International Journal of Computer and Electrical Engineering, Vol.2, No.6, December, 2010
1793-8163

A New Scheduling Algorithm for Real Time System

Yaashuwanth .C and Dr.R. Ramesh

Abstract—**The main objective of this paper is to develop a new algorithm for scheduling real time tasks. Real time scheduling algorithms such as rate monotonic and deadline monotonic plays an important role in scheduling real time tasks in a real time environment .There are some cases where may arise inconsistencies such as tasks having less task period but their execution is not very important. In this case, when scheduled under rate monotonic algorithm the cpu unnecessarily spends time in scheduling the tasks that are not uttermost importance. The proposed algorithm eliminate this drawback and combines the advantages of both Rate monotonic and Deadline monotonic algorithms. It also incorporates a priority component which is specified by the user which denotes the importance of tasks in the system.**

Key words—**Real time tasks, Rate monotonic, Deadline monotonic architecture, priority, deadlines, task periods**

I. Introduction

A real-time computing system can be defined as a real-time application which is expected to respond to stimuli within some small upper bound in response to time and any late result is as bad as a wrong one. Thus correctness of a real-time system could be stated true with logical perfection in the computational result and its timeliness. A *soft real-time system* is a system that has timing requirements, but occasionally missing the task deadlines it has negligible effects. In simple terms, the scheduler defines the state of tasks within the system - running, ready, suspended or killed (deleted). It also includes the dispatching function, i.e. loading a new task into the processor. Scheduling decisions are made at frequent intervals, these being based on a set of scheduling rules or strategy. For embedded systems, the primary objective is to ensure that all tasks are completed within specific time frames. Processor throughput is much less important. Unfortunately, a multitasking design solution brings with it a time overhead. The result is that the amount of time available for executing the application is reduced. What we need, therefore, is a scheduling strategy which gives efficient processor utilization with minimum overhead.

Sha et. Al [1] provided extensive work on fixed priority scheduling, dynamic priority scheduling, soft real-time scheduling, and feedback scheduling. The author [3] discussed issues of task value, overheads, static versus dynamic scheduling, preemption versus non-preemption. A variety of real-time scheduling policies, various task attributes and performance comparisons of the various policies are explained in Task scheduling policies for real-time systems[4]. A new schedulability analysis of periodic, a periodic using fixed priority is discussed . [6]

The authors [6][7] discussed the problems involved in multi-program scheduling on a single processor shows that full processor utilization can be achieved by dynamically assigning priorities on the basis of the tasks deadlines. Statistically framework for earliest deadline first scheduling algorithm and applications of earliest deadline first algorithm in real time environment is portrayed by the authors [9],[10].

Real-time scheduling theory has shown a transition from cyclical executive based infrastructure to a more flexible scheduling models such as fixed-priority scheduling, dynamic-priority scheduling, feedback scheduling or extended scheduling[1]. In fact, recent studies show that almost every existing real-time operating system provides only POSIX-compliant fixed-priority scheduling [12] since it can be easily implemented in commercial kernels. A. Burns[11] proved that standard analysis of fixed priority assumes that all the computations of each task must be completed in task deadlines but in practice this is not the case, deadlines are most currently associated with last observed event of the task. internal events and kernel overheads can occur after the deadlines.

II. Rate Monotonic

This policy considers a single task timing criterion, task period. Ready tasks are arranged in order dependent on their period that have the shortest period is placed in the front of the ready queue, being allocated highest priority. The next task is the one which has the second shortest period, and so on. Thus, as we go down the ready list, task periods always increase (a monotonic sequence). In this strategy, the most frequent task executes first. A running task may be pre-empted if a task with a shorter period is readied.

Drawbacks of Rate Monotonic Architecture: In this architecture all the tasks are considered of equal priority this may lead to the starvation of tasks having large task periods.

III. Deadline Monotonic

This scheduling policy is the extension of rate monotonic scheduling. Here the deadlines of a process is lesser than the period and the priority will be assigned based on their deadlines It allows the completion of process execution to be defined more precisely. It allows sporadic process to be easily incorporated.

Drawbacks of rate monotonic architecture: In this architecture all the tasks are considered of equal priority.

Yaashuwanth .C and Dr.R. Ramesh are with Department of Electrical and Electronics Engineering
Anna University Chennai, Chennai 600 025.
yaash_success@yahoo.co.in

This may leads to the starvation of tasks having large task deadlines.

From this literature it is evident that the rate monotonic and deadline monotonic architecture all tasks are considered to be of equal importance. The proposed architecture which is described combines the advantages of both rate monotonic and deadline monotonic by taking both task period and deadline and also adds priority component thus eliminating the drawback of both the architecture

IV. Proposed Architecture

The proposed architecture covers the drawbacks of both rate monotonic (RM) and deadline monotonic algorithm (DM). In rate monotonic all the tasks are assumed to be of equal importance and the tasks are scheduled in the task scheduler based on task period. Because of this drawback, there may be some unimportant tasks which may have less task period that could be scheduled before an important task (since all the tasks are considered to be of equal importance in RM scheduling). The same criteria also implies to deadline monotonic algorithm where all the tasks are assumed to be of equal importance and are scheduled based on deadlines. The proposed architecture eliminates this drawbacks by incorporating a priority component which is defined by the user in the same way as task period and deadline. Percentage of weightage will be allocated to all these criteria and based on these weightage, a new component scheduling component (SC) will be computed. Based on the scheduling value the tasks will be scheduled in the system. This proposed algorithm can be implemented in real time system where neither the task period nor deadline vary with time (same as rate monotonic and deadline monotonic).

TABLE 4.1 COMPARISON BETWEEN EXISTING ALGORITHMS AND PROPOSED ARCHITECTURE

Criteria	Rate monotonic (RM)	Deadline monotonic (DM)	Proposed architecture
Task importance	Equal importance to all tasks	Equal importance to all tasks	Priority component is added
Scheduling criteria	Task period	Task deadline	Task period Task deadline Task priority
Task scheduling	Tasks are arranged in ascending order based on task period	Tasks are arranged in ascending order based on deadline	Percentage of weightage is allocated based on the computations a new component scheduling component will be computed (SC)based on the scheduling component value tasks will be placed in ascending order
Time difference	Task period does not vary with time	Task deadline does not vary with time	Task period, Task deadline and Task priority does not vary with time
Type of scheduling	Static scheduling	Static scheduling	Static scheduling

V. Scheduling Component Calculation

The proposed architecture deviates from the current rate monotonic and deadline monotonic as it incorporates priority component. The proposed architecture contains three components) components are assigned importance weightage percentage.

(eg task period = 20%, deadline = 30% ,priority =50%)

Since it is a static scheduling algorithm and tasks with two different criteria can be bought together which incorporates the of third criteria called priority .

No. of priority in the system must be always equal to the weightage assigned to the priority component of the system. (percentage)

Task Period Component(TPC) is calculated by

Weightage percentage * Task period

(eg let us assume the task period for task as 40
This implies 20% * task period = 20 % * 40 = 8)

Task Deadline component(TDC) is calculated by

Weightage percentage * Task Deadline

(eg let us assume the task period for task as 40

This implies 30% * task Deadline = 30 % * 25 = 10)

Task Priority component(PC) is assigned by the user

(In the above example weightage of task priority is 50% since the weightage is equal to no of priority in the system. The priority in the system ranges from 1 to 50 eg let us assign the priority component as 25)

The priority range doesn't reflect the no of tasks in the system

Scheduling component(SC) is calculated by

Scheduling Component (sc) = task priority component(TPC) + Task Deadline Component (TDC) + Priority component (pc)

(eg SC = 8 + !0 + 25 = 43)

The scheduler rearranges the Scheduling component based on ascending order and schedules the tasks

If the scheduling component of two or more tasks are

IACSIT International Association of Computer Science and Information Technology WWW.IACSIT.ORG

same, then the component with next highest percentage of weightage is taken into criteria (here the priority of the task should be taken into account.

CASE STUDY

Here the period weightage percentage is assumed as 20% and deadline weightage percentage is assumed as 30% and priority weightage is 50%

Let us assume that all the tasks arrive at equal time

TABLE 5.1 INPUT

Task	Period (milli sec)	Deadline (milli sec)	Cpu Burst (milli sec)	Priority
1	25	40	7	6
2	50	60	12	1
3	25	30	8	10
4	40	15	3	2
5	10	15	7	8

Five processes has been defined (with its priority, period, deadline, cpu burst), these five processes are scheduled in the proposed architecture. The context switch, waiting time, turn around time have been calculated and the results are compared accordingly.

TABLE 5.2 OUTPUT

Task	Task Period Component (TPC)	Task Deadline Component (TDC)	Priority Component (PC)	Scheduling Component (SC)
1	5	12	44	61
2	10	18	49	77
3	5	9	40	54
4	8	5	48	61
5	2	5	42	49

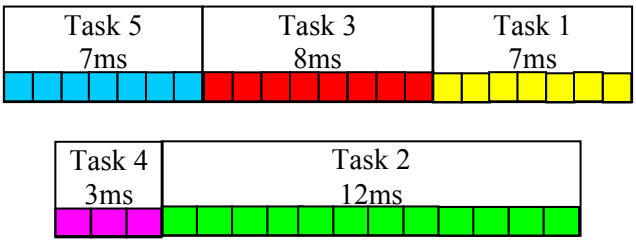

Figure 5.1 Task executions in proposed algorithm

Here for task 1 the period is 25 milliseconds and the deadline is of 40 milliseconds and it has 7 milliseconds cpu burst. We assign a priority component of 6 to task 1. In rate monotonic algorithm this task will be scheduled second since Rate monotonic algorithm deals with less task period scheduled first from all the tasks in the task set. In deadline monotonic algorithm this task will be scheduled fourth since Deadline monotonic algorithm deals with tasks with less deadline are scheduled first. In the proposed architecture we assign a priority component. Task 1 will be assigned 6 priority component. Task period component (TPC) , Task deadline component (TDC), and Priority component (PC) is calculated and Scheduling component (SC) is calculated by the addition of all the three criteria the task with highest scheduling criteria in the task set is allocated the processor. Task 1 in this task set will be the third task which will be allocated for the processor. Here Task 1 and Task 4 has the same scheduling component. But task 4 has the higher priority and will be the first to be allocated by the processor for execution. In this way all the tasks will be executed in the increasing order of scheduling component.

REFERENCES

[1] Sha, L., Abdelzaher, T, Arzen, K., Cervin, A., Baker, T.P., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., and Mok, A., Real time scheduling theory: A historical perspective, in Real-Time Systems, 28(2):46-61, Nov. 2004.

[2] Krithi Ramamritham and John A. Stankovic., Scheduling Algorithms and Operating Systems Support for Real-Time Systems, in Proceedings of the IEEE, Vol. 82, No. 1, pp. 55-67, Jan. 1994.

[3] John A. Stankovic, Marco Spuri, Marco Di Natale, and Giorgio Buttazzo, Implications of Classical Scheduling Results for Real-Time Systems, in IEEE Computer, pp. 16-25, June 1995

[4] Barbara Korousic-Seljak, Task scheduling policies for real-time systems, in Microprocessors and Microsystems, pages 501-511, Volume 18 Number 9 Nov. 1994.

[5] Liu, C.L., and James W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, in Journal of the ACM, Vol. 20, No. 1, pp. 46-61, Jan. 1973.

[6] Enrico Bini and Giorgio C. Buttazzo, Schedulability Analysis of Periodic Fixed Priority Systems, in IEEE Transactions on Computers, Vol. 53, No. 11, pp. 1462-1473, Nov. 2004.

[7] Brinkley Sprunt, Aperiodic Task Scheduling for Real-Time Systems, in Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, August 1990.

[8] Sha, L., Abdelzaher, T, Arzen, K., Cervin, A., Baker, T.P., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., and Mok, A., Real time scheduling theory: A historical perspective, in Real-Time Systems, 28(2):46-61, Nov. 2004.

[9] Zhi Quan and Jong-Moon Chang " A Statistical Framework for EDF Scheduling" journal on IEEE COMMUNICATION LETTERS VOL 7 NO.10 OCTOBER 2003 pg 493-495

[10] Houssine Chetto and Maryline Chetto " Some Results of Earliest Deadline Scheduling Algorithm" journal on IEEE TRANSACTION ON SOFTWARE ENGINEERING. VOL !%. NO.10 OCTOBER 1989 pg 1261-1269

[11] .Burns.A "Fixed priority scheduling with deadline prior to completion" Real-Time systems Research Group Department of computer science university of york, UK

[12] Ripoll I. et al. "Rtos state of the art analysis. Technical report", OCERA Project, 2003.

[13] Richard roehi "Designing a fairer round robin scheduling algorithm" SIGCOMM '95 Cambridge, MA USA 0 1995 ACM 0-89791 -711-1 /95/0008 www.cmg.org/proceedings/2005/5056.pdf

[14] Shreedhar. M , George Varghese "Efficient fair queuing using deficit round robin" SIGCOMM '95 Cambridge, MA USA 0 1995 ACM 0-89791 -711-1 /95/0008 *portal.acm.org/citation.cfm?doid=217382.217453*

Mr. C.Yaashuwanth completed his B.Tech. degree in Information technology at BSA CRESCENT Engineering College chennai, He completed M.E. in Embedded System Technologies at VEL TECH Engineering college Chennai. He is currently Pursuing his Ph.D program under Dr. R .Ramesh

Dr. R. Ramesh pursued his B.E. degree in Electrical and Electronics Engineering at University of Madras, Chennai, and completed his M.E. degree in Power Systems Engineering at Annamalai University, Chidambaram. He received his Ph.D. degree from Anna University, Chennai and has been a faculty of Electrical and Electronics Engineering Department of College of Engineering, Guindy, Anna University, Chennai since 2003. His area of interest are Real-time Distributed Embedded Control On-line power system analysis and Web services.