International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 6 (2015) pp. 14621-14628

http://www.ripublication.com

A Novel Scheduling Architecture For Real Time System

[1]R.C.Dharunendhar
[1]m.E. (Embedded System)
Department of ECE, Sathyabama University,
Chennai
[1]Dharunendharrc@Gmail.Com
[2]Dr.E.Logashanmugam
[2]faculty Head
Department Of ECE, Sathyabama University,
Chennai

Abstract

A real-time system is system that produce a response within a specified time. scheduling is the process of controlling and prioritizing messages sent to a processor. This project is design to increases the performance of the task completion in real time applications and to reduce the possibility of collision and thereby increase the communication reliability and to meet the critical requirement of timing determinism of applications. In the proposed method Effective scheduling technique is applied and Immediate response to the interrupts

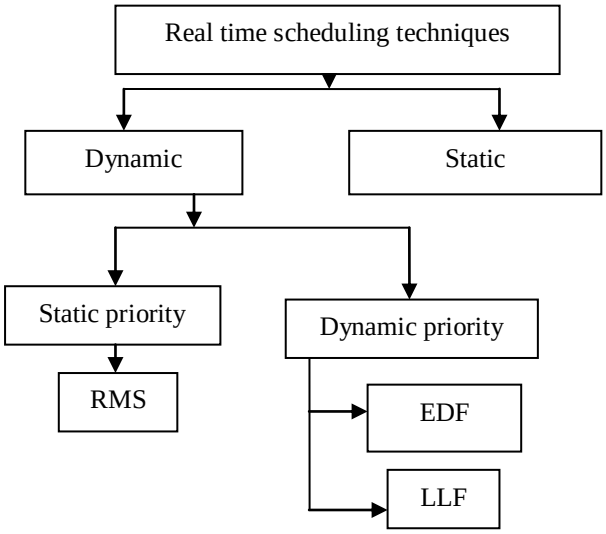
Keywords: Real-Time Operating Systems, Scheduling Algorithm, Earliest Deadline First (EDF), LLF: Least laxity first, RM: Rate monotonic.

Introduction

Real-Time Systems

A system in which the correctness of computation of output is not only dependent on the logical correctness but also on the time at which the output is delivered or A system is synchronized with time in the environment.

Real-time scheduling techniques can be largely divided into two categories: Static and Dynamic. Static algorithms assign priorities at design time. Dynamic algorithms assign priorities runtime, based on execution parameters of tasks.



Dynamic scheduling can be either with static priority or dynamic priority. RM(Rate Monotonic) is examples of dynamic scheduling with static priority. EDF(Earliest Deadline First) and LLF (Least Laxity First) are examples of dynamic scheduling with dynamic priority. EDF and LLF algorithms are optimal under the condition that the jobs are preemptive, there is only one processor and the processor is not overloaded

Rate Monotonic Algorithm

It assigns static priorities to tasks at the connection setup stage according to their request rates. Subsequently, each task is scheduled with the priority calculated at the beginning ,with no further rearrangement of priorities required. The priority corresponds to the importance of a task relative to other tasks. The task with the shortest period gets the highest priority, and the task with the longest period gets the lowest priority. It is an optimal and static, priority driven preemptive scheduling algorithm for preemptive ,periodic tasks.

Least Laxity First Algorithm

Least laxity first algorithm(LLF) assigns priority bases upon the slack time of a task. The laxity time is difference between the deadline, the remaining computation time and the run time. LLF always schedules first an available task with the smallest laxity. The laxity of a task indicates how much the task will be scheduled without being delayed. LLF is a dynamic scheduling algorithm and optimal to use a exclusive resource. LLF is commonly used in embedded systems. Since the run time is not defined, laxity changes continuously. The advantage of allowing high utilization is accompanied by a high computational effort at schedule time and poor overload performance.

Earliest Deadline First Algorithm

The earliest deadline first (EDF) algorithm is the best known algorithm for real-time processing. At any arrival of a new task, EDF immediately calculates a new order. It preempts the running task and schedules a new process according to its deadline. The interrupted task is rescheduled later. EDF schedules not only periodic tasks ,but also tasks with arbitrary requests, deadlines, and service execution times. However, EDF cannot guarantee its performance under overload scheduling condition EDF is an optimal and dynamic algorithm. A dynamic algorithm schedules every instance of each incoming task according to its specific demands. It may reschedule periodic tasks in each period. For a dynamic algorithm like EDF, the upper bound of processor utilization is 100 percent. If a set of tasks can be scheduled by any priority assignment, EDF is optimal scheduling algorithm.If EDF has already assigned the priority for a new task, the scheduler must rearrange the priorities of other tasks until the required priority is free. In worst case, the priorities of all tasks have to be rearranged, which may cause considerable overhead to the processor

Maximum Urgency First

Maximum Urgency First (MUF) scheduling algorithm resolves the problem of unpredictability of the system during transient overload that is when CPU load factor exceeds 100%. This algorithm is urgency based scheduling algorithm. It is a mixed priority scheduling algorithm and employs both fixed as well as dynamic priority for efficient scheduling of tasks. With this algorithm, each task is given an urgency which is defined as a combination of two fixed priorities (criticality and user priority) and a dynamic priority that is inversely proportional to the laxity. The critical priority is set to 1 if tasks are present in the critical set and the CPU load factor for these tasks is less than 100%.

Critical priority > dynamic priority > user priority The MUF algorithm assigns priorities in two phases. Phase one is concerned with the assignment of staticpriorities to tasks. Static priorities are assigned once and do not change after the system . the MUF scheduling algorithm as mentioned in V.Salmani et.al paper is as follows

In phase 1, fixed priorities are defined and the scheduler sorts the task in increasing order of their periods.

First N tasks having CPU utilization<100% are taken in critical tasks and the remaining tasks are considered in non-critical task set. Every task is given a optimal user priority that depends entirely on the user.

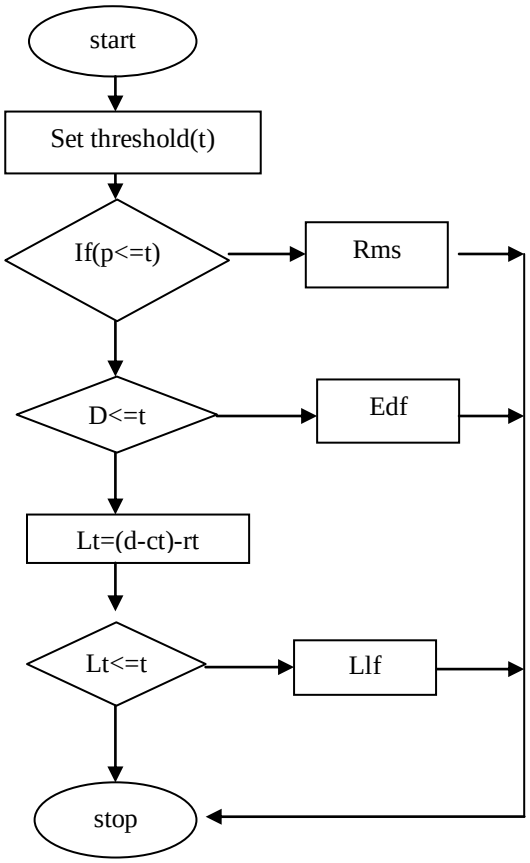
In phase 2, dynamic priorities are set.. If there is only 1 critical task the task is executed. If more than 1 critical task is there, the task with the minimum laxity is picked up for execution. If there are more than 1 tasks with the same laxity then the task with the highest user priority is considered and scheduled

Simulation

The proposed algorithm is simulated and verified its performance. CHEDDAR simulator is used to evaluate the performance of the algorithm. For simulation, the

number of deadline violation is measured while the number of processes is increased. The threshold value is 0.9 for this experiment

Proposed Algorithm:



Algorithm Perfomance metrics	Edf	Llf
Cpu utilization factor	High	High
No. of context switching	Less	High
Optimal	Yes	Yes
Deadline miss chances	Average	Average
Effectiveness	Optimal Easy to implement	Takes execution into consideration
Limitations	Not work in overload condition	In laxity, more context switching takes place

Types of Real-Time Tasks

Based on the way real-time tasks recur over a period of time, it is possible to classify them into three main categories: periodic, sporadic, and aperiodic tasks. In the following, we discuss the important characteristics of these three major categories of real-time tasks.

Periodic Task:

A periodic task is one that repeats after a certain fixed time interval. The precise time instants at which periodic tasks recur are usually demarcated by clock interrupts. For this reason, periodic tasks are sometimes referred to as clock-driven tasks. The fixed time interval after which a task repeats is called the period of the task.

Sporadic Task:

A sporadic task is one that recurs at random instants. A sporadic task Ti can be is represented by a three tuple:

$T_i = (e_i, g_i, d_i)$

where e_i is the worst case execution time of an instance of the task, g_i denotes the minimum separation between two consecutive instances of the task, d_i is the relative deadline. The minimum separation (g_i) between two consecutive instances of the task implies that once an instance of a sporadic task occurs, the next instance cannot occur before g_i time units have elapsed.

Aperiodic Task:

An aperiodic task is in many ways similar to a sporadic task. An aperiodic task can arise at random instants. However, in case of an aperiodic task, the minimum separation gi between two consecutive instances can be 0.

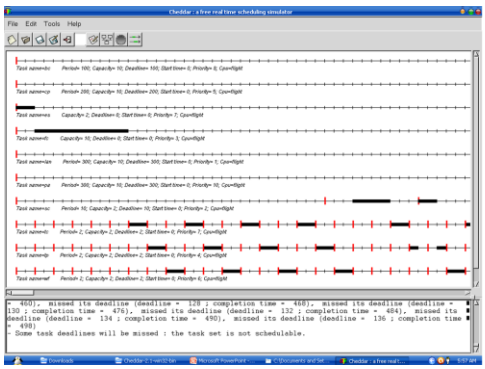
Comparison of real time scheduling algorithms

System Description

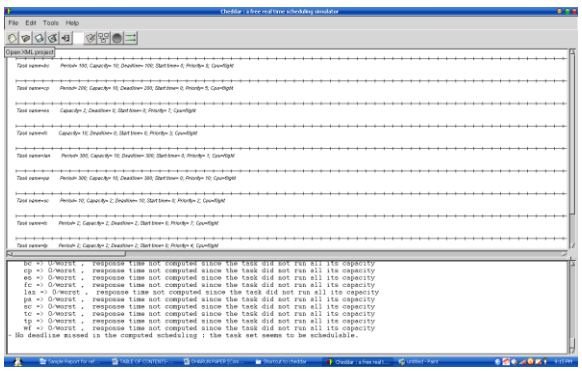
Automated avitiation control We consider simple flight control system. the system is divided into ten task .the ten task are listed below

Task	Shortcut	Type of task	Execution time	Deadline
Landing	Lan	Sporadic	10	180
Surface clearance	sc	sporadic	10	175
Fuel consumption	Fc	Aperiodic	10	170
Type of plane	Tp	periodic	10	165
Parking lot	Pl	Periodic	10	160
Chute availability	Ca	Sporadic	10	155
Weather forecast	Wf	Aperiodic	10	150
Turbulence check	tc	Periodic	10	145
Bird control	Bc	Periodic	10	140
Emergency support	Es	aperiodic	10	135

Below graph shows how the task is scheduled in cheddar simulation software . the below diagram shows the results for edf algorithm

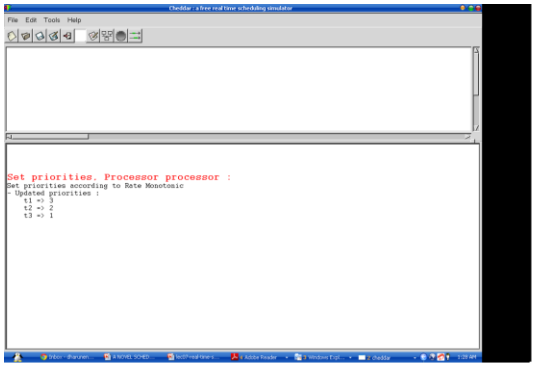


The below diagram shows the result for ten task using RMS



The below diagram shows the result for RMS

Algorithm



Conclusion

In this paper a review of Different scheduling algorithms for a real time system is done which specifies the working of these algorithms and their usage with the schedulability analysis. Real time operating Systems have been emerging in the field of research. A lot of work has been done in the field of real time scheduling algorithms so as to how to enhance the CPU utilisation and many different new algorithms have been developed to meet the utilisation of tasks within the cpu efficiently

Acknowledgements

We wish to give our heartiest gratitude to **Dr.E.Logashanmugam**, Faculty head of the electrical and electronics and Prof. SUGADEV, for his constant motivation, knowledge sharing and support behind this paper

References

[1]. B. Andersson and J. Jonsson, "The Utilization Bounds of Partitioned and Pfair Static-Priority Scheduling on Multiprocessors are 50 percent," 15th Euromicro Conference on Real-Time Systems (ECRTS'03), Porto, Portugal, July 02-04, 2003.

[2]. J. Anderson and A. Srinivasan, "Early release fair scheduling," In Proceedings of the EuroMicro Conference on Real-Time Systems, IEEE Computer Society Press, pp. 35-43, Stockholm, Sweden, June 2000.

[3]. N. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority preemptive scheduling," Software Engineering Journal, pp. 284-292, 1983.

[4]. H. Aydin, P. Mejia-Alvarez, R. Melhem, and D. Mosse, "Optimal reward-based scheduling of periodic real-time tasks," In Proceedings of the Real-Time Systems Symposium, IEEE Computer Society Press, Phoenix, AZ, December, 1999.

[5]. J. W. de Bakker, C. Huizing, W. P. de Roever and G. Rozenberg, "Real-Time: Thory in Practice," Preceedings of REX Workshop, Mook, The Netherlands, Springer-Verlag company, June 3-7, 1991.

[6]. J. M. Bans, A. Arenas, and J. Labarta, "Efficient Scheme to Allocate Soft-Aperiodic Tasks in Multiprocessor Hard Real-Time Systems," PDPTA 2002, pp. 809-815.

[7]. S. Baruah, N. Cohen, G. Plaxton, and D. Varvel, "Proportionate progress: A notion of fairness in resource allocation," Algorithmica , Volume 15, Number 6, pp. 600-625, June, 1996.

[8]. P. Berman and B. DasGupta, "Improvements in Throughput Maximization for Real-Time Scheduling," Department of Computer Science, Yale University, New Haven, CT 06511, January 31, 2000.

[9]. S. A. Brandt, "Performance Analysis of Dynamic Soft Real-Time Systems," The 20th IEEE International Performance, Computing, and Communications Conference (IPCCC 2001), April, 2001.

[10]. A. Burns, "Preemptive priority based scheduling: An appropriate engineering approach," Technical Report, YCS-93-214, Department of Computer Science, university of York, UK, 1993.

[11]. A. Burns, "Scheduling hard real-time systems: A review," Software Engineering Journal, Number 5, May, 1991.

[12]. G. C. Buttazzo, "Hard Real-Time Computing Systems: predictable scheduling algorithms and applications," Springer company, 2005.

[13]. J. Carpenter, S. Funk, P. Holman, A. Srinivasan, J. Anderson, and S. Baruah, "A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms," Handbook of Scheduling: Algorithms, Models, and Performance Analysis, Edited by J. Y. Leung, Published by CRC Press, Boca Raton, FL, USA, 2004.

[14]. M. Chen and K. Lin, "A Priority Ceiling Protocol for Multiple-Instance Resources," Proc. of the Real-Time Systems Symposium, 1991.

[15]. M. Chen and K. Lin, "Dynamic Priority Ceiling: A Concurrency Control Protocol for Real-Time Systems," Real-Time Systems Journal 2, 1990. [14] A. J. Ferreira

[16]. A. J. Ferreira and M. A. T. Figueiredo, mage compression," Signal Process., Image Commun., vol. 21, no. 5, pp. 378–389, 2006.