Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 6, June -2015

An Improved LLF_DM Scheduling Algorithm for Periodic Tasks by Reducing Context Switches

Mitul Patel[1], Bhavesh Oza[2]

[1]Computer Engineering Department, L.D. College of Engineering
[2]Computer Engineering Department, L.D. College of Engineering

Abstract *—To achieve successful completion of a job before its deadline is the most challenging part of scheduling in real time systems. There are two types of priority scheduling algorithms which are commonly used in real time system. One is fixed priority scheduling algorithms and the other is dynamic priority scheduling algorithms.Dynamic priority scheduling algorithm LLFachieves optimum result in under-loaded condition but performs poor in over-loaded condition of the system. Whereas fixed priority algorithm DM performs well in over-loaded condition but gives poor result in under-loaded condition of the system. The LLF_DM achieves optimum performance in under-loaded as well as in over-loaded condition but result in more number of context switches. Our proposed Improved LLF_DM algorithm reduces the number of context switches.*

Keywords- *LLF, DM, ILLF, Scheduling Algorithm, Context Switch, Real Time Operating System*

I. INTRODUCTION

Today is the era of the computer science and technology. Embedded components are highly integrated in the day to day life used systems. Embedded components are one kind of real time systems. Real time systems have a wide area of the application like automation industry, E-commerce, medical applications, telecommunication, manufacturing etc. So Real-Time systems cover a large part of computer industry.

There are mainly two types of real-time operating systems. First is hard and the second one is soft.All jobs must be executed within their deadlines in the hard real-time systems. Missing the deadlines of a job or some jobs may cause a failure of the system. So they are strict in the nature. Power plant controller and avionic devices etc. are the popular examples of this type.A hard real-time system guarantees that absolute deadlines must be met. e.g. if in a nuclear plant some fault occurs, then we must take certain actions quickly otherwise it may become a dangerous situation. On the other side, missing the deadlines of a job or some jobs is allowed up to certain level in the soft real-time system and it causes degradation in the performance of the system. Sothey are less strict in the nature. Multimedia application and cellphones etc. are examples of this type.It is tolerable but undesirable that a job miss the deadline e.g. in multimedia data from CD drive should be read and converted into music in fixed period of time, but occasionally missing deadline cannot be considered as intolerable.

II. THE SCHEDULING ALGORITHMS

2.1. The DM Scheduling Algorithm

The deadline monotonic scheduling algorithm is a static task priority algorithm [1]. It is also preemptive in nature. DM gives the higher priority to the task with the smaller deadline. If more than one tasks have the same deadline, then the processor is randomly assigned to one of them by the DM algorithm [9].DM becomes equivalent to the RM algorithm when the deadlines of tasks are equal to their period [10]. Scheduling in DM algorithm referred is shown in Figure 1 for three processes P0(2,8), P1(6,12) and P2(3,12) in a task system.

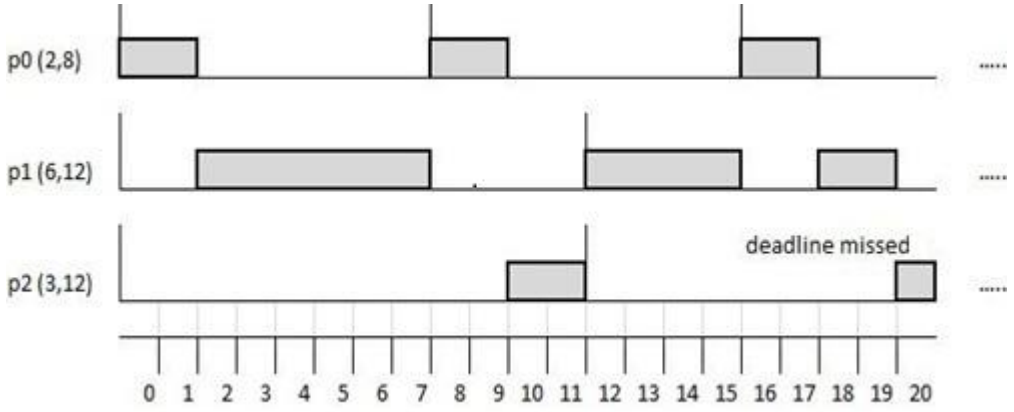

Figure 1. DM scheduling

2.2. The LLF Scheduling Algorithm

The least laxity first algorithm is a dynamic preemptive scheduling algorithm [1]. The name gives the idea for the priority assignment of the task that the highest priority is assigned to the task having the smallest laxity. The laxity l is defined at time t with the deadline interval d and remaining execution time c.

$$l = d - t - c$$

The LLF is an optimal algorithm for the single processor systems [5] [8]. The disadvantage of the LLF algorithm is the more number of context switches. The LLF requires the exact execution time of the task which is difficult to know before the completion of task. This is also a disadvantage of this algorithm. Scheduling in Least Laxity First (LLF) algorithm referred is shown in Figure 2 for three processes P0(2,8), P1(6,12) and P2(3,12) of a task system.

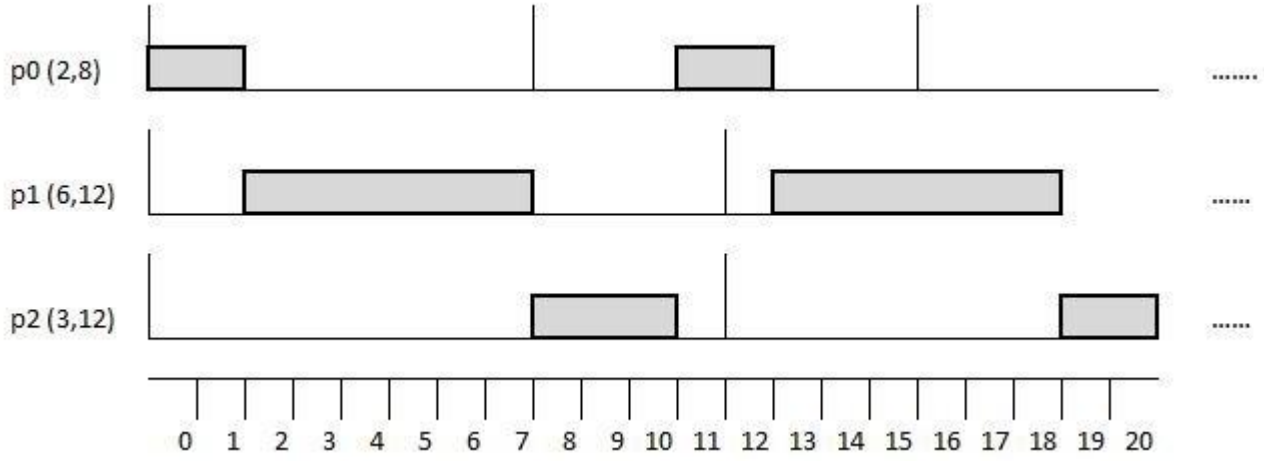

Figure 2. LLF scheduling

2.3. The ILLF Scheduling Algorithm

Scheduling algorithms play an important role in design of real time systems.Least Laxity First (LLF) is a well-known and extensively applied dynamic Scheduling algorithm which has been proved to be optimal onuniprocessor systems. The Least-Laxity-First (LLF) Scheduling algorithm assigns priority based on the slacktime of a process.Thealgorithm is impractical to implement because laxity tie results in the frequent context switches among the tasks [5].

The Improved Least Laxity First Scheduling Algorithm with intelligence time slice finds the timequantum by taking the greatest common divisor(GCD) of all the execution time of the processes. After everyunit of time slice the laxity of each remaining process(present in the ready queue) is calculate. The loop iscontinued until all the processes are being executed by the processor. Here as the GCD of execution the executiontime is always greater than equal to 1 so loop will be continue for lesser no of time or same no of time .i.e. whenthe GCD of the execution time is greater than 1 in that case scheduling algorithm will show betterperformance(loop will continue less no of times). The ILLF algorithm shows less context switching ascompared to MLLF scheduling Algorithm [3].

2.4. The LLF_DM Scheduling Algorithm

The LLF algorithm is performing well in the under loaded system. It is not able to achieve good performance in over loaded situation, whereas the DM acts vice a versa. That means achieve good result in the over loaded situation of the system and average performance in under loaded system.

The LLF_DM is mixture of the two algorithms: LLF and DM. The LLF_DM acts as the LLF in system having a low load and as the DM in system having a high load in the system [6]. This way it takes the advantage of both algorithms. It performs better in both the situation of system.

2.5. The Improved LLF_DM Scheduling Algorithm

As we have seen in the LLF_DM algorithm perform equally to the LLF in the under loaded state of the system and perform better than LLF in over loaded state. They consider only two parameters for performance comparison: SR and EPU. They do not count the number of context switches. We consider one more parameter NCS (Number of Context Switches) for performance comparison.

The ILLF algorithm produces the less NCSs than the MLLF and the LLF. The ILLF considers GCD of execution times of all processes in the system as an intelligent time slice for calculating laxities of ready tasks.Our proposed Improved LLF_DM algorithm switches to the ILLF instead of LLF in under-loaded condition for reducing the number of context switches. The Improved LLF_DM works same as the LLF_DM algorithm in the over-loaded condition.

III. SYSTEM AND TASK MODEL

We call each unit of work that is scheduled and executed by the system as a job and set of related jobs, which jointly provides some system function is a task [11]. All tasks are assumed to be periodic. The system knows about the deadline and required computation time of the task when the task is released.

There are no precedence constraints on the task; they can run in any order relative to each other as long as their deadlines are met. A task is ready to execute as it arrives in the system. We consider that the system is not having resource contention problem. The task set is assumed to be preemptive. Preemption and scheduling incur no overhead.

IV. SIMULATION METHOD

We have implemented the LLF, the LLF_DM and the improved LLF_DM algorithms in the same environment and have run simulations to accumulate empirical data.Periodic tasks have been considered for taking results. For periodic tasks, load of the system can be defined as summation of ratio of executable time and period of each task [6]. We have generated 50 task sets for 20 load values from 0.5 to 2.0. Each task set is having 3 periodic tasks. Each task set is simulated for 500 clock cycles.

The system is said to be overloaded when the tasks offered to the scheduler cannot be feasibly scheduled even by a clairvoyant scheduler A reasonable way to measure the performance of a scheduling algorithm during an overload is by the amount of work the scheduler can feasibly schedule according to the algorithm [2].We consider three different parameters to compare the performance of required scheduling algorithms.

4.1. Success Ratio (SR)

To achieve the deadline is very important for a job in RTS. The number of jobs achieving their deadline to the total number of jobs in the system is called SR (Success Ratio): [6] [7]

$$SR = \frac{Number\ of\ jobs\ successfully\ \ executed}{\text{Total number of jobs arrived}}$$

4.2. Effective Processor Utilization (EPU)

It gives information about how efficiently the processor is used and it is defined as: [4] [6]

$$EPU = \sum_{i \in R} \frac{Vi}{T}$$

Where V is value of a job is equal to computation time of job if it achieves the deadline otherwise zero. R is set of successfully executed jobs. T is the total time for the scheduling.

4.3. Number of Context Switches (NCS)

This parameter gives the number of context switches for the scheduling algorithm.

V. FINAL RESULT

We have taken results for LLF, LLF_DM and improved LLF_DM (ILLF_DM) algorithms for different load conditions. Load of task sets ranges from 0.5 to 2.0. Table 1 shows results in terms of SR, EPU and NCS for every algorithm in different load conditions.

Table 1. Load vs. SR, EPU, NCS for LLF, LLF_ DM and ILLF_ DM

	SR in %			EPU in %			NCS		
Load	**LLF**	**LLF_DM**	**ILLF_DM**	**LLF**	**LLF_DM**	**ILLF_DM**	**LLF**	**LLF_DM**	**ILLF_DM**
0.5	100	100	100	**50**	**50**	**50**	181	194	194
0.6	100	100	95.4	**60**	**60**	**57.1**	197	242	204
0.7	100	100	99.2	**70**	**70**	**69.5**	226	254	246
0.8	100	100	97	**79.9**	**79.9**	**77.5**	240	256	243
0.85	100	100	97.5	**84.9**	**84.9**	**82.7**	258	286	279
0.9	100	100	94.9	**89.9**	**89.9**	**85**	292	317	299
0.95	100	100	99.6	**94.8**	**94.8**	**94.4**	314	341	323
1	100	100	99.1	**99.7**	**99.7**	**98.9**	285	348	308
1.05	16.7	83.7	83.5	**16**	**73.6**	**73.4**	254	302	299
1.1	8.7	81.4	80.9	**8.7**	**73.4**	**72.8**	239	278	270
1.15	5.4	83.7	83.6	**5.5**	**76.7**	**76.6**	223	246	239
1.2	4.6	70.1	69.9	**4.6**	**57.8**	**57.5**	262	286	282
1.3	3.2	74.5	74.1	**3.2**	**63.5**	**62.9**	245	279	275
1.4	2.3	70.5	69.6	**2.1**	**59.3**	**58.34**	258	289	283
1.5	1.9	72	71.9	**1.6**	**56.5**	**56.4**	279	337	333
1.6	1.3	65.5	65.5	**1.3**	**53.8**	**53.8**	249	249	246
1.7	1.2	50.2	49.5	**1.2**	**43.7**	**43**	226	191	186
1.8	1.2	62.8	62.6	**1.1**	**52.5**	**52.2**	262	266	262
1.9	1	61.1	61.1	**0.8**	**49.9**	**49.9**	255	251	247
2	0.8	47.9	47.2	**0.8**	**38.1**	**37.4**	240	190	184

Figure 3 shows comparison of the results of Success Ratio (SR) for LLF, LLF_DM and ILLF_DM. The results are taken from under loaded condition (load value=0.50) to over loaded condition (load value=2.0).

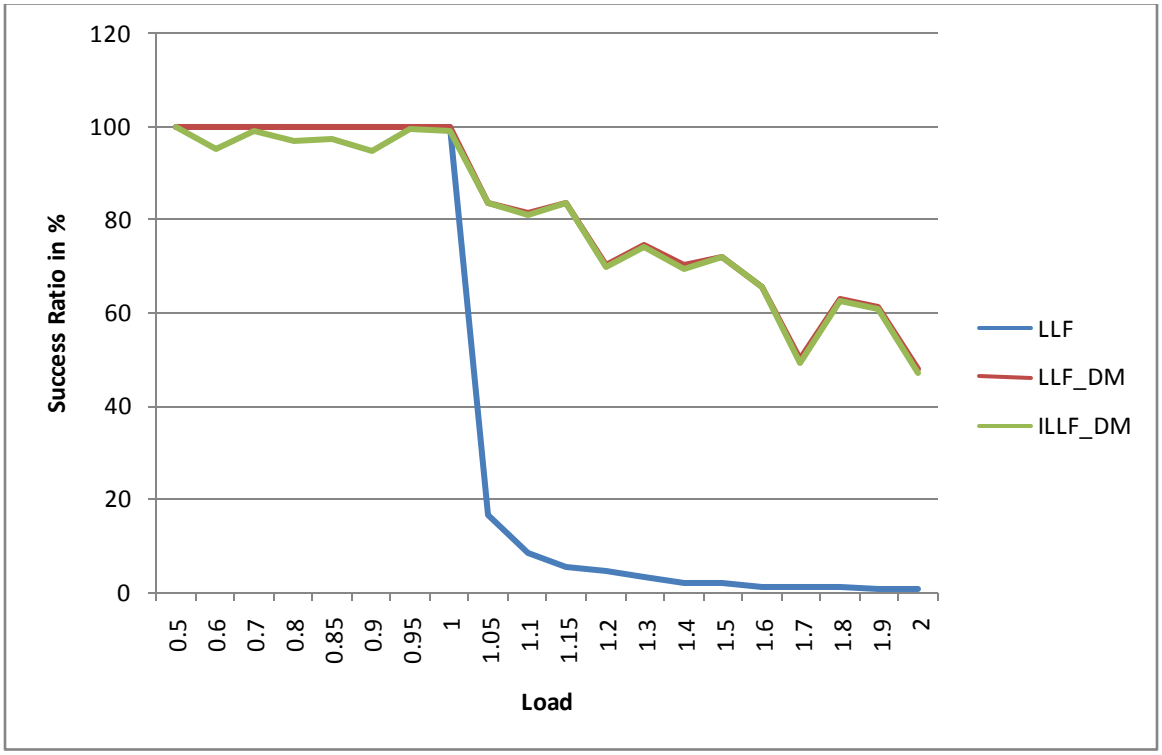

Figure 3. Load vs. SR in % for LLF, LLF_DM and ILLF_DM

Figure 4 (on next page) shows comparison of the results of Effective Processor Utilization (EPU) for LLF, LLF_DM and ILLF_DM. The results are taken from under loaded condition (load value=0.50) to over loaded condition (load value=2.0).

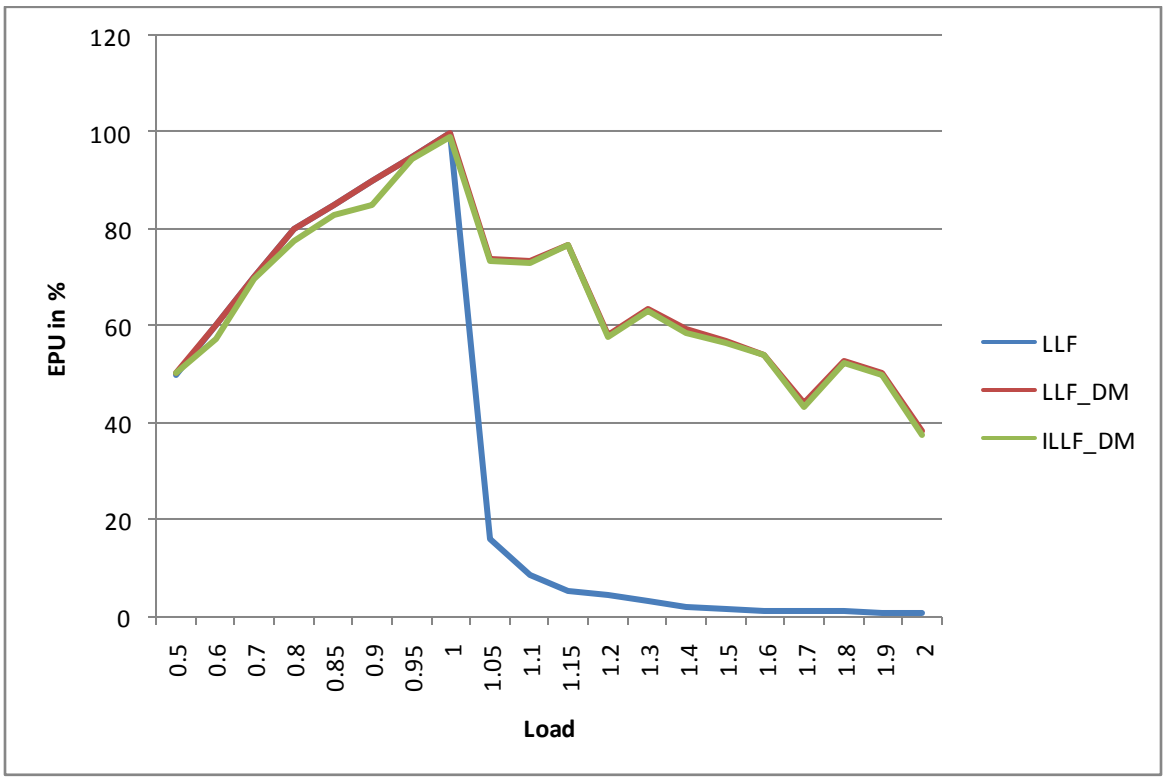

Figure 4. Load vs. EPU in % for LLF, LLF_DM and ILLF_DM

Figure 5 shows comparison of the results of Number of Context Switches (NCS) for LLF, LLF_DM and ILLF_DM. The results are taken from under loaded condition (load value=0.50) to over loaded condition (load value=2.0).

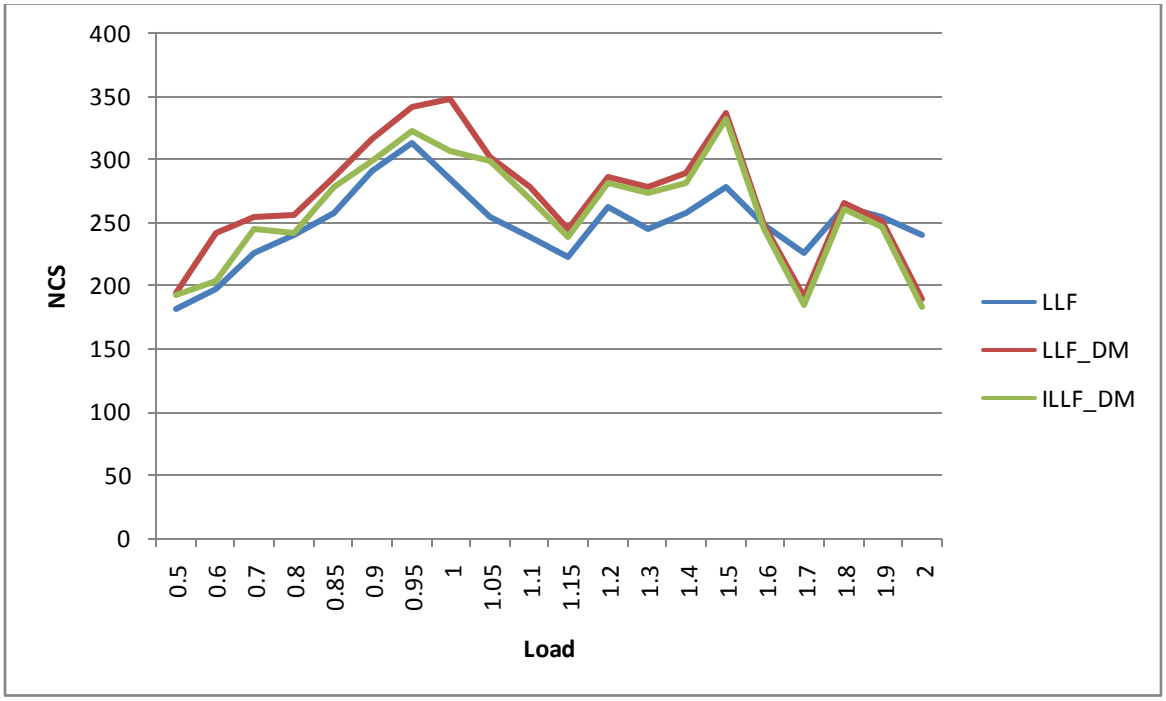

Figure 5. Load vs. NCS for LLF, LLF_DM and ILLF_DM

VI. CONCLUSION

The LLF algorithm achieves optimum results in under-loaded condition but fail to achieve the same in over-loaded condition. The LLF_DM performs better than dynamic priority scheduling algorithm LLF in the context of Success Ratio and Effective Processor Utilization But gives more number of context switches than the LLF algorithm.

The ILLF_DM algorithm is a variant of the LLF_DM algorithm with Greatest Common Divisor (GCD) as intelligent time slice for determining laxities of ready tasks. The LLF_DM performs approximately equally to the LLF in the context of Success Ratio and Effective Processor Utilization as shown in Figure 3 and Figure 4. The ILLF_DM algorithm reduces number of context switches than the LLF_DM algorithm in under loaded condition as shown in Figure 5. The LLF_DM and the ILLF_DM give approximately equally number of context switches in over loaded condition as shown in figure 5.

REFERENCES

[1] Jane W.S. Liu, Real-Time Systems, Pearson Education, India, 2013.
[2] A M Shah, Dynamic Scheduling for Real-Time Operating Systems, Ph. D. Thesis, Information TechnologyDepartment, Sardar Patel University, India,2010
[3] H. S. Behera, SatyajitKhuntia, SoumyashreeNayak, “An improved least-laxity-first scheduling algorithm for real-time tasks”, International Journal of Engineering Science and Technology (IJEST), Vol. 4, No. 4, April 2012, pp.1312-1319.
[4] Ketan Kotecha, Apurva Shah, “Efficient Dynamic Scheduling Algorithms for Real-Time Multiprocessor Systems”, International conference on High Performance Computing, Networking and Communication System-2008
[5] Sung-Heun Oh Seung-Min Yang, “A Modified Least-Laxity-First scheduling algorithm for real-time tasks,” IEEE International conference on real time Computing systems and application, Oct 1998.

[6] V.Prajapati, A.Shah, P.Balani, "Design of new scheduling algorithm LLF_DM and its comparison with Existing EDF, LLF, and DM algorithms for periodic tasks", International Conference on Intelligent Systems and Signal Processing (ISSP)-2013.

[7] Ramamritham K., Stankovik J. A., Shiah P. F. ,"Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems, IEEE Transaction on Parallel and Distributed Systems", Vol 1(2), pp. 184-194, 1990.

[8] A.K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology ,Cambridge, Massachusetts, May 1983.

[9] Albert M. K., Real-Time Systems Scheduling, Analysis, and Verification, CHENG, University of Houston

[10] Nasro Min-Allah, Hameed Hussain, Samee Ullah Khan and Albert Y. Zomaya, "Power efficient rate monotonic scheduling for multi-core Systems," Journal of Parallel Distributed Computing, vol. 72, issue 1, January 2012.

[11] Liu C. L., Layland L., "Scheduling algorithms for multiprogramming in a hard-realtime environment", Journal of ACM, Vol 20(1), pp. 46-61, 1973

