Study and Analysis of different Real Time Task Scheduling Algorithum

R. D. Ghodeswar Department of Comp. Sci and Engg, P.R.M.T.R. College of Engg. Badnera,Amravati
,E-mail: rdghodeswar24@gmail.com ,
Asso. Prof.R. R. Tuteja Department of Comp. Sci and Engg, P.R.M.T.R. College of Engg. Badnera,Amravati,
E-mail: ranu.tuteja@gmail.com,

Abstract— The main objective of real time systems is to complete its task and deliver services on timely basis by choosing an appropritate task scheduling algorithum. Here we discussed a brief overview of real time task scheduling algorithum by considering timing factor and functional requirement of the system.This paper summarizes the state of the real-time field in the areas of scheduling.The paper includes some mostly used scheduling approaches with example of each, these are: clock-driven approach, weighted round-robin apporach, priority driven approach. The main objective of this paper is to study and analysis of schedulability of clock driven scheduling algorithum and weighted round robin scheduling algorithums.

Index Terms— Real time system, Basic terms, Real time task scheduling algorithums, Clock Driven Apporach, Weighted Round Robin Apporach, Priority Driven Apporach Analysis of Schedulability.

1 INTRODUCTION

Real Time Systems: Real– time systems are the systems in which both factors are more important logically correct output as well as timing [1]. The thing that differntiate the real time system from non real time systems is it's time constraint, it means that system should complete task in a given time. Real time system can be categorised in two main types**:** Hard Real-Time System, Firm or Soft Real-Time System. This classification is done base on two terms tardiness and deadline of job. Tardiness is of job measure to check whether job meet its deadline or not. It is said that tardiness of job is zero if job completes at or before its deadline otherwise it is difference between its completion time and its deadline. In soft real time system tardiness increases while in hard real time system decreases may even become negative. Other is deadline if job misses its deadline then deadline is hard otherwise its timing constraint is hard. The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible when we consider soft real-time system. Mostly all the real-time systems in existence use preemption and multi-tasking.

2 BASIC TERMS

2.1 Jobs and Task

Job is basic unit of work done (schedule and executed) by the system. Task is the set of releted job that provide a certain function of system. .

2.2 Timing Constraints

Relese time: It is a time at which job becomes available for execution After its relese time job can be schedule and executed at any time.
Deadline: It is a time within job need to be completed. Job has no deadline if its deadline is infinity.
Response Time: It is a time duration from relese time of job to its completion.
Absolute Deadline: It is maximum allowable response time of a job sometimes it is also called as relative deadline

2.3 Task Precedence

If certain job say J2 depends on another job J1's execution that is J2 can begin its execution until J1 is executed this situation is known as task prcedance. It is said that J1 is predecessor of J2.

2.4 Data Sharing

In an operating system job in a system communicate via shared data that is data used by both jobs commomly. Like a job precedance here also problem of dependency arises as both jobs can not executed at same time. One job has to wait until other completes its execution to avoide the conflicts.

2.5 Types of Task

Periodic Task: Periodic task are those task which repeats after certain fix time period. This time period is called as period of task. Recurrence of task after certain fix time interval is demarceted by clock intervals.

Aperiodic Task: Aperiodic task occurs at randon instants.Two or more periodic task can occur at the same time that is in case of aperiodic task situation may occur when minimum duration between occurance of two job will be zero

Sporadic Task: sporadic jobs also occurs at random instants only the difference in aperiodic and sporadic jobs is that there is time gap in occurance of two consecutive jobs.

3 REAL TIME TASK SCHEDULING ALGORITHMS

Scheduling allocates processors time as well as resources to jobs ready for execution. In real time task scheduling the timely execution is major issue to be concsidered at the time of scheduling. We will comment on this as we examine the different scheduling paradigms in the next subsection. In real time system focused is on deadline of jobs. The main objective to be achived by the real time schedulars are minimizing total scheduling time, minimizing tardiness and minimizing number of tardy jobs.

3.1 CLOCK DRIVEN APPORACH

In this approach real time task scheduling the dicisions on which time the jobs of task will run are made at specific time istant. This approach is also known as time driven approach.
In this type of scheduling the scheduling points are determine by timer interrupts. The parameters of the jobs to be scheduled by the clock driven schedulars are previously known. Thus this schedular fix the schedule before the system stars that is schedule is computed off-line hence these schedulars are also called as off-line schedulars. This off-line schedule is stored and used for run time when system starts its execution. As schedule is previously computed it saves scheduler time of making scheduling decisions thus overload at run time can be minimized. The shortcomes of this type of scheduling is that it can not handle aperiodic and sporadic jobs satisfactorily.
The basic features of two important clock driven scheduler are discused here.

3.1.1 Table Driven Scheduling

Table driven schedular previously compute which task will execute at which instant an this information is stored in table at a time system is designed or configured.The schedular can make its own schedule at run time for set of task and saved in application table to be used by scheduler at run time.

3.1.2 Cyclic Schedular

Cyclic scheduler are simple easy and efficient to program. In cyclic scheduling the scheduling decision are made periodically rather than at arbitrary times. The scheduling decision are partition into intervals called frames. Scheduling decision are made at the beginning of the frame, there is no preemption within a frame. The performance of this type of schedular mainly depends on the size of the frame.

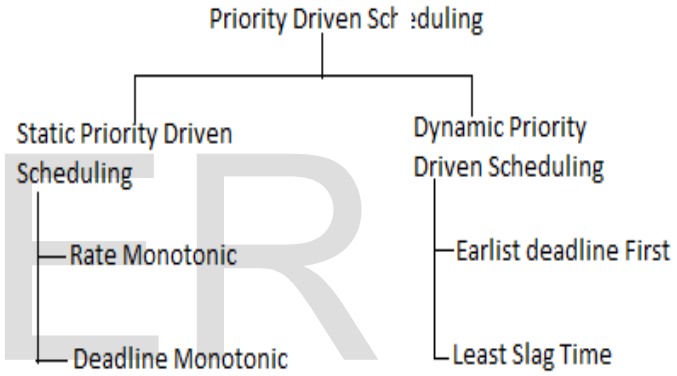
3.2 WEIGHTED ROUND ROBIN APPORACH

In round robin scheduling algorithum the ready jobs are executed in parts called quantum. All ready jobs are held in circular queue and schedule one after other in sequence order of their arrival time. Scheduled job will run for particular quantum. If it is not completed it is again added to the queue.
In weighed round robin scheduling the the quantum time that is the time allocated to the task is made veriable depending on the priority of the task. Thus it assigns higher quantum time to higher priority task.

Apart from the simplicity it also maximizes the proper utilization of resources. In many large-scale systems, it is desirable to trade some level of resource utilization for a simple and fast schedulability test. To achieve this goal, this paper proposes to employ the general framework developed in [5] to analyze the schedulability bound of the weighted round robin schedulars. Assigning weight to resources matches many resource allocation requirements in practice, i.e. assigning larger weights to users with greater resource consumption rates.

3.3 PRIORITY DRIVEN SCHEDULING APPORACH

In priority driven scheduling approach scheduling can be implemented by assigning priorities to jobs. A priority driven algorithum sorts the ready job at each time instants and schedules the heighest priority job. Hence priority driven algorithum is defined by the list of priorities assign to the job.



3.3.1 Rate Monotonic

Rate monotnic algorithum is important event driven scheduling algorithum and used in practical applications. In rate monotonic algorithum priorities are assign to the jobs based on their rate of occurance. Heigher the rate of occurance of job higher priority is assigned to the job, lower the rate of occurance lower the priority of job.

3.3.2 Deadline Monotonic

This algorithum assigns priorities to jobs according to their relative deadlines, shorter the relative deadline, the priority is higher [1].

3.3.3 Earliest Deadline First

The priority of each task is decided based on the value of its deadline. The task with nearest deadline is given highest priority and it is selected for execution. This algorithm is simple and proved to be optimal when the system is preemptive, under loaded and there is only one processor.

3.3.4 Least Slag Time First

It assigns priority based on the *slack time* of a process. Slack time is the amount of time left after a job if the job was started now. This algorithm is also known as Least Laxity First. It imposes the simple constraint that each process on each avail-

able processor possesses the same run time, and that individual processes do not have an affinity to a certain processor.

4 Analysis of Schedulability

4.1 Schedulability of Clock Driven Schedulars

In clock driven scheduling to get the best schedulabilty the following three constraint must be satisfied

Minimizing Context Switching: If task doesn't complete in single frame then context swiching occurs as task has to suspend and restart its execution which creates processing overload. To avoid this processing overload this constraint is imposed.To achive this the frame size should meet the condition $\max(\{ei\}) \leq F$ where F is frame size of task and ei is execution time of task Ti. This constraint impose lower bound on frame size.

Minimization of table size: This constraint requires that no. of entries in schedule table should be minimized in order to minimize storage requirement of schedule table. Some time it is required to divide major cycle in integral parts called minor cycle this would make size of schedule table large. We can formulate this constraint as [M/F]=M/F

If floor of M/F equals M/F then major cycle would contain an integral no. of frames.

Satisfaction of tasks deadline: This constraint defines an upper bound on frame size for a task Ti that if there is frame size larger than defined upper bound then tasks miss their deadline .Considering all task frame size must be smaller than

$$\max(\gcd(F, pi) + di)/2$$

Where pi and di are period and deadline of task Ti

EXAMPLE 1 A Cyclic scheduler is to be used to run the following set o periodic tasks on uniprocessor: T1 = (e1=1:p1=4) , T2 = (e2=1:p2=5) , T3 = (e3=1:p3=20) , t4 = (e4=1:p4=20) . Se;ect an appropriate frame size

Solution: - for the given set, an appropriate frame size is one that satisfies all the there required constraints. In the following, we determine a suitable frame size F which statisfies all the three required constraints

Constraint 1. Let F be an appropriate frame size, then max {ei}≤F . From this constraint we get F≥1.5

Constraint 2. The major cycle M for the given task set is given by M= LCM (4, 5, 20) = 20 M should be an integral multiple of the frame size F i.e. M mod F= 0. This consideration implies that F can take on the values 2,4,5,10,20 frame size of 1 has been ruled out science it would ommit the constraint 1

Constraint 3. To satisfy this constraint we need to check wether a selected frame size F satisfies the inequality 2F-gcd (F, pi) ≤ di for each pi

Let us try frame size 2

For F =2 and task T1:

$2*2-\gcd(2,4) \leq 4 \equiv 4 - 2 \leq 4$

Therefore, for p1 the inequality is satisfied

Let us try for F=2 and task T2:

$$2*2-\gcd(2,5) \leq 4 \equiv 4-1 \leq 4$$

Therefore for p2 inequality is satisfied

Let us try for F = 2and task T3:

$$2*2-\gcd(2,20) \leq 20 \equiv 4-2 \leq 20$$

Therefor for p3 inequilaty is satisfied

Let us try for F =2 and task T4:

$$2*2-\gcd(2,20) \leq 20 \equiv 4-2 \leq 20$$

Thus consrain 3 is satisfied by all task for frame size 2 So, frame size 2 satisfies all the three constraints. Hence 2 is feasible frame size

Let us try for frame size 4

Let us try for F =4 and task T1:

$$2*4-\gcd(4,4) \leq 4 \equiv 8-4 \leq 4$$

Therefore for p1 inequality is satisfied

Let us try for F =4 and task T2:

$$2*4-\gcd(4,5) \leq 5 \equiv 8-1 \leq 5$$

For p2 inequality is not satisfied we need not look further clearly for F=5 is not suitable frame size

Lets us now try for frame size 10

$$2*10-\gcd(10,4) \leq 4 \equiv 20-2 \leq 4$$

The inequlity is not satisfied for T1 we need not look any further clearly,F=10 is not suitable frame size

For F=20 and task T1, we have

$$2*20-\gcd(20,4) \leq 4 \equiv 40-4 \leq 4$$

Therefore F=20 is also not suitable.

So, only frame size 2 is suitable for scheduling.

Even though given example successfully find a suitable frame size that satisfied all three constraints it is quite probable that suitable frame size may not exist for many problems. In such a case to find a feasible frame size we might have to spilt the task(or a few task) that is causing violation of constraint into smaller subtask that can be scheduled in different frame.

4.2 Schedulability of Round Robin Schedulars

This portion of paper scedulability of weighted round robin algorithm is analyzed general methods and methodology used in []

The various terms and notation used are given here

T- Task that is to be schedule

Ti-Job in task set T where (i=0, 1, 2……..n)

To characterize the resource demand of task T analytically, we define $f(t)$, the **workload function** for T, as follows,

$f(t)$ = the summation of the sizes of all the jobs from T in $[0, t]$.

Similarly, to characterize the actual processor time received by task T, we define $g(t)$, the **service function** for T, as follows,

$g(t)$ = the total execution time rendered to jobs of task T during $[0, t]$

Service Constraint Function: A common alternative to $g(t)$ is the **generalized service constraint** introduced in [6], [7], and [8] (under the name of service curve). $G(I)$ is said to be a **generalized service constraint function** if for any $t \geq 0$, there exists $I \leq t$ that preserves the property

$$G(t) \geq f(t - I) + G(I)$$

Typically, we assume that $G(I)$ is non-decreasing and $G(0) \geq 0$. means that for any t , we can find I, where $0 \leq I \leq t$, such that 1) all the jobs released in $[0, t - I]$ have been served, and for jobs released in $[t - I , t]$, at least $G(I)$ amount of jobs have been served, as illustrated in Figure 2.

Figure 2: Components in the Generalized Service Constraint Function.

Normalized Deadline: To capture the tightness of the task deadline requirements of different systems, we define the **Normalized deadline ki** for *Ti* as follows:

$ki = Di / Si$,

Where *Di* is the relative deadline of task *Ti* and *Si* is the segment length in the s-shaped workload we follow the convention that for $i = 1, 2, n$

$i\ k = k$,

k can be viewed as the deadline using *S* as the measurement unit, and it characterizes tightness of the deadline requirements. The smaller the *k*, the more difficult it is to schedule the task. To measure the portion of time consumed in round robin operations relative to the length of rotation, we define the **overhead ratio** α as follows

$$\alpha = \tau_{0/TTRT}$$

where τ_0 is the overhead constant

TTRT is the target token rotation time,

To capture the effect of token rotation speed on the schedulability bound, we define the second system parameter **normalized token rotation frequency** as follows:

$$\Upsilon = \lfloor D_{min/TTRT} \rfloor$$

Where $D_{min} = \min(Di)$

Di- Relative deadline of job Ti and we assume that

$D\min \geq \text{TTRT}$

γ is the number of rounds the token rotates within a time interval of length Dmin. The larger the γ, the faster the token rotates.

To illustrate how the bound result can be used in practical systems, we introduce the following two examples.

Example 1: Consider a simple real-time robot controller who is responsible for the routing control of three data sampling robots, A, B, and C. The three robots are driving at different speeds and the controller communicates with the robot at a different frequency, i.e. once every 2.0, 5.0, and 20.0 seconds, for A, B ,and C, respectively. For the three robots, the controller takes 0.20, 0.30, and 0.60 seconds to finish the route selection and communication. The route selection and communication must finish within 4.0, 3.0, and 20.0 seconds for A, B, and C to avoid robot damages. The controller uses a WRR scheduling discipline with target token rotation time of 2.0 seconds per round. There is a cost of 0.002 seconds per context switching (changing the robot to be served). The weights for robots are assigned in normalized fashion. Now we need to decide whether the controller can finish all the routing tasks within their deadlines.

From the above description, we know there are three periodic tasks $\Gamma = \{T1, T2, T3\}$ where

$$Fi_{(I)} = \lceil I/Pi \rceil Ci$$

C1=0.20 P1=4.0 D1=4.0

C2=0.30 P2=5.0 D1= 3.0

C3= 0.60 P3=20.0 D3=20.0

For this set of tasks we have $\alpha = 0.001$, $\gamma = 2$, $k = 1$, and $\mu = 1$. To decide whether the task set is schedulable or not, we must calculate the total system workload rate as follows:

$$W(1,\Gamma) = \sum_{i=1}^{3} \frac{Fi(Di)}{Di} = \frac{c1}{p1} + \frac{c2}{p2} + \frac{c3}{p3} = \frac{0.20}{4} + \frac{0.30}{5} + \frac{0.60}{20} = 0.14$$

Schedulability bound can be calculated as

$$W^*([K]/K) = \frac{1}{1/\gamma + 1} \bullet (1 - n\alpha) \bullet \frac{1}{\mu} \bullet \min(1, k)$$

For this set of tasks we have $\alpha = 0.001$, $\gamma = 2$, $k = 1$, and $\mu = 1$. To decide whether the task set is schedulable or not, we must calculate the total system workload rate as follows:

By substituting $n = 3$, $\alpha = 0.001$, $\gamma = 2$, $k = 1$, and $\mu = 1$ into (a) we have a schedulability bound of 0.33 since 0.14<0.33, we conclude that task set is schedulable. [6]

5 Conclusion

From above discussion we can conclude that clock driven algorithm cyclic schedular are simple efficient and easy to program than table driven schedular as timer has to set every time task is set. For weighted round robin schedulars, the maximum amount of service every task can receive in each round is upper-bounded by its allocation. As such, no tasks can consume more service than what has been assigned.

Acknowledgment

Along the way, development of "Study and analysis of different real time task scheduling algorithums" was helped by discussion with R. R. Tuteja Asso. Prof at Prof. Ram Meghe Institute of Technology and Research, Badnera. The author would like to thank to R. R. Tuteja for her helpful comments and discussion on analysis portion of this paper.

References

[1] Jane W.S. Liu, *Real-Time Systems* , Pearson Education, India,pp. 121 & 26, 2001.

[2] N. Fisher et al., "The Non-preemptive Scheduling of Periodic Tasks upon Multiprocessors", Journal of Real-Time Systems, vol. 32, n. 1-2, pp. 9-20, 2006.

[3] Hyeonjoong Cho, Binoy Ravindran & E. Douglas Jensen " Optimal Real-Time Scheduling Algorithm for Multiprocessors " Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS'06)- 2006U.

[4] C. Devi and J. Anderson. Tardiness bounds forglobal edf scheduling on a multiprocessor. In *IEEE RTSS*, 2005.

[5] J. Wu, J.-C. Liu and W. Zhao, "On schedulability bounds of static priority schedulers," *Proc.* 11th IEEE Real-Time and Embedded Technology and Applications Symposium, San Francisco, CA, Mar, 2005, pp. 529-540.

[6] Jianjia Wu, Jyh-Charn Liu, and Wei Zhao Schedulability Bound of Weighted Round Robin Schedulers for Hard Real-Time Systems

[7] J. Y. Le Boudec and P. Thiran, *Network Calculus, a Theory of Deterministic Queuing Systems for the Internet*, New York: Springer-Verlag, 2001.

[8] S. Chang, "On deterministic traffic regulation and service guarantee: a systematic approach by filtering," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1096-1107, Aug. 1998.

[9] Rajib Mall "Real Time Ststem Theory And Practice",Version 2 IIT Kharagpur.

[10] M. Kaladevi and Dr. Sathiyabhama"A Coperative study of scheduling algorithums for real time task" Vol.1, No4,2010.

[11] Krithi Ramamritham and John