Comparison of Real Time Task Scheduling Algorithms

Vijayshree Shinde
PG Scholar, Department of Computer Engineering
Terna Engineering College, Navi Mumbai, India

Seema C. Biday, PhD
Professor, Department of Electronics Engineering
Terna Engineering College, Navi Mumbai, India

ABSTRACT
In a Real-Time System, the correctness of the system is not only depending on the logical result of the computation but also on the time at which result is produced is very important. In real time system, scheduling is effected using certain criteria that ensure processes complete their various tasks at a specific time of completion. The quality of real-time scheduling algorithm has a direct impact on real-time system's working. We studied popular scheduling algorithms mainly Earliest Deadline First, Rate Monotonic, Deadline Monotonic, Least laxity First, Group Earliest Deadline First and Group Priority Earliest Deadline First for periodic task. We observe that the choice of a scheduling algorithm is important in designing a real-time system. We conclude by discussing the results of the Real-Time scheduling algorithm survey.

Keywords
Real-Time system, Real-Time task scheduling, Deadline, Execution time, Period, EDF,RM, DM, GPEDF, GEDF, LLF.

1. INTRODUCTION
Real time system must respond to externally generated inputs within a specified period to avoid failure. The deadline of a task is the point in time before which the task must complete its execution [1]. There are three types of deadlines, which are mentioned below,

- Soft Deadline

In this type of deadline, task could miss some deadline and the system could still work correctly. Reservation systems is one of the example of soft deadline.

- Firm Deadline

This deadline is one in which the results come after the deadline is missed is of no usefulness. Infrequent deadline misses are tolerable. These sorts of deadlines are utilized as a part of system which are playing out some vital operations.

- Hard Deadline

If task miss some deadline, then catastrophe results will occur, such type of deadline is known as hard deadline. The system which are performing critical applications like air traffic control go under this category.

The application of real time systems can be found in Robotics, Pacemakers, Chemical Plants, Antimissile Systems, and Embedded Systems etc. to name a few [2]. There are three kinds of real-time tasks, depending on their arrival pattern: periodic task (Periodic tasks execute at every known fixed time interval. Normally, periodic tasks have constraints which indicate that instances of time constraints), Aperiodic task (aperiodic tasks execute at any random time constraints and would not have pre-defined timing sequence) and Sporadic task (Sporadic tasks are combination of both periodic and Aperiodic, where in, the executing time is Aperiodic but the executing rate is periodic in nature). The time constraints are usually a deadline. Scheduling mechanism is the important concept of a computer system, it is the strategy by which computer system decided which task should be executed at any given time. Scheduling algorithm for uniprocessor systems must guarantee to apportion the enough time to all the system task at specific purposes of time that they can meet their deadline as far as possible.

The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible when we consider soft real-time system [3]. To achieve this goal, vast researches on real-time task scheduling have been conducted. Real-time scheduling can be divided into two categories: Static and Dynamic. In static algorithm al priorities are assigned at design time and those priorities is remains constant for the life time of a task. Dynamic algorithms assign priorities at runtime, based on execution parameters of tasks. Dynamic scheduling can be either with static priority or dynamic priority. Rate Monotonic [4] and Deadline Monotonic [5] are examples of dynamic scheduling with static priority. EDF [4](Earliest Deadline First) and LST [6] (Least Slack Time First) are examples of dynamic scheduling with dynamic priority. EDF and LST algorithms are optimal under the condition that the jobs are preemptive, there is only one processor and the processor is not overloaded [7]. But the limitation of these algorithms is, their execution diminishes exponentially if the system turns out to be marginally overloaded. This paper makes comparison of different task scheduling algorithms.

The rest of paper is organized as follows: in section 2, described real-time task model, section 3 described real-time task scheduling algorithms adopted in this paper, in section 4, described the comparison of real-time task scheduling algorithms, and in section 5, paper is concluded.

2. REAL-TIME TASK MODEL
Let T= {T1, T2, Ti,….. Tn} be a set of N periodic task in a uniprocessor system. The tasks are mutually independent and the processor time is the only resource that needs to be scheduled. Each task Ti is defined as Ti = (Ci, Pi, Di), where Ci is its execution time, Pi is its period and Di is its deadline, $Ci \leq Di$.

3. REAL-TIME TASK SCHEDULING ALGORITHMS
3.1 Earliest Deadline First Algorithm
In 1973 Liu and Layland, suggested the most popular real time scheduling algorithms Earliest Deadline First (EDF) [4]. EDF is a dynamic priority algorithm in which task with the earliest deadline has the highest priority. EDF is an optimal uniprocessor scheduling algorithm. The optimal scheduling algorithm gives 100% CPU utilization

EDF algorithm gives best performance and minimize miss ratio, when systems operating under low or moderate levels of resource and data contention. However, the performance of Earliest Deadline First algorithm is suddenly degraded in an overloaded system. This is because, under heavy loading, tasks gain high priority only when they are close to their deadlines.

Consider Table 1, which represents a sample tasks set that will be used as common example throughout this paper to better understand the differences among real time task scheduling approaches. This task set is schedule using fully pre-emptive Earliest Deadline First Scheduling algorithm show in figure 1.

Table 1. Real-Time Task set

Task	C_i	D_i	P_i
T1	1	4	4
T2	2	5	5
T3	2	7	7

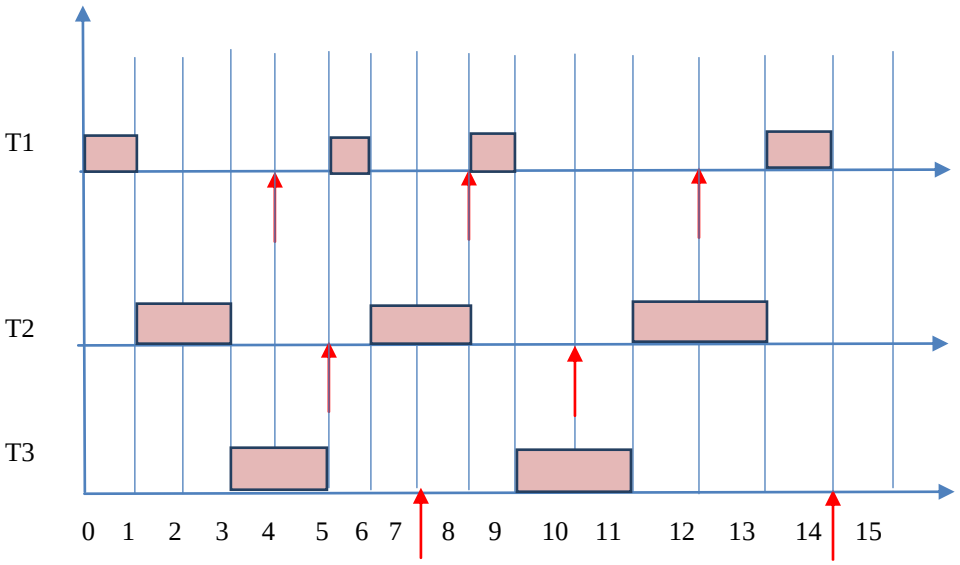

Fig 1: The timing diagram of EDF scheduling

3.2 Rate Monotonic

RM is a preemptive and static priority scheduling algorithm on uniprocessor systems [4]. RM assigns the higher priority to the task with the shortest period, assuming that periods are equal to deadlines (Pi=Di), because if the demand rate is more, the period would be shorter and the priority would increase. Therefore, it is used in periodic tasks. One major limitation is CPU not always fully utilize when fixed priority scheduling algorithm is used. In this algorithm, all the tasks will meet their deadline if the CPU utilization factor (U) is less than N(21/N-1) where N is the number of tasks to be scheduled [8].

- Periodic tasks have constant known execution times and are ready for execution at the beginning of each period(T).
- Deadlines(D) for tasks are at the end of each period: (D = T)
- The tasks are independent, that is, there is no precedence between tasks and they do not block each other.
- Scheduling overhead due to context switches and swapping etc. are assumed to be zero.

Sample task set are schedule by RM scheduling algorithm as shown in figure 2.

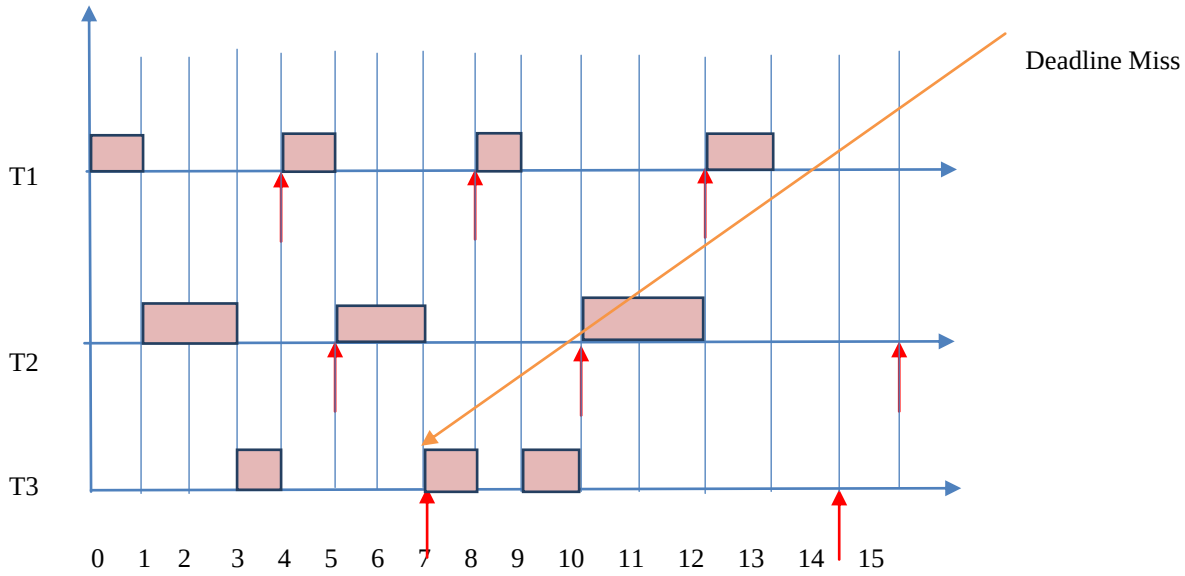

Fig 2: The timing diagram of RM scheduling

3.3 Deadline Monotonic

Deadline Monotonic (DM) is the optimal fixed priority scheduling algorithm where the assigned priorities are inversely proportional to the task deadline. DM used when D < T which allows us to see RM as a special case of DM. DM executes at any time instant the instance of the ready task with the shortest deadline, first. If two or more tasks have the same deadline, then DM randomly selects one for execution next. DM becomes equivalent to the RM algorithm when the deadlines of tasks are equal to their period [9].

3.4 Least Laxity First

LLF is another optimal dynamic-priority scheduling algorithm. The laxity of a process is defined as the deadline minus remaining computation time. The laxity of a job is the maximal amount of time that the job can wait and still meet its deadline. The algorithm gives the highest priority to the dynamic job with the littlest laxity. Then the job with the highest priority is executed. While a process is executing, it can be preempted by another whose laxity has less than that of running process. A problem arises with this scheme when two processes have similar laxities. One process will run for a short period while and then get preempted by the other and vice versa. Hence, numerous context switches happen in the lifetime of the processes. The least laxity first algorithm is an optimal scheduling algorithm for systems with periodic real-time tasks [10].

3.5 Group Earliest Deadline First

gEDF was developed for improving the success ratio of EDF during overload condition of soft real time multimedia application. The initiator pioneered the idea of group scheduling, where jobs with near deadlines were group together using an algorithm. After grouping jobs within a group are schedule using shortest job first scheduling [9, 11]. Group range parameter (Gr) determines which job gets into which group. It is simply a percentage value of the job at the head of a queue's absolute deadline. Mathematically it is defined as

gEDF Group = $\{\tau k \mid \tau k \in QgEDF , d k - d1 \leq d1 Gr ,$

$1 \leq k, m \leq | QgEDF | \}$ (1)

in which:

- d1 is the dynamic deadline of the first job in the group
- QgEDF is a queue for gEDF and
- | QgEDF | represents the length of queue
- m is the number of all ready jobs in a system [9]

3.6 Group priority Earliest Deadline First

GPEDF perform schedulability test prior to grouping a particular job. Following method is used to solve the problem of how to group jobs together [12].

GPEDF perform schedulability test prior to grouping a particular job. Following method is used to solve the problem of how to group jobs together [12].

$$j=1,2,\ldots,N \quad \sum_{i=1}^{j} \frac{C_i}{D_i} + \frac{C_{sum} + C_{ex}}{D_j} \leq 1 \qquad (2)$$

all the jobs in job set Jt behind first job in a set can be executed before first job and the system will still be schedulable, where Csum is the sum of the execution times of the jobs in job set except first job in job set, and Cex is the sum of execution time of the jobs that would be ready later than time t and have absolute deadline shorter than last job in job set. If there is a job set Jt which satisfies eq. (2) at time t, the order of the jobs in job set Jt can be changed randomly. This means that the jobs can be form in job set Jt into a group, in which the jobs can be reordered as required without reducing the schedulability. GPEDF scheduling algorithm can be described in three parts as follows;

First part is enqueue, when a new job arrives, enqueue sort new job into Jt. The second dequeue and third exqueue methods invoked every time unit. dequeue deletes the jobs which have absolute deadlines shorter than current time t. exqueue creates group of jobs and execute shortest job first

algorithm within groups. When there is no group in the system, the jobs are put into group one by one according to the order they appeared in job set Jt and will be stopped until one job cannot satisfy Eq. (2). If a job cannot form a group with other jobs, then it forms the group with itself. If the system is overloaded and when there is only one job in the group at that time a job may not be completed successfully because the remaining time may not be enough for the job to execute.

In the GPEDF scheduling algorithm, jobs with short execution time can be executed first in the group, which leaves more time for other jobs to execute. This allows more jobs to be completed, the number of switches is decreased and the response is reduced.

all the jobs in job set Jt behind first job in a set can be executed before first job and the system will still be schedulable, where Csum is the sum of the execution times of the jobs in job set except first job in job set, and Cex is the sum of execution time of the jobs that would be ready later than time t and have absolute deadline shorter than last job in job set. If there is a job set Jt which satisfies eq. (1) at time t, the order of the jobs in job set Jt can be changed randomly. This means that the jobs can be form in job set Jt into a group, in which the jobs can be reordered as required without reducing the schedulability. GPEDF scheduling algorithm can be described in three parts as follows;

First part is enqueue, when a new job arrives, enqueue sort new job into Jt. The second dequeue and third exqueue methods invoked every time unit. dequeue deletes the jobs which have absolute deadlines shorter than current time t. exqueue creates group of jobs and execute shortest job first algorithm within groups. When there is no group in the system, the jobs are put into group one by one according to the order they appeared in job set Jt and will be stopped until one job cannot satisfy Eq. (1). If a job cannot form a group with other jobs, then it forms the group with itself. If the system is overloaded and when there is only one job in the group at that time a job may not be completed successfully because the remaining time may not be enough for the job to execute.

In the GPEDF scheduling algorithm, jobs with short execution time can be executed first in the group, which leaves more time for other jobs to execute. This allows more jobs to be completed, the number of switches is decreased and the response is reduced.

4. COMPARISON OF REAL TIME TASK SCHEDULING ALGORITHMS

We observed the performance of scheduling algorithm from the work done by the various researchers in the field of real time scheduling, as shown in table 2.

Table 2. Comparison of real time scheduling algorithm

Algorithms	Implementation	Priority Assignment	Scheduling Criteria	Preemptive/ Non-Preemptive	CPU Utilization	Efficiency
EDF	Difficult	Dynamic	Deadline	Preemptive	Full Utilization	Efficient in Underloaded Condition
RM	Simple	Static	Period	Preemptive	Less	Efficient in overloaded condition as compared to EDF
DM	Simple	Static	Relative Deadline	Preemptive	More as compared to RM	Efficient
LLF	Difficult	Dynamic	Laxity	Preemptive	Full Utilization	Efficient
GEDF	Difficult	Dynamic	Deadline and within group Shortest Execution time (SJF)	Non-Preemptive	Full Utilization	Efficient in Non-preemptive environment
GPEDF	Difficult	Dynamic	Deadline and within group SJF	Preemptive	Full Utilization	Efficient

5. CONCLUSION

Comparative study of some existing real time scheduling algorithms has been done in this paper. It has been observed that deadline is the most important concept in real time systems and to meet this deadline real time scheduling algorithm is the most important topic. Allocating and scheduling the tasks are very complicated in the real-time system. Different approaches for allocation of tasks and scheduling the tasks are defined by different researchers. The tasks may be either static or dynamic. Earliest deadline first is an optimal dynamic priority algorithm. EDF is efficient scheduling algorithm if the CPU utilization is less than 100% but algorithm gives poor performance when system is

overloaded. Fixed priority scheduling algorithms are easier to implement and hence widely used. Rate Monotonic is an optimal static priority scheduling algorithm. Deadline Monotonic is an optimal static priority scheduling algorithm when deadline is less than or equal to period. Group priority Earliest Deadline first algorithm is efficient scheduling algorithm it gives better context switching, response time and CPU utilization.

In future a new algorithm should be developed which improve the performance of EDF scheduling algorithm in overloaded condition and give the best performance in underloaded condition. The new algorithm will be very useful when future workload of the system is changeable

6. ACKNOWLEDGMENTS

Thanks to the Computer Engineering department and Terna Engineering College for their valuable support

7. REFERENCES

[1] R. L. Panigrahi and M .K. Senapaty, "Real Time System for Software Engineering: An Overview", Global Journal for Research Analysis, Vol. 3, Issue 1, pp. 25-27, January 2014.

[2] Jane W.S. Liu, Real-Time Systems , Pearson Education, India,pp. 121 & 26, 2001.

[3] Mehrin Rouhifar and Reza Ravanmehr, "A Survey on Scheduling Approaches for Hard Real-Time Systems", International Journal of Computer Applications (0975 – 8887) Volume 131 -No.17, December 2015.

[4] C. Liu and James Leyland, January 1973, "Scheduling algorithm for multiprogramming in a hard real-time environment", Journal of the Association for Computing Machinery, 20(1): 46-61.

[5] J.Leung and J. Whitehead, "On the complexity of fixed-priority schedulings of periodic, real-time tasks", Performance Evaluation 2(4):237-250 December 1982.

[6] A.Mok and M.Dertouzos, 1978, "Multiprocessor scheduling in a hard real-time environment",7th Texas Conference on Computing Systems.

[7] Ketan Kotecha and Apurva Shah, "Adaptive Scheduling Algorithm for Real-Time Operating System", 978-1-4244-1823-7/08/$25.00 c_2008 IEEE

[8] J. Lehoczky, L. Sha and Yv Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior", IEEE International symposium on real time system, pp. 166-171, 1989.

[9] W. Li, K. Kavi, and R. Akl, "A non-preemptive scheduling algorithm for soft real-time systems", Computers and Electrical Engineering, vol.33, no. 1, pp. 12–29, 2007.

[10] Arezou Mohammadi and Selim G. Akl," Scheduling Algorithms for Real-Time Systems", Technical Report No. 2005-499, July 15, 2005.

[11] Zahereel Ishwar Abdul Khalib , Badlishah R. Ahmad and Ong Bi Lynn Ong, "High deadline meeting rate of non-preemptive dynamic soft real time scheduling algorithm",296301,DOI:10.1109/ICCSCE.2012.648715, 2012 IEEE.

[12] Li, Q. & Ba, W, "A group priority earliest deadline first scheduling algorithm", Frontiers of Computer Science October 2012, Volume 6, Issue 5, pp 560–567.