Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EVALUATION OF EDF AND RM SCHEDULING ALGORITHMS: CHOICES AND TRADEOFFS

[1]V.KAVITHA, [2]Dr. V.KANNAN AND [3]Dr. S.RAVI

[1]Research Scholar, Department of Electronics and Communication Engineering, Dr. M.G.R. Educational and Research Institute University, Chennai.

[2] Principal, Jeppiaar Institute of Technology, Kanchipuram.

[3]Professor and Head, Department of Electronics and Communication Engineering, Dr. M.G.R. Educational and Research Institute University, Chennai.

E-mail: [1] kavithasathish2006@gmail.com , [2]drvkannan123@gmail.com,[3] ravi_mls@yahoo.com

ABSTRACT

Since the first results published in 1973 by Liu and Layland on the Rate Monotonic (RM) and Earliest Deadline First (EDF) algorithms, a lot of progress has been made in the schedulability analysis of periodic task sets. Priority based real time scheduling algorithms such as RM and EDF have been analyzed extensively in this literature to achieve optimized results in real rime operations. In the paper, RM and EDF scheduling techniques have been used, analyzed and compared based on different parameters in real time environment and these traditional priority scheduling algorithms are analyzed by addressing the following metrics: Best case response time, Worst case response time, response time jitter and latency. Past work has been extended in this direction by characterizing the behavior of the scheduling algorithms in detail using theoretical analysis as well as experimental evaluation. The results of this analysis can be used to control design choices for real time systems. Various issues have been presented on which there is still a need to work.

Keywords: *EDF, Jitter, Latency, priority scheduling, Response time, RM*

1. INTRODUCTION

A real-time operating system (RTOS) is developed for real-time applications. (Example: mobile telephones, industrial robots, or scientific research equipment). It is a subtype of operating system and it has lot of characteristics similar to common operating system in many aspects. RTOS provides basic OS functions, priority allocation, memory management, memory allocation, task management, task predictability, scheduling, interrupt latency control, timer and IPC synchronization functions [1]. It enables hardware systems to become available and provides upper level applications with system rich calls. It schedules execution in a timely manner, manages system resources and provides consistent foundation for developing application code. Realtime operating systems perform scheduling of tasks using "priority-based preemptive scheduling.

Since Liu and Layland's seminal work on scheduling algorithms for periodic tasks, several real-time scheduling algorithms have been proposed and analyzed in the literature. In particular, Rate Monotonic (RM) and Earliest Deadline First (EDF) are the most commonly implemented algorithms in practice as well as the most analyzed algorithms in literature. In RM algorithm tasks have to be periodic in nature and deadline must be equal to its period. Tasks are scheduled according to their period. The rate of task is the inverse of its period. This algorithm implemented by assigning fixed priority to tasks, the higher its rate, higher the priority. The most common dynamic priority scheduling algorithm for a real-time system is the EDF[7]. Here priorities are dynamically reassigned at run-time based on the time still available for each task to reach its next deadline. Both static and dynamic systems are scheduled by EDF algorithm [2].

But there are other constraints such as Response Time, Latency and Response Time Jitter, which may be of equal if not of more importance than preemptions (or energy consumption) for specific application scenarios. Most of the comparative evaluations so far have either focused only on RM and EDF or only on limited analysis of preemption and energy consumption. In this paper we attempt a comprehensive experimental evaluation of RM and EDF scheduling algorithms by comparing them on several metrics using task sets.

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. RELATED WORK

A. R.R.Maggavi, D.A.Torse implemented an embedded platform for RM algorithm using uc/os-II. In this paper, RM scheduling is implemented on uc/os-II and its operation in terms of processor utilization and task execution time were discussed. ROM image file was ported into 8051 based microcontroller flash and tested for scheduling with RMA. The results have indicated optimum utilization of processor with RMA scheduler for realizing low cost hardware and software for developing embedded system. The overall CPU utilization was 70.18% with RMA approach using 8051 [8].

B. Biju K Raveendran attempted a comprehensive experimental evaluation of six different scheduling algorithms by comparing them on several metrics using simulated task sets. Scheduling algorithms for periodic tasks, several real-time scheduling algorithms have been proposed and analyzed in the literature. In particular, Rate Monotonic (RM) and Earliest Deadline First (EDF) are the most commonly implemented algorithms in practice as well as the most analyzed algorithms in literature. More recently, the proliferation of mobile embedded computing devices running on limited battery power has brought to focus the issue of power-aware computing – in particular power-aware scheduling. Among the factors affecting power consumption, context switching is one that is often clearly identified as 'overhead'. A few real-time scheduling algorithms have been proposed with the aim of reducing the number of preemptions. But most of the comparative evaluations so far have either focused only on RM and EDF or only on limited analysis of preemption and energy consumption [9].

C. Omar U. Pereira Zapata, Pedro Mejıa Alvarez implemented the best known partitioned and global multiprocessor scheduling algorithms and to compare their performance. The performance of the global and the partitioned algorithms were compared using RM and EDF scheduling policies under off-line and online algorithms. Performance evaluation study was designed with the aim to introduce the conditions under which some algorithms are better than the others and as a future reference for the design of new algorithms. Extensive simulation experiments were conducted to evaluate the performance and computational of the algorithms. In our simulation experiments, the schedulability tests were evaluated for different values and number of tasks [10].

3. PARAMETERS FOR COMPARISON

3.1 Response Time

It is the time taken in an interactive program from the issuance of the command to the commence of the response of the command.

Here, worst case and best case response times under EDF and RM scheduling are going to be discussed.

3.2 Response Time Jitter

Jitter is the size of the variation in the arrival or departure times of a periodic task. Jitter normally causes no problems as long as the actions all stay within the correct period.

Response time jitter refers to maximum length of the interval in which a task can be released non-deterministically.

3.2 Latency

Latency is a measure of the time between a particular task and a system's response to that task, and it's quite often a focus for real-time developers.

For a task τi, we define the maximum input-output latency as $Li = \max_i$(finish-time– start-time)

4. SCHEDULABILITY ANALYSIS

In 1973, Liu and Layland analyzed properties of two basic priority assignment rules: the Rate Monotonic (RM) algorithm and the Earliest Deadline First (EDF) algorithm. According to RM, tasks are assigned fixed priorities ordered as the rates, so the task with the smallest period receives the highest priority. According to EDF, priorities are assigned dynamically and are inversely proportional to the absolute deadlines of the active jobs. The major contribution of Liu and Layland was to derive a simple guarantee test to verify the schedulability of a periodic task set under both algorithms. Each periodic task τi is considered as an infinite sequence of jobs τi,k (k =1, 2, . . .). Each job is characterized by a worst-case execution time Ci, a relative deadline Di and an absolute deadline di,k =ri,k + Di (ri,k) is the arrival time of task) [4].

The ratio Ui =Ci /Ti is called the utilization factor of task τi and represents the fraction of processor time used by that task. The value,

$$\sum_{i=1}^{n} \text{Ui} \qquad (1)$$

is called the total processor utilization factor and represents the fraction of processor time used by the periodic task set. Clearly, if Up >1 no feasible

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

schedule exists for the task set with any algorithm. If Up ≤1, the feasibility of the schedule depends on the task set parameters and on the scheduling algorithm used in the system.

The basic schedulability conditions for RM and EDF proposed by Liu and Layland were derived for a set of n periodic tasks under the assumptions that all tasks start simultaneously at time t = 0, relative deadlines are equal to periods (that is, di,k =k Ti) and tasks are independent. Under such assumptions, a set of n periodic tasks is schedulable by the RM algorithm if [5],

$$\sum_{i=1}^{n} Ui \le 2^{1/n} - 1 \qquad (2)$$

RM algorithm is able to feasibly schedule task sets with processor utilization up to about 88%.

Under the same assumptions, a set of n periodic tasks is schedulable by the EDF algorithm if and only if[3],

$$\sum_{i=1}^{n} \text{Ui} \le 1 \qquad (3)$$

5. PRELIMINARIES

In this paper, a performance analysis of two scheduling algorithms is carried out where the priorities of the tasks are assigned statically (using RM) and dynamically (using EDF). The problem to be studied is to schedule a set of n real-time tasks τi = τ1, τ2,…τn, and a task is usually a thread or a process within an operating system. The parameters that define a task are: the execution time Ci, the period Ti, and the deadline Di[6]. It is considered that only periodic and preemptive tasks may be executed in the system. Each periodic task, denoted by τi, is composed of an infinite sequence of jobs. The period Ti of the periodic task *τi* is a fixed time interval between release times of consecutive jobs. Its execution time Ci is the maximum execution time of all the jobs. The period and the execution time of task τi satisfies that Ti > 0 and 0 < Ci < Ti = Di; (i = 1….. n).

5.1 Response Time Analysis

To analyze response times of EDF and RM algorithms, the idea is that tasks sometimes must be activated at the same time, called a critical point. A task set is schedulable when all tasks are activated at the same critical point and all deadlines are met. A critical point will always appear in a system provided no task in the set has an offset.

5.1.1 Worst-case response time analysis

Under RM scheduling, assuming $D_i \le T_i$, the worst-case response time of task τi, is given by the well-known equation[5].

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{Ri}{Tj} \right\rceil Cj \qquad (4)$$

Here, *Ci* is the maximal execution time, Ri is the response time for task τi and the set hp(i) consists of all tasks with higher priority than task τi. Priorities are assumed to be unique and tasks will not voluntarily suspend themselves. The terms in the summation express how many times each higher priority task will interfere with task τi multiplied with the higher priority task's execution time, *Cj*. To solve the above equation the response time, *Ri*, must be factored out.

This is in general impossible. Therefore one must iterate on Ri with the following formulation:

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{Ri}{Tj}^{n} \right\rceil Cj \qquad (5)$$

Where n is the iteration number. In the first iteration the first response time, R_i^1 , is usually set to the task's execution time, Ci. Then it is iterated until two subsequent response times are the same, or when the last response time exceeds the task's deadline.

Under EDF scheduling, the calculation of worst-case response-time analysis is more complicated and based on the theorem "the worst case response time of task τi is found in a busy period in which all tasks are released synchronously at the beginning of the period at the maximum rate".
The worst-case response time of task τi is given by,

$$R_i = \max_{a \ge 0} \left[Ci, \left[\max \{ Li(a) - a \} \right] \right] \qquad (6)$$

Where the busy interval $Li\,(a)$ is given by the equation,

$$Li(a) = Wi\{a, Li(a)\} + \left(1 + \left\lceil \frac{a}{Ti} \right\rceil\right) Ci \qquad (7)$$

Here, a denotes the arrival time or release time of task and d=a+ D*i*, L is the constant delay of arrival time and $R_{i,\text{ the}}$ response time of task. Where the higher-priority workload W_i {a,Li(a)} is given by,

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$W_i(a, L_i(a)) = \sum_{j \neq i, D_j \leq a + D_i} \left\{ \min \left\lceil \frac{L_i(a)}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} C_j \quad (8)$$

Only a finite number of values of a must be checked when evaluating the above equation.

5.1.2 Best-case response time analysis

Under RM scheduling, exact best-case analysis has recently been developed for the case D≤ T. The best-case response time of task τi is given by the equation

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} - 1 \right\rceil C_j \quad (9)$$

Where C_i^b denotes the best-case execution time of task τi.

Under EDF scheduling, no exact best-case analysis is known to exist. A trivial lower bound, R_i, on the best-case response time of task τi is given by,

$$\underline{R_i^b} = C_i^b \quad (10)$$

This is actually a quite good bound for the shortest-period tasks. The longest-period tasks can, however, have much longer best-case response times, especially if the system load is high. A tighter lower bound on the best-case response time can be obtained by interference analysis, our proposed lower bound, $\underline{R_i}^b$, is given by the equation

$$\underline{R_i^b} = C_i^b + \sum \left\lceil \min \frac{\{R_i^b, D_i - D_j\}}{T_j} - 1 \right\rceil C_j^b \quad (11)$$

This expression provides only a lower bound, since it does not take any initial (partial) interference from task Tj into account. It is possible to improve the formula slightly by including some obvious cases where initial interference must occur. It is conjectured, however, that the expression for the exact best-case response time is as complex as the formula for exact worst-case response time under EDF.

5.2 Jitter Analysis

So far, it is assumed that tasks are released at a constant rate (at the start of a constant period) .This is true in practice and a realistic assumption. However, there are situations where the period or rather the release time may 'jitter' or change a little, but the jitter is bounded with some constant. The jitter may cause some task missing deadline and certain systems might require that jitter be minimized as much as possible.

Real-time programmers commonly handle tasks with tight jitter requirements in one of two ways:

• If only one or two actions have tight jitter requirements, set those actions to be top priority. Note: This method only works with a very small number of actions.

• If jitter must be minimized for a larger number of tasks, split each task into two, one which computes the output but does not pass it on, and one which passes the output on. Set the second task's priority to be very high and its period to be the same as that of the first task. An action scheduled with this approach will always run one cycle behind schedule, but will have very tight jitter.

Most real-time systems use some combination of these two methods.

6. TASK MODEL AND METHODOLOGY

In this section– Rate Monotonic (RM) and Earliest Deadline First (EDF) priority scheduling algorithms are going to be implemented using an example so that the ensuing discussion and analyses are clear. Both these algorithms are priority based scheduling algorithms for periodic tasks with hard deadlines. Table 1 show the example task set. We took the following assumptions in our presentation of the algorithms:

- Arrival times for first instances of all tasks are assumed to be 0.
- Period (J) for a job J is the same as period (T) where J is an instance of task T.
- For each job J, deadline (k) = arrival time (k) + period (k). All instances of a periodic task should have the same computation time.
- All instances of a periodic task should have the same relative deadline, which is equal to the period.

Table 1: Example task set

Task	Arrival time(for first instance)	Period (ms)	Execution time(ms)
τ1	0	4	1
τ2	0	5	2
τ3	0	20	7

This job list is derived from Table 1 under our assumption deadline (k) = arrival time (k) + period (k) for any job k. These jobs are arranged in the order of deadline and when the deadline is same, in the order of arrival – as this is the likely arrangement of a queue.

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Job list for table 1

Job	Arrival time (ms)	Execution time (ms)	Deadline
k1(τ1)	0	1	4
k2(τ2)	0	2	5
k3(τ1)	4	1	8
k4(τ2)	5	2	10
k5(τ1)	8	1	12
k6(τ2)	10	2	15
k7(τ1)	12	1	16
k8(τ3)	0	7	20
k9(τ2)	15	2	20
k10(τ1)	16	1	20

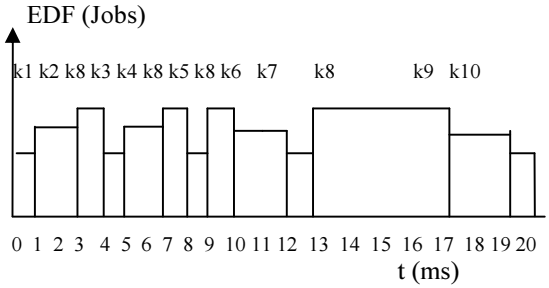

Figure 1: Timeline execution of EDF algorithm

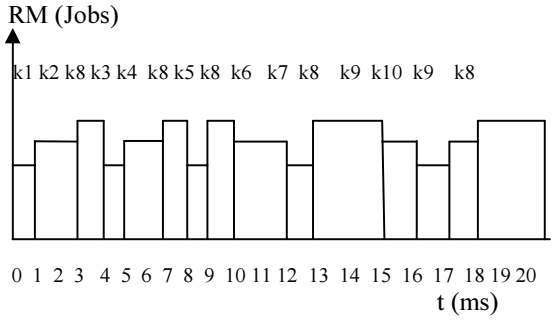

Figure 2: Timeline execution of RM algorithm

A task set is schedulable when all tasks are activated at the same critical point and all deadlines are met. A critical point will always appear in a system provided no task in the set has an offset.

6.1 Calculation Of RM Algorithm Parameters

6.1.1 Worst case response time

For the given task set τ1=4ms, τ2=5ms, τ3=20ms, C1=1ms, C2=2ms and C3=7ms, the worst case response time of RM algorithm can be computed using the equation (5),

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{Ri^{n}}{Tj} \right\rceil Cj \qquad (5)$$

When priorities are set as fixed the response times of tasks can be calculated. We begin with the highest priority task τ1. No other task will interfere its execution, so the response time will be equal to its maximal execution time 1.

R $\tau 1^1$ =*1 ;(* Best case response time of first task) the response time for the next highest priority task, τ2, is then calculated. Only one task, τ1 has higher priority and will interfere with τ2. Response time for τ2 is calculated as:

R $\tau 2^1$= 2+2/4*1=2+1=3

R $\tau 2^2$= 2+3/4*1=2+1=3

↓

(First iteration response)

R $\tau 2^1$=R $\tau 2^2$=3;

The sequence of iterations terminates with the response time 3 for task *τ2*.

Next task to consider is *τ3*, which will be interfered by two higher priority tasks. The response time iterations are:

R $\tau 3^1$= 7+7/4*1+7/5*2=7+2+3=12

R $\tau 3^2$= 7+12/4*1+12/5*2=7+3+6=16

R $\tau 3^3$= 7+16/4*1+16/5*2=7+4+8=19

R $\tau 3^3$= 7+19/4*1+19/5*2=7+5+8=20
R $\tau 3^4$= 7+20/4*1+20/5*2=7+5+8=20

R $\tau 3^3$= R$\tau 3^4$=20

6.1.2 Best case response time

The best case response time of RM algorithm can be computed using the equation (9) as shown.

$$R_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{Ri^{n}}{Tj} - 1 \right\rceil Cj \qquad (9)$$

It is calculated for all the jobs using the same iterative method used for worst case response time.

R $\tau 1^b$=1;

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$R\ \tau2^b$=2;

$R\ \tau3^b$=12;

6.1.3 Output jitter

Output jitter and constant is compute for each task if response-time analysis is done. Let Ri and Rbi denote, respectively, the worst-case and best-case response times of task τi. The jitter, J_i, is then given by,

$$Ji = R\ i - R_i^b \quad (12)$$

J τ1=0; J τ2=1 and J τ3=8 respectively under RM scheduling for the given task set.

6.1.4 Latency

Another parameter that it is important to minimize in control applications is the input-output latency. Assuming that a control task τi acquires inputs at the beginning of each instance and delivers control outputs at the end, the maximum input-output latency is defined as,

$$Li = \max_i(\text{finish-time} - \text{start-time})$$

It is also defined as,

$$Li = R_i^b \quad (13)$$

Under RM scheduling,

L τ1=1;L τ1=2 and L τ3=12 respectively.

6.2 Calculation Of EDF Algorithm Parameters

6.2.1 Worst case response time

To calculate worst-case response time of task τi under EDF scheduling, consider equations given below:

$$R_i = \max_{a\geq 0}\left[Ci, \left[\max\{Li(a) - a\}\right]\right] \quad (6)$$

$$Li(a) = Wi\{a, Li(a)\} + \left(1 + \left\lfloor \frac{a}{Ti} \right\rfloor\right) Ci \quad (7)$$

$$Wi\{a, Li(a)\} = \sum_{j\neq i, Dj\leq a+Di} \left\{\min\left\lceil \frac{Li(a)}{Tj} \right\rceil, 1 + \left\lfloor \frac{a+Di-Dj}{Tj} \right\rfloor\right\} Cj \quad (8)$$

For task τ1, $W_i\{a,Li(a)\}$=1; Li(a)=1 and R τ1=1

Similarly, R τ2=2.75 and R τ3=13 respectively.

6.2.2 Best case response time

The best case response time of EDF algorithm can be computed using the equation (9) as shown.

$$\underline{Ri^b} = Ci^b + \sum \left\lceil \min\frac{\{Ri^b, Di-Dj\}}{Tj} - 1 \right\rceil Cj^b \quad (11)$$

But for tasks with short period, response time will be equal to computation time as per equation (10)

So, $R\ \tau1^b = C\ \tau1^b = 1$;

$R\ \tau2^b$=1.25;

$$R\tau3^b = 7 + \min\left\lceil \frac{\{7,20-5\}}{5} - 1 \right\rceil * 2 + \min\left\lceil \frac{\{7,20-4\}}{4} - 1 \right\rceil * 1$$

=8.55;

6.2.3 Output jitter

The jitter calculation of EDF algorithm can be computed using the same equation (12) as shown.

$$Ji = R\ i - R_i^b \quad (12)$$

J τ1=0; J τ2=1.5 and J τ3=4.45 respectively for the given task set.

6.2.4 Latency

Under EDF scheduling,

L τ1=1; L τ1=1.25 and L τ3=8.55 respectively.

The above results are tabulated to make comparative analysis.

Table 3: Parameters of RM scheduling

Tasks	$R\ i$	R_i^b	Li	$J\ i$
T1	1	1	1	0
T2	3	2	2	1
T3	20	12	12	8

Table 4: Parameters of EDF scheduling

Tasks	$R\ i$	R_i^b	Li	Ji
T1	1	1	1	0
T2	2.75	1.25	1.25	1.5
T3	13	8.55	8.55	4.45

7. RESULTS

In this work, the parameters of under EDF and RM scheduling were calculated and the results are shown in fig. 3, fig. 4, fig. 5 and fig. 6 respectively.

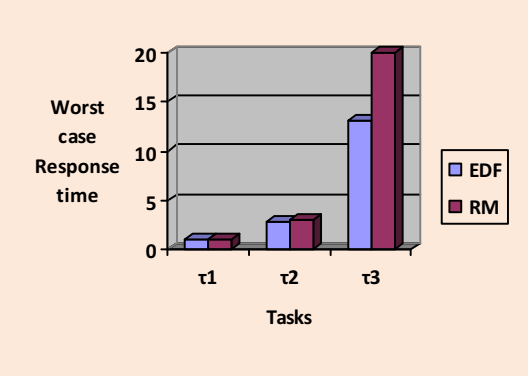

Figure 3. Worst case response time comparative analysis

Journal of Theoretical and Applied Information Technology
20th August 2014. Vol. 66 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

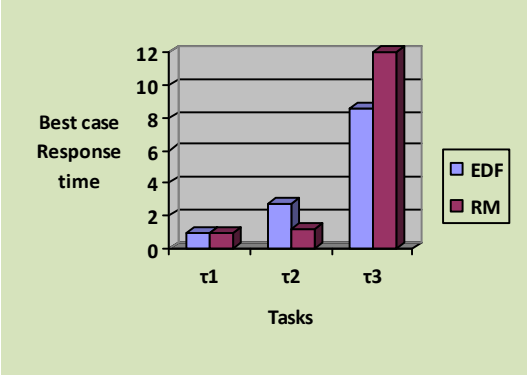

Figure 4. Worst Case Response Time Comparative Analysis

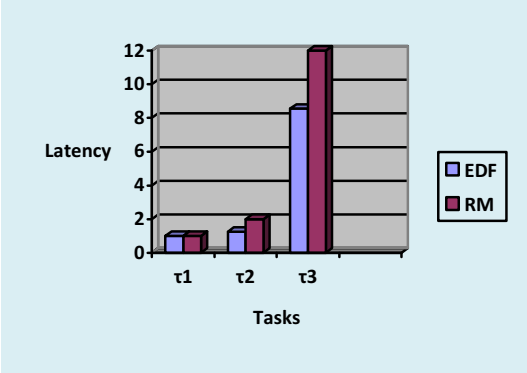

Figure 5 Latency Vs Tasks Under EDF And RM Scheduling

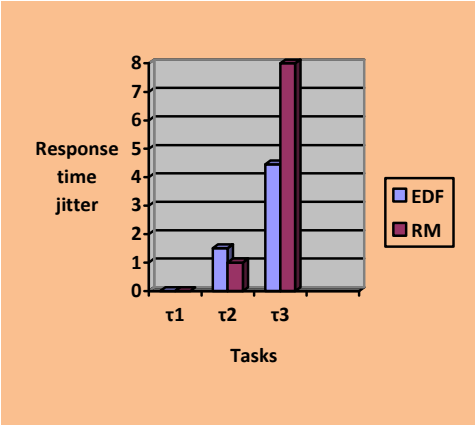

Figure 6. Response Time Jitter Comparative Analysis

A comprehensive summary of this evaluation is presented (see Table 5) in the form of desired performance characteristics and the corresponding choice of scheduling algorithms. The behavior of RM and EDF as a function of the processor load, for a fixed number of tasks has been tested and it is interesting to observe different behavior of RM and EDF for high processor loads. For larger task sets, the number of preemptions caused by RM increases, thus the overhead due to the context switch time is higher under RM than EDF. RM reduces the jitter of high priority tasks at the expenses of tasks with lower priority, EDF treats tasks more evenly, obtaining a significant reduction in the jitter of the tasks with long periods for a small increase in the jitter of tasks with shorter periods and EDF can always achieve a shorter latency than RM, for any task. When considering the development of the scheduling algorithm on top of a kernel based on a set of fixed priority levels, it is indeed true that the EDF implementation is not easy, nor efficient.

Table 5: Ready Reckoner For Choice Of Scheduling Algorithm

Desired performance characteristics	Desired scheduling algorithm
Low response time	RM under low load and EDF under high load
Low response time jitter	RM under low load and EDF under high load
Low latency	EDF
Ease of implementation	RM under low load and EDF under high load
Context switch	More in RM than EDF

8. CONCLUSION

The main goal of this paper was to survey the best known scheduling algorithms and to compare their performance metrics. Performance metrics of the RM and EDF scheduling algorithms were compared. This paper has only treated output jitter. In some applications, sampling jitter is also an issue. The topic of best-case response-time analysis needs to be investigated further as it is not the exact one and having more complications which may produce wrong results under high load conditions. For instance, exact analysis best-case response time

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

under EDF could be developed. There are a lot of misconceptions about the properties of these two scheduling algorithms that for a number of reasons unfairly penalize EDF. The typical motivations that are usually given in favor of RM state that RM is easier to implement, it introduces less runtime overhead, it is easier to analyze, it is more predictable in overload conditions, and causes less jitter in task execution. However, EDF introduces less runtime overhead than RM, when context switches are taken into account. In fact, to enforce the fixed priority order, the number of preemptions that typically occur under RM is much higher than under EDF.The performance evaluation study was designed with the aim to introduce the conditions under which one algorithm is better than the other. Finally, both RM and EDF are not very well suited to work in overload conditions and to achieve jitter control. To cope with overloads, specific extensions have to be done in this work as a future scope for the design of new algorithms.

REFRENCES:

[1] Raj Kamal, "Embedded Systems-Architecture, programming and Design 2nd Ed", *McGraw-Hill Education (India) Pvt. Ltd., ISBN-13: 0070667640*, 2008, pp. 296-423.

[2] D.G.Harukith, M. S. Ali and Poonam Lohiya, Real time scheduling algorithms for wireless sensor networks", *Circuits and Systems: An International Journal (CSIJ),* Coimbatore Inst. Vol. 1, No. 1, pp.11-18, January 2014.

[3] Shweta.R , Dinesh Rotake, "Design and implementation of EDF Algorithm with hardware core processor", *International Journal of Advance Research in Computer Science and Management Studies,* Volume 2, Issue 1,pp. 269-275,January 2014.

[4] Jerzy Martyana, "Scheduling Algorithm for delay and jitter reduction of periodic tasks in realtime systems", *PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review),* ISSN 0033-2097, R. 87 NR 1/2011.

[5] M.V.Panduranaga Rao, K.C.Shet, , "A Research in realtime scheduling policy for embedded system domain", *Clei Electronic Journal,* Volume 12, Number 2, Paper 4, AUGUST 2009.

[6] Debashreet Das, Sibarama Panigrahi and Ashok Bhoi, "Feasibility under fixed priority scheduling with fixed preemption points", *Proceedings of the 16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(*RTCSA 2010), Macau, China, August 23-25, 2010.

[7] Zhang, Burns, "Schedulability Analysis for realtilme systems with EDF scheduling", *IEEE Transactions of computers,* VOL. 58, PP: 1-9, 2009.

[8] R.Maggavi, D.A.Torse,"RM Analysis for uc/os-II on Embedded systems", *Electric Power Energy Research,* Vol. 54, No. 2, pp. 101-111, 2000.

[9] Biju K Raveendran, K.Durga Prasad, "Variants of priority scheduling algorithms for reducing context switches in realtime systems", *Springer-Verlag Berlin Heidelberg, S. Chaudhuri et al. (Eds.):* ICDCN 2006, LNCS 4308, pp. 466 – 478, 2006.

[10] *Omar* U. Pereira Zapata, Pedro Mejıa Alvarez,"EDF and RM multiprocessor scheduling algorithms: Survey and performance evaluations", *fopereira pmejiag@computacion.cs.cinvestav.mx,* pp: 1-24, 2006.