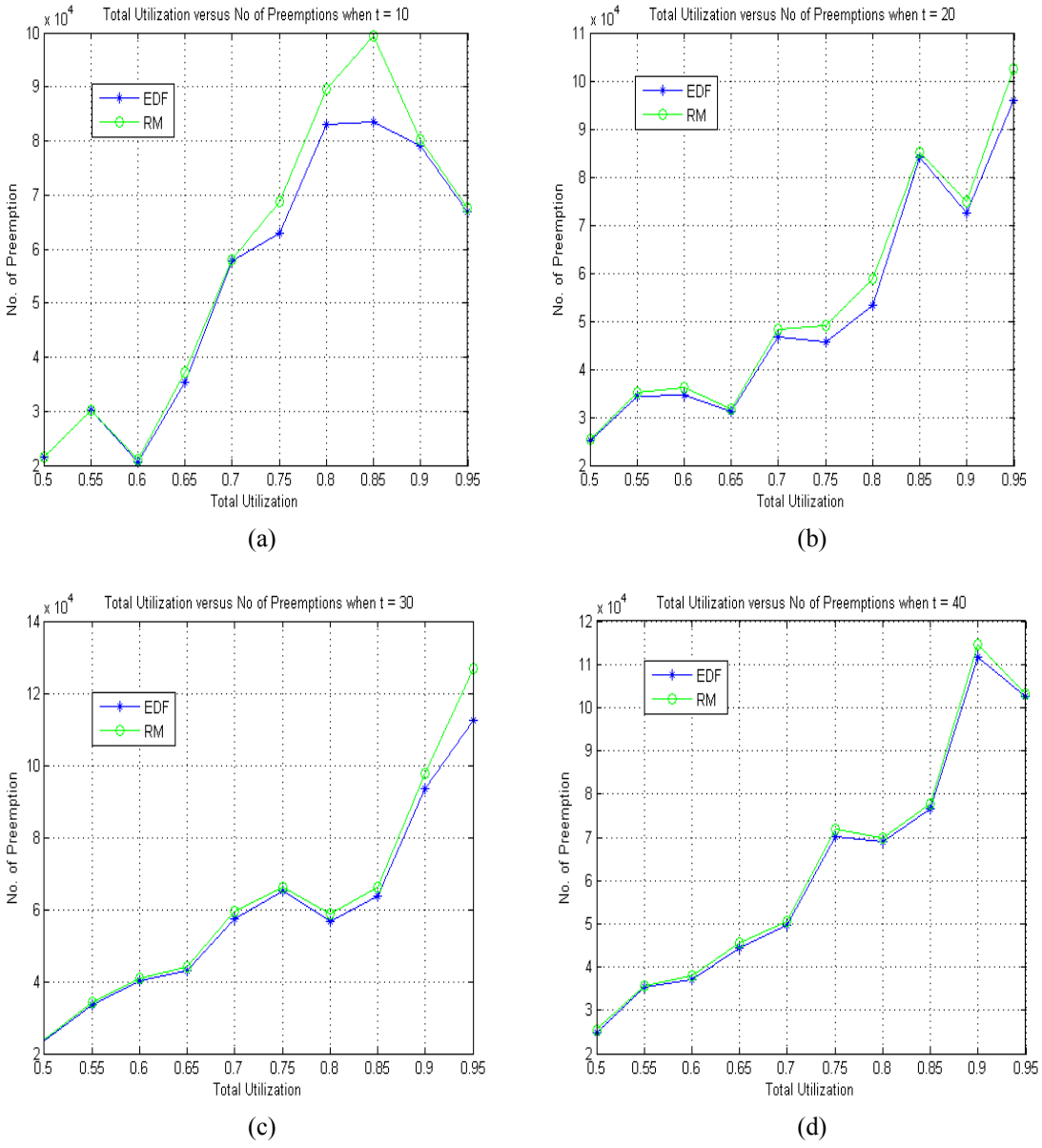


Figure 7. No. of Preemption vs. Total Utilization for T = Constant (10, 20, 30 and 40)

We have carried out the second experiment to find the effect of total utilization on the schedulability of the given algorithms by keeping the constant number of tasks as shown in Figure 8. Figure 9 shows the performance of scheduling in terms of feasibility ratio vs. no. of tasks for the constant utilization. Here, we have found the decrease in feasibility ratio along with increase in number of tasks and total utilization.

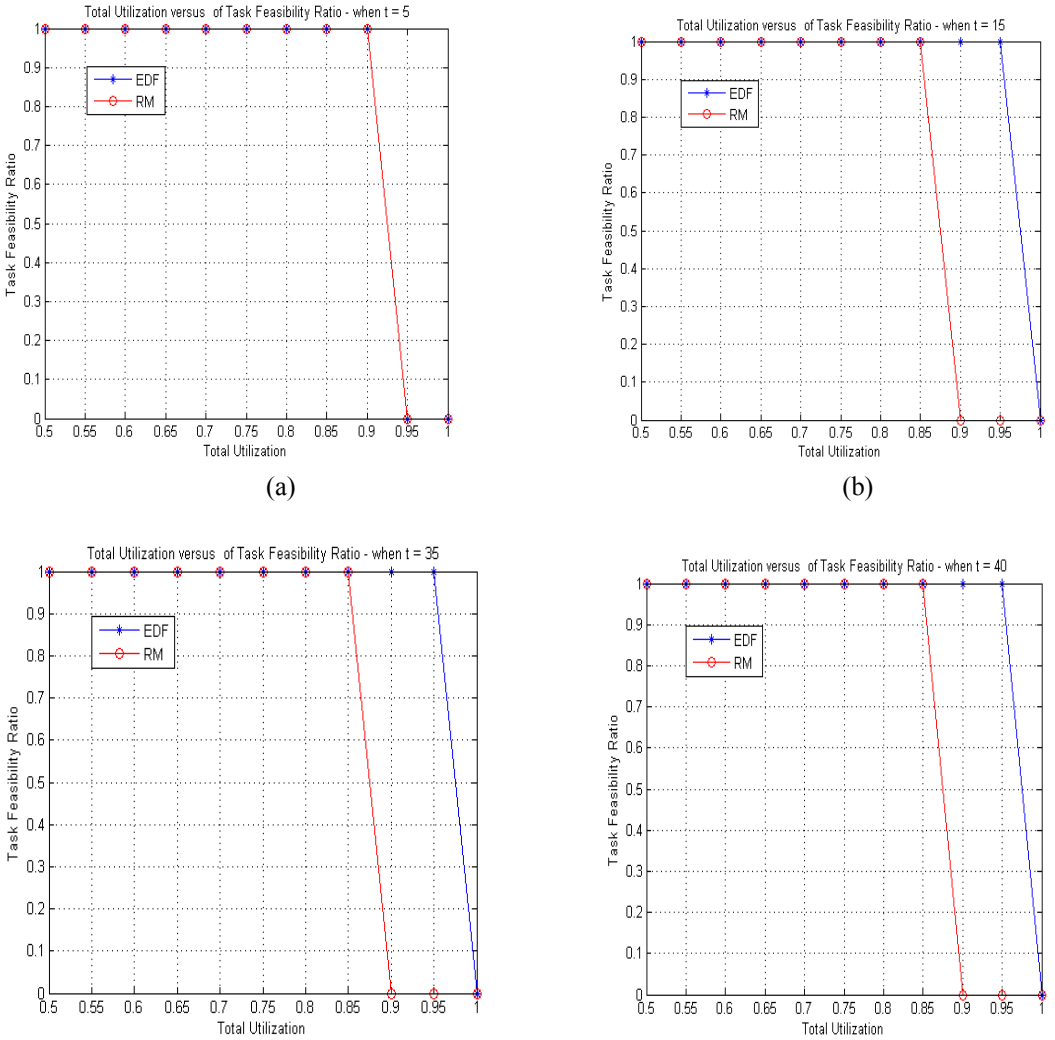


Figure 8. Task Feasibility Ratio vs. Total Utilization for τ = Constant (5, 15, 35, 40)

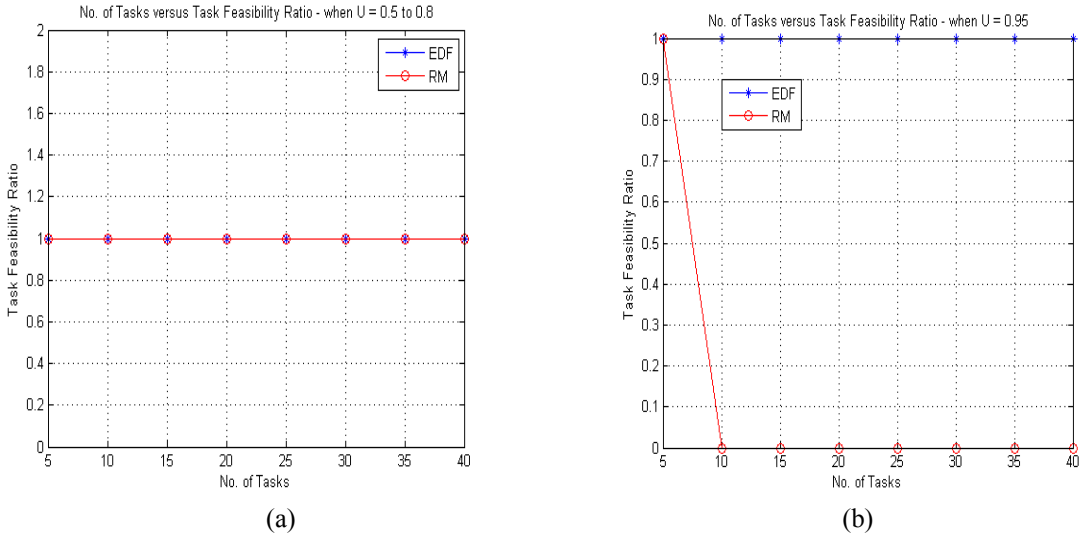


Figure 9. Task Feasibility Ratio vs. No. of Tasks for U = Constant (0.5 - 0.8, 0.95)

3. CONCLUSION

This paper presented the analysis of the most wanted industrial needed scheduling algorithms for increasing the performance of real-time systems. From the result of simulation, we have found each of them dominates each other's at different scenarios. The real benefit of RM over EDF is easier implementation in the commercial kernel. However EDF allows the full CPU utilization which means more efficient utilization of resources. These properties are very essential for a real-time system dealing with resource constraints.

REFERENCES

[1] Schneider S., "Concurrent and Real-time Systems: the CSP Approach," John Wiley and Sons Ltd, US, 2000.
[2] X. Pan, *et al.*, "Research on Real-Time Scheduling Strategy for Transient Fault Tolerance in NC System," *Proc of the Sixth Int Conf on Intelligent Systems Design and Applications*, 2006.
[3] Liberato F., *et al*., "Tolerant to Mutliple Transient Faults for Aperiodic Tasks in Hard Real Time Systems," *IEEE Transactions on Computers*, vol. 49, pp. 906-914, 2000.
[4] Shye A., *et al.*, "Using Process - Level redundancy to Exploit Multiple Cores for Transient Fault Tolerance," *Proc pf 37th Annual IEEE / IFIP Int Conf on Dependable Systems and Networks, (Edinburgh)*, pp. 297-306, 2006.
[5] M. Pandya and M. Malek, "Minimum Achievable Utilization for Fault-Tolerant Processing of Periodic Tasks," *IEEE Transactions on Computers*, vol. 47, pp. 1102-1112, 1998.
[6] M. Shanmugasundaram, *et al.*, "Approaches for Transient Fault Tolerance in Multiprocessor - A State of Art," *Indian Journal of Science and Technology*, vol. 8, pp. 1-9, 2015.
[7] Beitollahi H., *et al*., "Fault-tolerant Earliest-Deadline-First Scheduling Algorithms," *IEEE Symposium on Parallel and Distributed Processing (Long Beach, CA),* pp. 1-6, 2007.
[8] H. Aydin, "Exact Fault-Sensitive Feasibility Analysis of Real-Time Tasks," *IEEE Transactions on Computers,* vol. 56, pp. 1372 – 1386, 2007.
[9] Asadi M., *et al.*, "A Modified BCE Algorithm for Fault-Tolerance Scheduling of Periodic Tasks in Hard Real-Time Systems," *Third Asia Int Conf on Modeling & Simulation*, pp. 287-291, 2009.
[10] Mosse D., *et al*., "A Nonpreemtive Real-Time Scheduler with Recovery from Transient Faults and its applications," *IEEE Transactions on Software Engineering*, vol. 8, pp. 752-767, 2003.
[11] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment," *Journal of the Association for Computing Machinery*, vol. 20, pp. 46-61, 1973.
[12] H. Beitollahi and G. Deconinck, "Fault-Tolerant Rate-Monotonic Scheduling Algorithm in Uniprocessor Embedded Systems," *12th Pacific Rim International Symposium on Dependable Computing (Riverside, California)*, pp. 395-396, 2006.
[13] R. M. Pathan, "Fault-Tolerant Real-Time Scheduling Algorithm for Tolerating Multiple Transient Faults," *4th Int Conf on Electrical and Computer Engineering (Dhaka)*, pp. 557-580, 2006.
[14] Buttazzo G. C., "Rate Monotonic vs. EDF: Judgment Day," *ACM Journal of Real Time Systems,* vol/issue: 29(1), pp. 5-26, 2005.
[15] B. Alam and A. Kumar, "A Real Time Scheduling Algorithm for Tolerating Single Transient Fault," *Int Conf on Information Systems and Computer Networks (Mathura),* pp. 11-14, 2014.
[16] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time System*, vol. 30, pp. 129–154, 2005.
[17] L. Yang and Q. Zhu, "A Kind of New Real-time Scheduling Algorithm for Embedded Linux," *Indonesian Journal of Electrical Engineering and Computer Science*, vol/issue: 12(6), pp. 4444-4450, 2014.
[18] M. Awadalla, "Processor Speed Control for Power Reduction of Real-Time Systems Heuristically," *International Journal of Electrical and Computer Engineering*, vol/issue: 5(4), pp. 701-713, 2015.