Improved Deadline Monotonic Scheduling With Dynamic and Intelligent Time Slice for Real-time Systems

H. S. Behera, Sushree Sangita Panda and Jana Chakraborty

Department of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, Orissa, India.
hsbehera_india@yahoo.com, sushreesangita@rocketmail.com , jana.chakraborty@gmail.com

ABSTRACT

In this paper the deadline-monotonic scheduling algorithm is improved to schedule processes in real-time systems. In real-time systems each task should have deadline that is greater than execution time and less than time period. Failure to meet the deadline in real-time systems degrades the system's performance. The proposed algorithm ensures that the processes meet their deadlines using iterative calculations using the exact schedulability test to determine which processes are schedulable. After finding the schedulable processes, they are scheduled using a suitable scheduling algorithm. Simple Round Robin scheduling algorithm shows high context switching and higher waiting time and response time. An improved-RR algorithm is proposed which calculates intelligent time slice for individual processes and taking dynamic time quantum into account. A comparative study is made to observe the interference due to higher priority processes and it was found that the proposed algorithm performs better than [1].

KEYWORDS

Real-time systems, Deadline monotonic scheduling (DM), Schedulability test, Round Robin scheduling algorithm Turnaround time and Waiting time

1. INTRODUCTION

A real-time system must produce the result in a specified time frame. If a process does not complete within the time frame it leads to system failure for hard real-time system. In case of soft real-time systems it does not lead to such disastrous results; however the system performance is degraded. Thus the main concern should be on the issue of meeting deadline of a process before its time period gets over. The development of appropriate scheduling algorithm has been isolated as one of the crucial challenges for the next generation of real-time systems [7]. Some of the applications of real-time systems are space research, weather forecast, seismic detection, audio and video conferencing, railway and flight reservation etc.

A real-time system is often considered as a finite collection of independent recurrent tasks, each of which generates a potentially finite sequence of jobs. Every job is characterised by an arrival time, execution requirement and deadline and a job completes execution between its arrival time and deadline. One of the scheduling method that is used is rate monotonic algorithm that assigns priorities based on their time periods, shorter the time period higher the priority. The constraint that rate monotonic priority assignment policy imposes on the process set is that it must be periodic, independent and have deadline equal to time period. Deadline monotonic priority assignment is a priority assignment policy used with fixed priority pre-emptive scheduling. With deadline monotonic priority assignment, tasks are assigned priorities according to their

DOI : 10.5121/ijitcs.2012.2202

deadlines; task with shortest deadline is assigned highest priority. Thus by devitalizing the constraint deadline = time period, we would provide a more flexible condition where deadline ≤ time period. In the task model each recurrent task τ_i is characterized by three parameters i.e. $\tau_i = (C_i, D_i, T_i)$, where

T_i= time period for sequence of successive jobs.
C_i= execution requirement and
D_i= deadline

By taking the given specifications for a set of tasks, the DM schedulability test is performed to determine whether the set is scheduled, such that all jobs complete by their deadline. This paper outlines deadline monotonic scheduling approach for a collection of processes. The schedulability test is performed to determine which processes in the process set are schedulable.

N.C. Audsley et.al [1] has proposed an algorithm which is valid only for hard real time systems whereas the proposed approach in this paper works for real-time systems. In addition to that in this paper all the schedulable processes were scheduled using improved RR algorithm with intelligent time slice and dynamic time quantum and the performance metrics were obtained and analysed.

A. RELATED WORK.

The priority assignment scheme that caters for processes with the relationship:
Computation time <= deadline <= time period was defined by Leung et.al [9]. It proposed an algorithm for deadline monotonic scheduling in which priority assigned to processes are inversely proportional to the length of their deadline. To generate a schedulability constraint for deadline monotonic scheduling the behaviour of processes released at a critical instant is fundamental if all processes are proved to meet their deadlines during executions beginning at a critical instant these processes will always meet their deadlines [8]. The deadline monotonic approach for hard real-time systems was proposed by N. C. Audsley et.al [1] and [3]. According to [6] fixed priority scheduling with deadline prior to completion for real-time systems is considered. With reference to [7] misconception about real time computing which is a serious problem for next generation computers was considered.

2. DEADLINE MONOTONIC SCHEDULING

The deadline monotonic scheduling algorithm is a priority driven scheduling algorithm that assigns priority to tasks according to their deadlines: the smaller the deadline greater the priority.

The schedulable processes should have the following relationship:

Computation time <= deadline <= time period

For the ith process (if we have n processes then process 1 has the highest priority and process n has the lowest priority in the system).

$C_i <= D_i <= T_i$

Deadline monotonic priority assignment is an optimal static priority scheme which implies that by using an algorithm deadline monotonic priority ordering for processes that will schedule the process set where process deadlines are unequal to their time period.

A simple DM schedulability test that has runtime linear in the number of tasks in the system was proposed by Liu and Layland, that proves any collection of tasks satisfies:

$$\sum_{i=1}^{n} U_i \leq n(2^{1/n} - 1)$$

Where the utilization U_i of process τ_i is given by:
$U_i = C_i$

2.1 SCHEDULABILITY TEST

The basis for schedulability test is that all processes are released simultaneously and check, the execution of all processes for a single execution. The schedulability test is given by:

$\forall i : 1 \leq i \leq n : C_i/D_i + I_i/D_i \leq 1$

Where I_i is the interference due to higher priority processes with the execution of τ_1 is given by:

$$\sum_{j=1}^{i-1} \lceil D_j/T_j \rceil C_j$$

The schedulability test suggests that for a process τ_i to be schedulable, the sum of its computation time and the interference that is imposed by higher priority processes should not be greater than D_i. This schedulability test is sufficient but not necessary.

In order to build a test which is sufficient and necessary the exact values of I_i are required. For this the schedule has to be evaluated in order to find out the exact interleaving of higher priority processes, which is costly as it require the solution of D_i equations per process τ_i.

I_i^d is the interference to process τ_i by higher priority process between the release of τ_i and time t, where t lies in the interval $[0,D_i]$.

Condition for the schedulability of process τ_i is

$$\frac{I_i^d}{d} + \frac{C_i}{d} \leq 1$$

Where

$$d = \sum_{j=1}^{1} C_j$$

$$I_i^d = \lceil d/T_j \rceil C_j$$

The above equations require a lot of D_i calculations. For n-process system maximum number of equations required is:

$$1 + \sum_{i=1}^{n} D_i$$

Since the value of t_i assumes that only one release of each process occurs in $(0,d_0)$ the constraint will fail if there had been any release of higher priority process within the interval $(0,d_0)$.

Therefore, the next point time at which τ_i may complete execution is:

$$d_1 = I_i^d + C_i$$

The schedulability is given by:

$$\frac{I_i^d}{d_1} + \frac{C_i}{d_1} < 1$$

Again the constraint will fail if releases have occurred in the interval (t_0 , d_1). Therefore we have to build a series of equations in order to prove the schedulability. The equations terminate if $t_k > D_i$ for process τ_i and equation k. So here τ_i is unschedulable.

2.2 PROPOSED APPROACH.

In the proposed algorithm deadline monotonic scheduling for a set of processes in a system using exact schedulability test is performed. Using schedulability for many processes if a process in a process set is unschedulable it remains unschedulable [1]. But by using the proposed approach unschedulable processes can also be scheduled by performing iterative calculations.

Finally all the schedulable processes are scheduled using the improved Round Robin scheduling algorithm with intelligent and dynamic time quantum. For the purpose of scheduling Intelligent Time Slice is calculated which allocates the frame for each task based on their priority and the time quantum is the lower of the two adjacent intelligent time slice. Let the original time slice (OTS) is the time slice to be given to any process. Priority Component (PC) is assigned depending on the priority which is inversely proportional to the priority number, process having highest priority is assigned 1 and rest 0. Shortness Component (SC) is the difference between the burst time of current process and the previous process, if the difference is less than 0, then SC=1 else 0. For calculating Context Switch Component, first PC, SC and OTS are added and then subtracted from the burst time, if difference is less than OTS, then it will be considered as CSC. Finally ITS=OTS + PC + SC + CSC.

A. *Pseudo code (conjunction of deadline monotonic scheduling with priority based RR scheduling).*

Let n: no. of processes.
C_j: execution time for j^{th} process.
I_i^d: interference on process j by i higher priority processes at time t.
C_i: execution time for i^{th} process.
D_i: deadline for process i.
d: summation of execution time of all processes till process j.
ITS: Intelligent Time Slice.
TS: Original Time Slice.
PN: priority number.
PC: priority component.
SC: shortness component.
CSC: context switch component.
int PN= 1…N

$$d_0 = \sum_{j=1}^{i} C_j$$

Initialize: C_i= 0, D_i=0, T_i=0
foreach τ_i do

$$d = \sum_{j=1}^{1} C_j$$

value= TRUE
while (value) do
if($\frac{I_i^d}{d_1} + \frac{C_i}{d_1} \leq 1$)
value = FALSE
the process τi is schedulable
else
d_1 is calculated using:
$d_1 = I_i^d + C_i$
endif
if (d > Di)
exit
the process τ_i is unschedulable
endif
endwhile

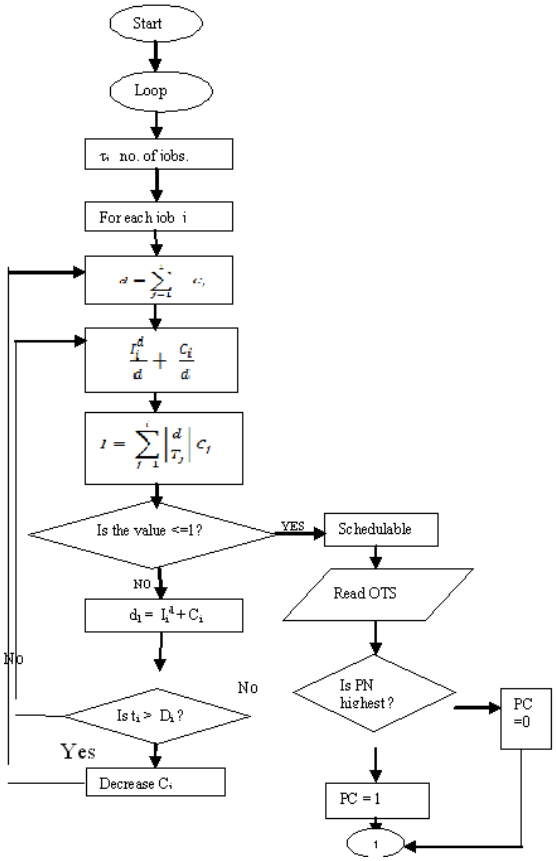
if(burst time of current process-burst time of previous
process < 0)
SC=1;
else SC=0;
endif
if(burst time - (PC + SC + OTS) < 0)
CSC=1;
else CSC=0;
endif
ITS = OTS + PC + SC + CSC
Check two adjacent jobs. The one which has the
lower ITS
let, L=job with lower ITS. TQ=L

$$A_{TAT} = \sum_{i=1}^{N} \frac{TAT\ of\ all\ the\ processes}{no.of\ processes}$$

$$A_{WT} = \sum_{i=1}^{N} \frac{WT\ of\ all\ the\ processes}{no.of\ processes}$$

endfor

Fig-1: pseudo code for schedulability of many processes in conjunction with improved Round Robin algorithm with dynamic and Intelligent Time Slice



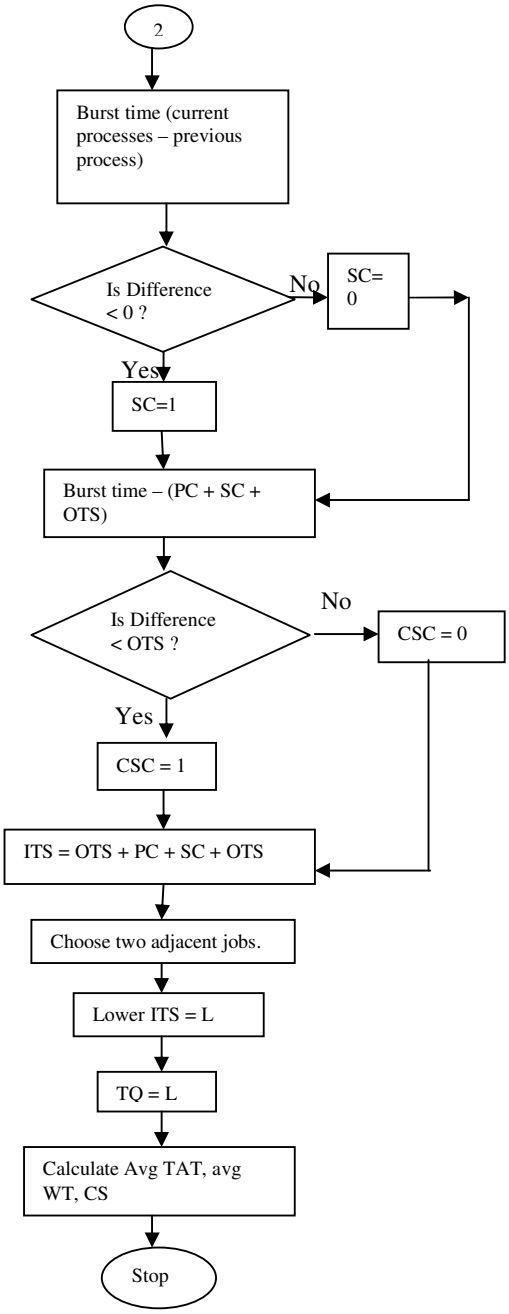


Fig-I Flowchart for the proposed algorithm

2.3 ILLUSTRATION

7 jobs J_1, J_2, J_3, J_4, J_5, J_6, J_7 are present in a process set along with C_i computation time 1, 2, 2, 2, 4, 3, 5, time period T_i 6, 10, 14, 18, 25, 28, 37 and deadline D_i 5, 8, 11, 15, 20, 27,35, respectively in a system. Then for τ_1 , t_0 is calculated and I_i^d is evaluated , and τ_1 is schedulable. If I_i^d is greater than 1 then iterations are performed, here $d_1 = I_i^d + C_i$ and the value of t_1 is put in the equation $I_i^d/d + C_i/d$ till the value is less than or equal to 1. After all the

jobs are found to be schedulable the jobs are put in a ready queue and taking a time quantum and the arrival time 0 the Round Robin scheduling is performed with Intelligent Time Slice. The burst sequence and user priority are given as 18,24,25,30,36,43,45 and 1,2,1,3,4,1,5 respectively. Original time slice is taken as 10. The PC, SC, CSC is calculated according to the algorithm. Two adjacent processes are then taken and the lower of the two ITS is taken as the time quantum for those two processes.

2.4 EXPERIMENTAL ANALYSIS

A.Assumptions

All the processes were processed in the real-time systems with single processor environment and all the processes are independent. The time period is more than the deadline and deadline is greater than the computation time. All the attributes like burst time, numbers of processes, priority, Intelligent Time Slice are known before submitting the processes to the processor. The arrival time is assumed to be 0. All the processes are CPU bound.

B.Experimental Frame Work

The experiment consists of several input and output parameters. The input parameters consist of deadline, computation time, time period, burst time, time quantum, priority and number of processes. The output parameters consist of interference on a process by other higher priority processes, average waiting time, average turnaround time and number of context switches.

C.Data Set

Several experiments have been performed in order to evaluate the performance of the proposed algorithm. Particular set of processes are taken to examine schedulability. The data set are the processes with burst time increasing, decreasing and random to perform scheduling.

D.Performance Metrics

For the experimental analysis the significance of performance metrics is as follows:

1. Interference: for the better performance of the algorithm, interference due to higher priority processes should be less.
2. Turnaround time(TAT):for the better performance of the algorithm, average turnaround time should be less.
3. Waiting time(WT): for the better performance of the algorithm, average waiting time should be less.
4. Number of Context Switches (CS): for the better performance of the algorithm, number of context switches should be less.

E.Experiments Performed

To evaluate the performance of the proposed algorithm, a set of 7 processes was taken for simplicity. The algorithm works effectively even if it is used with a very large number of processes. In this the schedulability for many processes is compared with exact schedulability test. After finding the jobs that are schedulable Round Robin scheduling algorithm was applied where the arrival time of processes is 0 and time quantum is the lower of the Intelligent Time Slice for two adjacent processes.

Case 1: 7 processes are taken and their schedulability is determined by using deadline monotonic scheduling.

Jobs	C_i	T_i	Di
J_1	1	6	5
J_2	2	10	8

J_3	2	14	11
J_4	2	18	15
J_5	4	25	20
J_6	3	28	27
J_7	5	37	35

Table 1: it gives the values of computation time, time period and deadlines for a set of 7 processes.

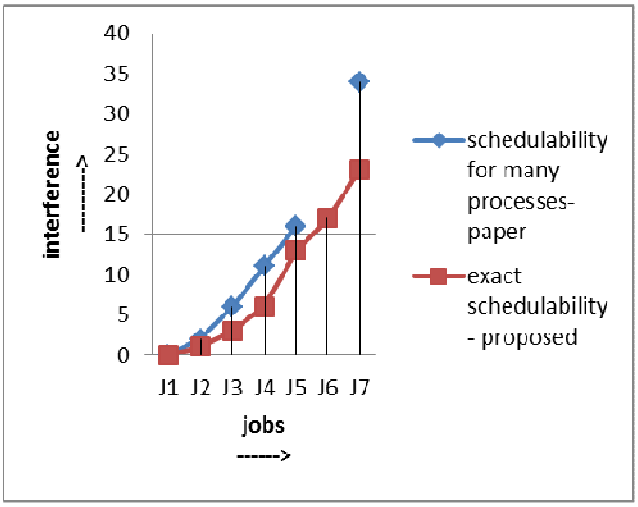


Fig-2: Interference plot for jobs in the process set to distinguish between schedulability for many processes and exact schedulability test.

After executing the proposed algorithm all the processes are found schedulable. But when schedulability for many processes was applied only jobs J_1, J_2, J_3, J_4, J_5 and J_7 are schedulable and J_6 are unschedulable.

Case 2: 7 processes are taken with arrival time= 0, and increasing burst time and priority.

JOBS	BURST TIME	PRIORITY NO.	PC	SC	CSC	ITS
J_1	18	1	1	0	1	12
J_2	24	2	0	0	0	10
J_3	25	1	1	0	0	11
J_4	30	3	0	0	0	10
J_5	36	4	0	0	0	10
J_6	43	1	1	0	0	11
J_7	45	5	0	0	0	10

Table 2(a): it is the output using the proposed scheduling algorithm for increasing burst time.

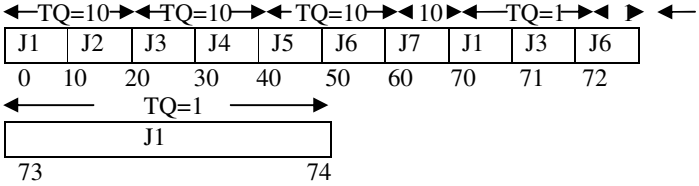


Fig-3: Gantt chart for algorithm as proposed in table 2(a).

ALGORITHM	AVG TAT.	AVG WT	CS
Proposed Algorithm.	57	30	11

Table 2(b): performance metrics for increasing burst time.

Case 3: 7 processes are taken with arrival time=0, and decreasing burst time and priority.

JOBS	BURST TIME	PRIORITY NO.	PC	SC	CSC	ITS
J_1	45	5	0	0	0	10
J_2	43	1	1	1	0	12
J_3	36	4	0	1	0	11
J_4	30	3	0	1	0	11
J_5	25	1	1	1	0	12
J_6	24	2	0	1	0	11
J_7	18	1	1	1	1	13

Table 3(a):output using the proposed scheduling algorithm for decreasing burst time.

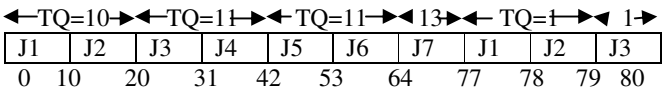


Fig-4: Gantt chart for algorithm as proposed in table 3(a).

ALGORITHM	AVG TAT.	AVG WT	CS
Proposed Algorithm.	54.71	31.4	10

Table 3(b): performance metrics for decreasing burst time.

Case 4: 7 processes are taken with arrival time=0, and random burst time and priority.

JOBS	BURST TIME	PRIORITY NO.	PC	SC	CSC	ITS
J1	30	3	0	0	0	10
J2	25	1	1	1	0	12
J3	36	4	0	0	0	10
J4	43	1	1	0	0	11
J5	45	5	0	0	0	10
J6	18	1	1	1	1	13
J7	24	2	0	0	0	10

Table 4(a): it is the output using the proposed scheduling algorithm for random burst time.

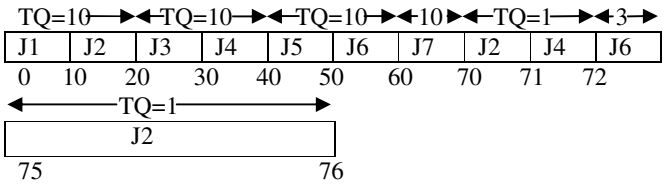


Fig-5: Gantt chart for algorithm as proposed in table 4(a).

ALGORITHM	AVG TAT.	AVG WT	CS
Proposed Algorithm.	54.17	30	11

Table 4(b): performance metrics for random burst time.

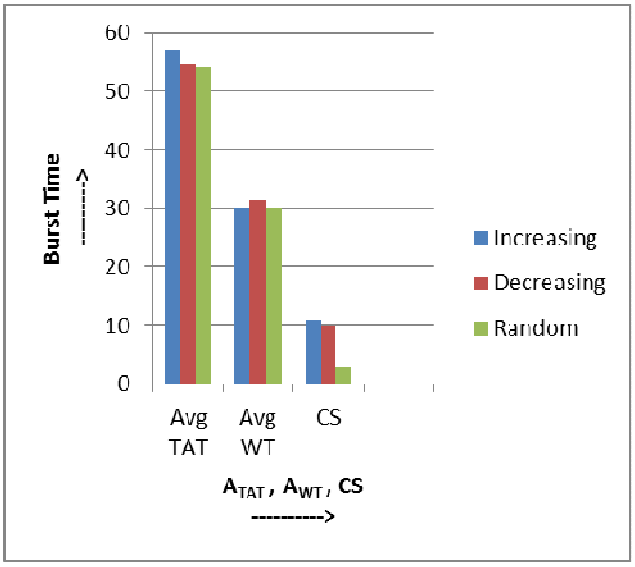


Fig-6: Average turnaround time, Average waiting time and context switch time of the proposed algorithm for increasing, decreasing and random burst time.

3. CONCLUSION AND FUTURE WORK.

A comparative study was made between the interference of processes using exact schedulability and schedulability for many processes and we found that the interference of processes due to higher priority processes is less in case of exact schedulability test. Moreover a process J_6 is found to be unschedulable using schedulability for many processes whereas using exact schedulability all the processes were found to be schedulable.

In this paper, schedulability tests have been presented for deadline-monotonic scheduling algorithm, and schedulable processes were scheduled using Round Robin algorithm. All the processes were assumed to have a critical instant. This is ensured as all the processes have an initial release at time 0. When execution of higher priority processes does not overlap, the deadline of τ_i, then I_i will be exact. However when executions of higher priority processes do overlap, this test does not pass many processes.

An exact value of I_i must therefore be known, so that exact interleaving of higher priority processes is known. The proposed approach provides schedulability test using exact schedulability test, where the complexity is related to the periods and computation time of processes in the system. Finally the round robin scheduling algorithm was applied to schedulable processes.

This algorithm can be used to see the effect upon system utilization. Further this algorithm can be investigated to prove more and more useful for task oriented results. Besides, some research work could be done to synchronize and vary the timing characteristics of the processes in the process set.

4. REFERENCES.

[1]. N.C. Audsley, A. Burns, M.F. Richardson, A.J. Wellings "Hard Real-Time Scheduling: The Deadline-Monotonic Approach" Dept of Computer Science, University of York, York, YO1 5DD, England.

[2]. Yaashuwanth .C & R. Ramesh "Intelligent Time Slice for Round Robin in Real Time Operating Systems" Dept of electrical and electronics engineering, Anna University Chennai, Chennai 600 025

[3]. Audsley, N.C. (1990), Deadline Monotonic Scheduling, YCS 146, *Dept. of Comp. Sci., Univ. of York.*

[4]. Barbara Korousic –Seljak (1994) "Task scheduling policies for real- time Systems" Journal on MICROPROCESSOR AND MICROSYSTEMS, VOL 18 NO. 9, pg501-512.

[5]. Bur89a. A. Burns and A. J. Wellings, Real time Systems And Their Programming Languages, Addison Wesley (1989).

[6]. Burns. A (1994) "Fixed priority scheduling with deadline prior to completion" Real time systems Research Group Department of computer science university of York, UK.

[7]. Sta88a. J. A. Stankovic, Misconceptions About Real Time Computing: A serious problem for next generation systems, IEEE Computer21 (10), pp. 10-19(Oct 1988).

[8]. Liu, C.L. and J. W. Layland (1973) scheduling algorithms for multiprogramming in a hard real time environment

J. ACM, 20, pp.40-61

[9]. Leung, J., and Whitehead, J. On the complexity of fixed priority scheduling of periodic, real time tasks.

Authors' Biodata

1. Dr. H. S. Behera is currently working as a faculty in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India. His areas of interest include Distributed Systems, Data Mining and Soft Computing.

2. Sushree Sangita Panda is a final year B.Tech student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India.

3. Jana Chakraborty is a final year B.Tech student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India.