ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 2, February 2018

A model-based scheduling approach for selection of Real-Time Scheduling Algorithm on basis of Different Parameters

[1] Ajitesh Kumar, [2] Mona Kumari, [3] S.K.Gupta
[1][2] GLA University, [3] BIET Jhansi

***Abstract* - In Modern days, the real-time system plays an important role in our modern and digital society. The success of any real-time application is totally depends upon the selection of optimal scheduling algorithm. In real time application, every task should have the nature of deadlines and time when they arrived, on the basis of these parameters we observe the response time of different scheduling algorithm then we select the optimal algorithm for a particular application. So in this paper, our aim is to reduce the complexity of real-time system researcher for selection of scheduling algorithm for a particular application. This model-based approach is an extent the state of any real-time system in the area of scheduling. This approach works in any uniprocessor system.**

Keywords: Hard RTS, Deadline, WCET.

I. INTRODUCTION & MOTIVATION

The real time system is much different to non-real time system. They perform and gives result within a certain duration of time. In real time application they monitor and controls the process and must react to change in timely fashion, sometimes in milliseconds. In Hard Real time system, a sophisticated coordination is required and timely responses to the events is a challenge, so for scheduling of any real time application is required optimal scheduling algorithm. This motivates the study and evaluation of different real time algorithm and choose the optimal algorithm for a time critical task because failure to lead the loss of life and properties.

For any RTS, the major issue is scheduling algorithm. There are many types of scheduling algorithm exists due to various needs and requirements of real time applications [5]. The selection of scheduling algorithm having very importance in any real time application and greatly influenced the algorithm will serve what kind of system. In this research article we present a model based approach that can be very helpful to the real time system researcher for selection of optimal algorithm for a particular real time application [8].

The two important characteristics of any real time application is time of completion of task and when they arrived. The tasks arrivals may be periodic with constant intervals and may be aperiodic with random in nature. [12] So that the important factor is a time constraint task should be completed within their time interval and did not miss their deadlines. The task having hard deadline must be executed and gives result with in their specified time period and never miss their deadline.

In different real time application, the nature of deadline may be differing and if any task misses their deadline, the significance of execution for such a task is nil. If any hard deadline missed, that causes the failure of system. Missing deadline is major factor for any hard real time system so that the chosen of scheduling algorithms for a particular task set is playing a very vital role to successfully completion of task set and gives their specified result in a given period of time, so in this paper we are having much more concern with timing requirement in hard real time system [14].
Any real time task set using some terminology to better understand and carried out.

Release Time: Time required to any task to be ready to release
Deadline: The finishing time of any task
Slack: Time available for maximum delay of any task

WCET: The time period required to complete any task successfully and in critical condition.

Run Time: Require time for completion of task
Hard Real Time System: Deadline (Time Constraint) should not be changed in any condition

Periodic Task: Task should have arrived after fixed time interval

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 2, February 2018

Aperiodic Task: Arrived at any random time constraints and such task should not be having any predefined arrival sequence

Sporadic Task: The pattern of arrival of task is a combination of both periodic and aperiodic task, in this type of task, they have Aperiodic execution time and rate of execution is periodic in nature.

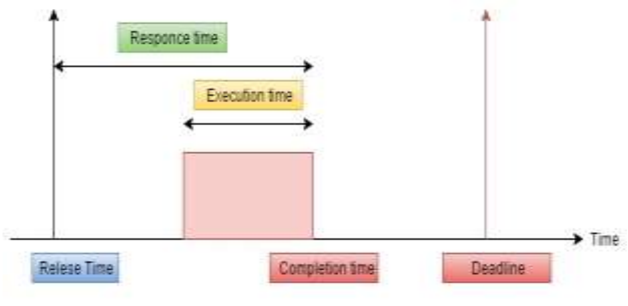


Fig. 1 Basic Characteristics Of Real time task Set

Pre-emptive/Non Pre-emptive: In Any real time, system it is compulsory to execute higher priority task first but if any lower task is executed with some resources it should not be allow to higher priority task to execute first. In same respect some task is never pre-empted their job. [6]

Fixed/Dynamic Priority: The priority of task is decided either fixed in nature or may be changed according to their need.

Independent/Dependent Task: In multitasking output system one task execution may be depended on other task results so may be some task having dependent and some task have independent [9].

The dependent task is having the shared memory and they communicate by transferring the information given by one task and required by another task.

We are organizing this paper is as follows. In section II we discuss some basic concept of real time system and having comparative study of different types of real time system algorithm and in section III we present the model based approach for a selection of an optimal scheduling algorithm for any real time application being the main section of the paper. In the next section IV, we are having some experiment result of real time application and finally section V covers the conclusion and references.

II. ANALYSIS OF SCHEDULING ALGORITHM

For Real Time System we are having so many scheduling algorithms but all algorithm falls in two categories one is offline algorithm and another categories is online scheduling.
In offline scheduling algorithm all decision like priority allocation, deadline, resource allocation and execution time are decided before the system started and scheduler having complete knowledge about all task set. The offline scheduling is very useful for any hard real time application because if scheduler knowing the complete knowledge about task set then it can execute as they meet their deadlines.
In online scheduling, all decision should be taken at runtime of the system and all scheduling decision depend upon the assigned priority of the task set. The priority decision may be static or dynamic in nature, in static the decision about priority of the task is taken before start of execution and dynamic assignment the decision about priority should be taken at run time.

The goal of any real time system is to scheduler ensure that the all task should meet their deadline and preventing from simultaneous access to shared resources. In this paper we are having the concern about only uniprocessor system so we are going to discuss and analysis the uniprocessor.

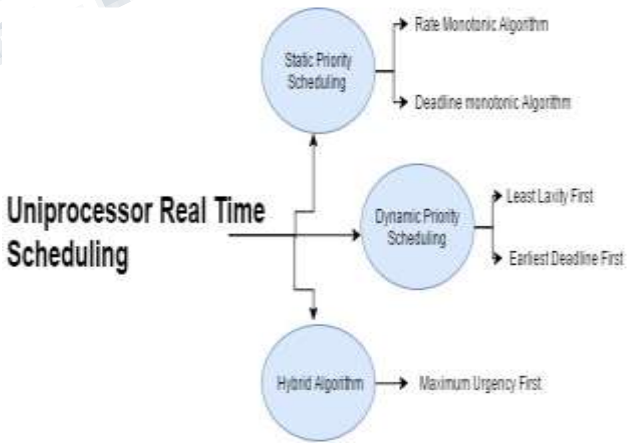

Uniprocessor Real Time Scheduling
Static Priority Scheduling
Rate Monotonic Algorithm
Deadline monotonic Algorithm
Dynamic Priority Scheduling
Least Laxity First
Earliest Deadline First
Hybrid Algorithm
Maximum Urgency First

Fig 2. Real Time Scheduling Algorithm

Rate Monotonic Algorithm:
In rate monotonic is a static priority algorithm and preemptive in nature.
This algorithm works on a simple concept. That is shorter the execution period and having the higher priority.

That why they are used the periodic task set. This algorithm should not be works for aperiodic and sporadic task set [2].

1.The priority task should come after a fixed interval of time and execute at the beginning of each period.
2.The deadline is shows the end of the period.
3.The tasks are not to be block by each other.
4.The scheduling overhead is assumed to be negligible.

Deadline Monotonic(DM)
In deadline monotonic algorithm also works with fix priority task set, is very similar to the RM.

Some extensions have been performed on RM to increase its performance. So that, when the tasks share its resources, we can also use the RM.

In order to prevent the simultaneous use from the shared resources, is used a technique called semaphore. In which case, when the task arrives to the critical section, it will lock and after the task exiting is released. The critical section is a part of the code for access to a shared source.
blocking. It occurs when a task will not execute by tasks with lower priority. To solve this problem, there are two methods as follows: PIP (Priority inheritance protocol) If a task blocks the task with higher priority, dynamically task priority will change. PCP (Priority ceiling protocol) This protocol has a semaphore that be allocated as the priority ceiling. Hence, it prevents the deadlock occurrence [6].

Dynamic algorithm:
Here there are two algorithms those are dynamic in nature that's varies at run time and decision should be made at run time.

EDF (Earliest Deadline First)
This algorithm is dynamic in nature and based on a simple concept.
Earliest deadline should have the highest priority and run first. This algorithm gives best performance at run time when at any time instance the scheduler decide which task should be run first. At run time such type of decision is very difficult for the scheduler. [5]
This algorithm works on pre-emptible task set.
All the condition and assumption should be valid and same of RM algorithm accept deadline equal to the execution time period.

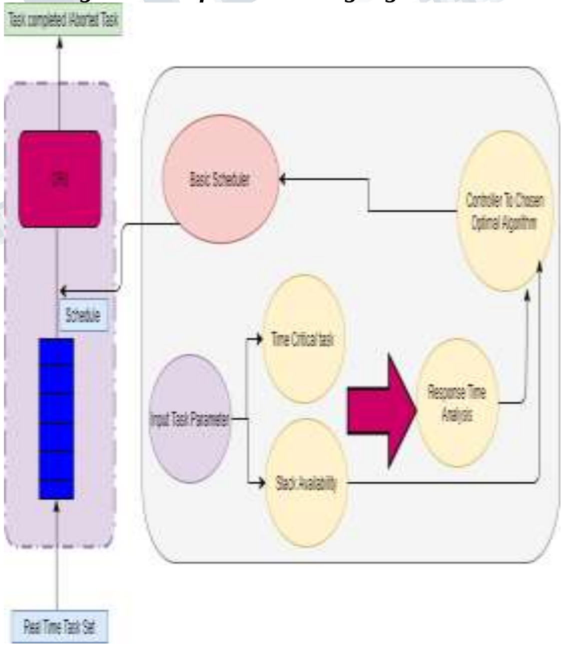
Least Laxity First(LLF)
This algorithm work on the concept of lowest slack availability run first on the processor.
This algorithm decide the schedule based on the laxity means slack. [9]
The laxity is the time interval where the task should be relaxed and execute without missing deadline.

III. PROPOSED SCHEDULER APPROACH

In this proposed method for selection of optimal algorithm from different available algorithm we are having a step by step method.

In this algorithm the response time analysis is done by accepting the task set requirement in all expect. We are having a ready task queue that hold the task arrival in the system. And the task should have the different parameters for the execution, these parameters should be analysis by given input to the analyzer

Fig:2 Model for scheduling algorithm selection



Step by step model based algorithm gives the solution of selection of scheduling algorithm for any real time application.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 2, February 2018

Step 1: Initialize the ready queue of the model

Step 2: Enter the task sets

Step 3: All parameters should be entering in queue like release time, execution time, deadline, WCET period and if there is any relative deadline

Step 4: Now firstly check the task should be periodic/Aperiodic/Sporadic in nature

Step 5: According to their nature the task set should be analyses in task set parameter analyzer

Step 6: First we calculate the slack availability in the task

Step 7: If slack is not available

Step 8: then go to step 16

Step 9: according to availability of slack we calculate the response time period by using different formulas

Step 10: Calculate Using different parameter, we check the priority of the task if the priority of the task set is fixed go to next step and if dynamic in nature go to step 17

Step 11: Then {Check the condition Deadline is equal to the execution time period} if it is then

Step 12: calculate utilization bound area of the task set

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1)$$

$$U \leq 0.69 \text{ as } N \rightarrow \infty$$

Where C is computation time and T is execution time period and n is no of task.

Also we calculate response time of the task set

Where R is release time of the task.

Step 13: Then {Check the condition Deadline is less than execution time period}
True then go to step 11

Step 14: Then {Check the condition Deadline is less or equal to the execution time period}
If true, then go to step 11

Step 15:
If the deadline is equal to the execution time, then

Utilization ratio = $\sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1$
And for the response time period go to step 11
Else
Check deadline is equal or less than execution time then also go to step 11 and calculate response time

Step16:
Now controller chosen the optimal algorithm based on above step parameters.
If Deadline is equal to execution time interval
Then
rate monotonic algorithm and deadline first both are optimal and then controller check the response time of the task set, those have less response time that can be chosen by controller as optimal algorithm
otherwise
Deadline first algorithm should be chosen by the controller

Step 17:
If the slack is available and we can reach the deadline with the execution time, then controller chosen the Least laxity first algorithm directly and this algorithm the scheduler does not any critical path.

Step 18:
Now the controller chosen the optimal algorithm

Step 19:
This information should be passes to the basic scheduler by controller

Step 20: Now the basic scheduler having the information about which scheduling algorithm is optimal for this particular task set

Step 21: Then the basic scheduler schedule the task set on the basis of algorithm which are chosen by controller.

Step 22: This schedule given to the CPU and task are executed according the schedule

Step 23: Task Executed/ task Aborted by CPU
Step 24: END

This model based algorithm that help the basic scheduler for chosen from the different available algorithm in real time single processor system.
In this model we are implemented and tested two static algorithms, rate monotonic and Deadline first algorithm and two for dynamic, one is earliest deadline first and another one is least laxity first algorithm.

IV. EXPERIMENTAL RESULT AND ANALYSIS FOR REAL TIME APPLICATION

For verifying the performance of this model based algorithm we are implemented and tested this approach in MATLAB and experimental result shows that it works for selection of the optimal algorithm.

Here we are considering a real time case study where 5 tiny computing device raspberry pi that can be repair in an electronics hub and that can be scheduled like that it will complete that job of repairing within a time limit and no job should miss their deadline.

List of Job repairs:

Perform the replacement of the port16 of the Pi1 at 2:00 hrs.
Pi2 has restart automatically problem and ready at 12:00 hrs.
Pi3 has OS problem and repair by 4:00 hours.
Pi4 needs RAM replacement and needs to be ready before 3:00 hrs.
Pi5 has SD card slot problem and needs to repair at 1:00 hrs.
Time required for repairs:
Port16 replacement needs 1 Hr.
Restart Problem needs 2 hrs.
OS problem 2 hrs.
RAM replacement 1 Hrs.
SD Card Slot problems needs 1 Hrs.
So that
T1= (Pi1, 1, 9, 14)
T2= (Pi2, 2, 10, 12)
T3= (Pi3, 2, 8, 16)
T4= (Pi4, 1, 11, 15)
T5= (Pi5, 2, 9, 13)

1. All task is entered in ready queue (T1, T2, T3, T4, T5) and task analyzer found the task set are periodic in nature and having the fixed priority and then the availability the slack should be check and find the slack should not be available so that the time critical path should be find and the information passes to the controller to choose the best optimal algorithm. So for that the controller check the deadlines of each and every task.

2. And controller find the deadline are less or equal to the execution time period.
3. So that the controller performs the step 11 in the given model based algorithm.

4. The controller found the response time of the task set is 0.46875s.

5. So that controller should decide and passes this information to the basic scheduler to schedule these task on the basis of response time analysis and utilization ratio of the task.

6. The information by controller is passes that RM should be used and this is best optimal algorithm

7. Visualized by basic scheduler based on controller information and scheduled by standard plot function, plot schedule(TS,'pooc',0)

8. And final schedule given to processor and all task should be scheduled.

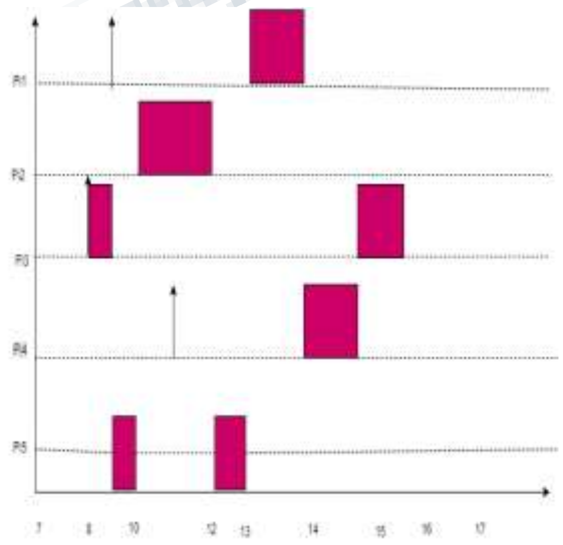


Fig3 shows scheduling by processor for raspberry pi making batched in MATLAB

This experimental result shows that this model based approach should be work in very efficient way and schedule the task set by chosen best optimal algorithm from available algorithms.

This model based approach having the capability of chosen the optimal algorithm by controller and passed the information to the basic scheduler and schedule and gives information to the processor and processor will execute the task set in such a manner no task should miss their deadline.

In this example this algorithm also reduces the context switching [1] in the processor that also help the processor to save the consumption of the energy.

CONCLUSION

This model based approach will work in all aspect and experimental result shows that this basic scheduler works for finding out the best optimal algorithm for any real time application.
As we know that a variety of real time algorithm are available but this algorithm finds out the best optimal algorithm and also scheduler gives the schedule to the CPU.
This paper provides the analysis summary of schedule. This model presents the algorithm that works on the basis of controller. The controller has the ability to take the decision.
This algorithm only chosen the optimal algorithm form four algorithm this can be improved and added some more algorithm features so that this is the feature aspect of the model based algorithm.

REFERENCES

[1] Ranvijay, Rama shankar Yadavand Smriti "Efficient energy constrained scheduling approach for dynamic real time system", First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010)

[2] Mehrin Rouhifar and reza, "A servey on Scheduling algorithm Approaches for Hard real Time Systems " IJCA December 2015.

[3] Robert. I, Davis, R and Burns A, "A Review of Fixed Priority and EDF Scheduling for Hard Real-Time Uniprocessor Systems", EWiLi'13, August 26–27, 2013, Toulouse, FRANCE

[4] Lindh. F, Otnes. T, Wennerström. J, "Scheduling Algorithms for Real-Time Systems".

[5] Davis, R. I. and Burns. A, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems", ACM Computer Survey, Vol. 43, No.4, Article 35, 44 pages, 2011.

[6] Shamim Shiravi and Mostafa E. Salehi, "Fault Tolerant Task Scheduling Algorithm for Multicore Systems", The 22nd Iranian Conference on Electrical Engineering (ICEE 2014), 2014, Shahid Beheshti University

[7] Sha L., Rajkumar R. and Lehoczky J. P., "Priority Inheritance Protocols: An Approach to Real Time Synchronization", IEEE Transactions on Computers 39(9), pp. 1175-1185, September 1990.

[8] G. E. Moore, "Cramming more components onto integrated circuits", Electronics, Vol. 38, No. 8, McGraw-Hill, 1965

[9] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores" in Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1-6, 2011.

[10] W. Y. Lee, "Energy-efficient scheduling of periodic realtime tasks on lightly loaded multicore processors", Parallel and Distributed Systems, IEEE Transactions on, vol. 23, pp. 530-537, 201

[11] J.-J. Chen and T.-W. Kuo, "Energy-efficient scheduling of periodic real-time tasks over homogeneous multiprocessors", in the 2nd international workshop on power-aware real-time computing, pp. 30-35, 2005

[12] K. Manudhane, A. Wadhe, "QoS-Aware Approaches to Real-Time task scheduling on Heterogeneous Clusters", international Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, pp. 174−180, 2013

[13] K. Houben and A. Halan, "An Energy-Aware Dynamic Scheduling Algorithm for Hard Real-Time Systems", 3rd Mediterranean Conference on Embedded Computing, MECO – 2014, ACM, PP. 14-17.

[14] Abhaya K. Samal , R. Mall and C. Tripathy, "Fault tolerant scheduling of hard real-time tasks on

multiprocessor system using a hybrid genetic algorithm”, Elsevier. Swarm and Evolutionary Computation, 2014.

[15] S. Ghosh, R. Melhem, D. Mossé, “Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems”, IEEE Trans. Parallel Distrib. Syst. 8 (3), pp. 272-284, 1997.

[16] Mohammad H. Mottaghi and Hamid R. Zarandi, “DFTS: A dynamic fault-tolerant scheduling for real-time tasks in multicore processors”, Elsevier.Microprocessors and Microsystems, Vol. 38, pp:88–97, 2014.

[17] A. Wiese, V. Bonifaci and S. Baruah, “Partitioned EDF scheduling on a few types of unrelated multiprocessors”, Springer, Real-Time Syst, vol. 49, pp:219–238, 2013.

[18] G. Yao, R. Pellizzoni, S. Bak, E. Betti and M. Caccamo, “Memory-centric scheduling for multicore hard real-time systems”, Springer, Real-Time Syst, vol. 48, pp:681– 715, 2012.