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

Enhancement in LLF Scheduling Algorithm: E_LLF Scheduling Algorithm for Real-Time Systems

Pooja Patel[1], Prem Balani[2], Vishal Prajapati[3]

[1]*M. E. Information Technology, G H Patel College of Engg. & Tech., Vallabh Vidyanagar, Gujarat, India.*
[2,3]*Assistant Professor, Department of IT, G H Patel College of Engg. & Tech., Vallabh Vidyanagar, Gujarat, India*

Abstract *- Scheduling Algorithm is used to schedule set of tasks. Basically, Two types of scheduling algorithm exist. One is Static scheduling algorithm which assigns all priorities at design time, and it remains constant for the lifetime of a task. Another is Dynamic Scheduling Algorithm. It assigns priority at runtime, based on execution parameters of tasks which can change its priority during its execution. Least Laxity First (LLF) and Earlier Deadline First (EDF) are two most common Dynamic scheduling algorithm used in Real-Time Systems. LLF algorithm schedule task which has least laxity or slack time. LLF algorithm gives optimum result in under loaded condition. But, Performance of LLF algorithm decrease drastically in overloaded condition.*

In this paper, we have proposed Enhanced Least Laxity First (E_LLF) scheduling algorithm which works same as LLF scheduling algorithm in under loaded condition. But, In overloaded condition algorithm deletes those jobs which have already missed their deadline (or expected to miss its deadline) or ignore job with maximum execution time based on priority criteria so that other jobs have chance to complete their execution on processor.

Keywords- *Scheduling Algorithm; LLF; Laxity; Slack;*

I. INTRODUCTION

Real-Time Systems: A system is called real-time system, when we need quantitative expression of time (real-time) to describe the behavior or nature of the system [5]. A real-time system is a system where the correctness of the system behavior depends not only on the logical results or finaloutcomes of the computations, but also on the physical time when these results are produced [3][10].

This algorithm assigns priority based on Laxity (or Slack-Time): The smaller the Laxity (Slack-Time), higher the priority. At any time t, the Slack (or Laxity) of a job with deadline d is equal to d-t minus the time required to complete the remaining portion of job. [4]

Thus, Slack (or Laxity) Time = d-t-e(t)

The laxity is the maximum amount of time a job may be forced to wait if it was to execute on a processor and still completes its execution within deadline. Thus, The laxity of a task is the maximum time the task or job can delay execution without missing its deadline in the future.[1]

This LLF scheduling algorithm also known as Least Slack Time (LST) First Scheduling Algorithm or Minimum-Laxity-First (MLF) Scheduling Algorithm.

LLF is a little more general than EDF because it takes into account laxity time (or Slack), which is more meaningful than only considering deadline as in EDF for tasks of mixed computing sizes. In addition, LLF offers probably more graceful degradations. [7] If system is preemptive and under loaded, LLF (Least Laxity First) scheduling algorithm has been proved to be optimal algorithm for single processor system. But, limitation of any Dynamic scheduling algorithm is that its performance decreases drastically when system becomes slightly overloaded.

II. SYSTEM AND TASK MODEL

All tasks are periodic. Each task having Arrival Time (A), Period (P), Deadline (D), Execution Time (E) and Priority(PR). These are five parameters we have considered in our algorithm. Here After each Period time new job will be added of same type. Thus, Task generated at each multiple of period Pi. Static priority 0 and 1 has been given to each task. These priorities only used in overloading condition. The task with static priority 1 is more important than task with static priority 0.

III. PROPOSED SCHEDULING ALGORITHM : E_LLF SCHEDULING ALGORITHM

If any task going to miss its deadline means it's in overloading condition. It will come out of overloading condition when any one job going to completes its execution successfully within deadline.

Algorithm / Flow:

1. Find out Least Laxity Job (Least_Laxity_Job).

2. Find out Maximum Execution Time containing job (Max_Exe_Time_Job).

3. Check for system overloading

    Is System Overloaded?

    If No, Return Least_Laxity_Job for execution. If Yes, (System is in overloading condition.)

        Check Remaining Exe_time (Least_Laxity_Job) > Remaining time in Deadline?

        If No, Check for other condition. If Yes,

            Delete that Least_Laxity_Job as that job has already missed the deadline or expected to miss its deadline.

            Find out another Least_Laxity_Job for execution and return it.

        Otherwise, check Is Max_Exe_Time_Job = Least_Laxity_Job? And Is priority (Least_Laxity_Job) = 0 ?

        If No, Return Least_Laxity_Job for execution. If Yes,

            Ignore the Max_Exe_Time_Job in scheduling.

            Find out another Least_Laxity_Job and return it.

IV. SWITCHING CRITERIA

Initially the proposed algorithm uses LLF algorithm considering that the system is in under loaded condition. But when one job missed the deadline, it will be identified as overloaded condition. When system is in overloaded condition, E_LLF scheduling Algorithm first of all find out job with least laxity and job with maximum execution time. After that If least laxity job has already missed the deadline or their expected remaining execution time is more than the remaining time in meeting deadline then that job will be discarded and another least laxity job will be in execution. Otherwise (that least laxity job is not expected to missed its deadline then) check least laxity job and maximum execution time containing job are same or not. If they are same, then check for priority of that least laxity job. If it is 0 then ignore that maximum laxity containing job in scheduling and execute another job with least laxity time so that at least other jobs with less execution time have chance to complete their execution. If static priority is 1 then that task is important and we can't ignore it in the scheduling.

Thus, In this case, previously found least laxity job will be in execution.

After one job has completed its execution successfully, again the algorithm will act like LLF algorithm considering that overloaded condition has been disappeared and now system is in under loaded condition.

During under loaded condition, LLF algorithm is used for getting optimum result and during overloaded condition algorithm discard that job which has no chances to meet its deadline or ignore maximum execution time job in scheduling based on some static priority 0 and 1 so that at least other jobs have chance to complete their execution successfully. Thus, It is used for achieving better performance.

V. SIMULATION

Performance Measures:

Load (L) of the system with periodic tasks can be determined using following equation.[2]

$$L = \sum_{i=1}^{m} \frac{Ei}{Qi}$$

Where,

m = Number of tasks

E = Execution time required by the task P = Period of the task

D = Deadline of the task Q = P if P>= D

Q = D if P < D

The system is identified to be overloaded when the tasks offered to the scheduler cannot be feasibly scheduled even by a clairvoyant scheduler. For periodic tasks, according to above equation, the system can be considered as overloaded system when its load value is greater than 1.00. An appropriate way to measure the performance of a dynamic scheduling algorithm during an overloaded condition is by the amount of work the scheduler can feasibly schedule according to the algorithm and tasks complete their execution successfully within time limit. Therefore, Success Ratio (SR) and Effective Processor Utilization (EPU) are main performance measures and they defined as:

In real-time systems, deadline meeting is the most important. Therefore, the most appropriate performance metric is the Success Ratio and it defined as [1][2][3][6][8][9],

$$SR = \frac{\text{Number of jobs successfully Completed within Deadline}}{\text{Total number of jobs arrived}}$$

Effective Processor Utilization (EPU) gives information about how efficiently the processor is used and it is defined as[1][2][6][8][9],

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

Where,

Vi is value of a job and,

Value of a job = Execution time of a job, if the job completes within its deadline.

Value of a job = 0, if the job fails to meet the deadline.

R is set of all the jobs which are executed by the CPU.

T is total time of scheduling.

VI. FINAL RESULTS

SR and EPU have been calculated for different Load value from 0.50 to 3.00. We have given total 18 files as input which have tasks with different Load value. Each input file containing total 200 Task Sets. Performance of both Least Laxity First (LLF) scheduling algorithm and Enhanced Least Laxity First (E_LLF) scheduling algorithm measured in same the environment and with help of same task set.

Results of LLF and E_LLF scheduling algorithm are as shown below :

LLF Scheduling Algorithm :

Load	SR	PUT
0.5	100	50.95
0.6	100	60.93
0.7	100	70.84
0.75	100	75.64
0.8	100	80.57
0.85	100	85.46
0.9	100	90.37
0.95	100	95.29
1	100	99.78
1.01	99.84	99.57
1.02	89.63	84.56
1.03	73.89	68.05
1.04	55.96	49.5
1.05	47.52	41.93
1.1	23.89	20.4
1.5	5.57	3.78
2	3.59	1.9
3	1.74	0.81

E_LLF Scheduling Algorithm :

Load	SR	PUT
0.5	100	50.95
0.6	100	60.93
0.7	100	70.84
0.75	100	75.64
0.8	100	80.57
0.85	100	85.46
0.9	100	90.37
0.95	100	95.29
1	100	99.78
1.01	99.85	99.62
1.02	97.01	93.44

1.03	94.4	90.23
1.04	90.53	84.62
1.05	88.05	81.43
1.1	80.54	71.05
1.5	54.22	47.65
2	40.79	38.43
3	20.8	24.77

VII. RESULT COMPARISON

Figure 1 shows Load Vs. %Success Ratio (SR) comparison of LLF and E_LLF scheduling algorithms. Figure 2 shows Load Vs. %Through Put (PUT) comparison of LLF and E_LLF scheduling algorithms.

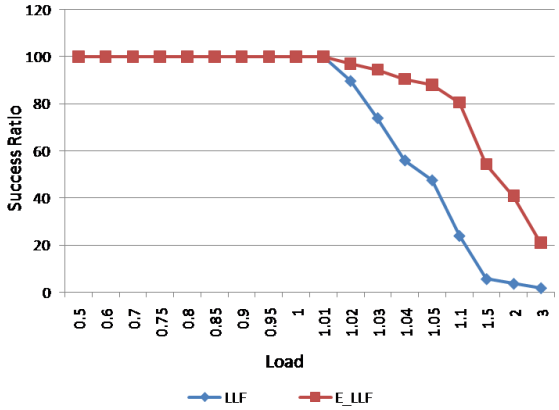

Figure 1: Load Vs. % Success Ratio (SR)

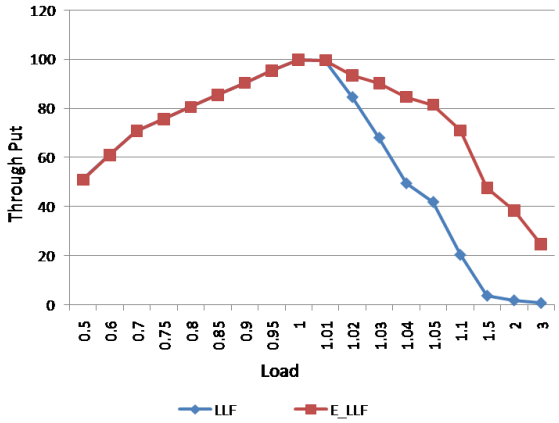

Figure 2: Load Vs. % Through Put (PUT)

ACKNOWLEDGEMENT

We would like to thank Dr. Apurva Shah at MS University for helping in simulation of scheduling algorithms. We would also like to thank IT Department of G H Patel College of Engineering and Technology.

VIII. CONCLUSION

In under loaded condition

Proposed E_LLF scheduling algorithm gives optimum result same as LLF Algorithm.

In Overloaded Condition

Proposed E_LLF scheduling algorithm deletes jobs which have already missed the deadline or expected to miss its deadline. OR

Ignores job with maximum execution time in task set based on some static priority assigned to the task. Thus, It gives improved performance than LLF scheduling algorithm.

REFERENCES

[1] Prajapati, V.; Shah, A.; Balani, P., "Design of new scheduling algorithm LLF_DM and its comparison with existing EDF, LLF, and DM algorithms for periodic tasks" Intelligent Systems and Signal Processing (ISSP), 2013 International Conference, 1-2 March 2013, pp.42-46.

[2] A M Shah, Ph. D. Thesis, "Dynamic Scheduling for Real-Time Operating Systems", Information Technology Department, Sardar Patel University, India, 2010.

[3] Thakor, D.; Shah, A., "D_EDF: An efficient scheduling algorithm for real-time multiprocessor system," Information and Communication Technologies (WICT), 2011 World Congress on , vol., no., pp.1044,1049, 11-14 Dec. 2011

[4] Real-Time Systems by Jane W. S. Liu (Pearson Publication).

[5] Real-Time Systems Theory and Practices by Rajib Mall.

[6] Prem Sindhi &Ravindra K. Gupta, "Enhancement in LLF Real-Time dynamic scheduling algorithm using conventional RM algorithm", International Journal of Computer Applications (0975 – 8887), Volume 31– No.6, October 2011,pp. 6-10.

[7] Guangyi Chen; WenfangXie, "On a laxity-based real-time scheduling policy for fixed-priority tasks and its non-utilization bound," Information Science and Technology (ICIST), 2011 International Conference, 26-28 March 2011, pp.7-10.

[8] Prem Sindhi, Mr. Ravindra K. Gupta, "Performance comparison of real-time scheduling", International Journal on Science and Technology (IJSAT) Volume II, Special Issue I, 2011, pp205-211, 2011.

[9] Apurva Shah, KetanKotecha, "Efficient Scheduling Algorithms for Real-Time Distributed Systems" 1st International Conference on Parallel, Distributed and Grid Computing (PDGC - 2010), 2010, pp. 44-48.

[10] J. Stankovic, "Misconceptions about Real-Time Computing", IEEE Computer, 21, October 1988.

