Design of a Feed back Real Time Scheduler based Standard EDF Scheduling algorithm and its Evaluation and Performance Comparison with the standard scheduler

Nikhil Bhargava,
Research Engineer, C-DOT, New Delhi
176, Gali Batashan Chawri Bazar
Delhi 110006
India
911123264334
nikhil_bhargav@rediffmail.com

Abstract

Scheduling multiple tasks in a resource constrained system is a core problem in the area of real-time system design. The well-known classical scheduling algorithms such as Rate Monotonic (RM), Spring and Earliest Deadline First (EDF) [15] are essentially open-loop techniques which perform well when the range of task parameters such as their execution times, periods and deadlines are either fixed or can be accurately modeled. However, in many applications like mobile computing, air- flight control and industrial robotics, the task parameters change randomly due to interaction with an unpredictable environment, dynamic data variation, system overheads and changing device characteristics. In order to address these issues, research focus has recently shifted to adaptive or stochastic scheduling techniques which can take into account the myriad of randomly changing parameters to tune the task schedules dynamically or probabilistically depending upon some feedback [2, 3, 4,5,6,13]. In this paper I have carry forwarded the initial work [15] of a feedback control scheduler based on tuning the execution time of the tasks. In this paper, I propose a scheme to maintain performance profile data in order to derive relevant information about system behavior and based on this, apply run-time predictive control to an adaptive scheduling system. This scheme dynamically tunes the scheduler using prediction techniques with minimum overhead. The PID controlled adaptive scheduler control the performance of the system by varying execution time or relative deadline or a combination of both these parameters.

Introduction

Real Time System can be defined, as “Any system where a timely response by the computer to external stimuli is vital within a fixed deadline is a Real Time System” This is one of many incomplete definitions of an RT system. In general, a real time system is one in which a substantial fraction of the design effort goes into making sure that task deadlines are met. Real time scheduling algorithms fall into two categories: static and dynamic scheduling. In static scheduling, the scheduling algorithm has complete knowledge of the task set and its constraints, such as deadlines, computation time, precedence constraints and future release times. The Rate Monotony (RM) algorithm and its extensions [12, pg 333] are static scheduling algorithms.

In dynamic scheduling, however, the scheduling algorithm does not have complete knowledge of the task set or its timing constraints. The Dynamic Scheduling can be further divided into two categories: scheduling algorithms that work in resource sufficient environments and those that work in resource insufficient environments. Under certain conditions, EDF [15] is an optimal dynamic scheduling algorithm in resource sufficient conditions.

1. Overview of EDF

A processor following the EDF algorithm always executes the task whose absolute deadline is the earliest. EDF is a dynamic priority-scheduling algorithm; the task priorities are not fixed but change on the closeness of their absolute deadline. EDF is also called the deadline-monotonic scheduling algorithm [12].

2. General Description of EDF

EDF or Earliest Deadline First algorithm is a dynamic-priority real time scheduling algorithm for uniprocessor systems where task priorities are dynamically changing based on their relative deadline. The Following assumptions are made for EDF algorithm.

A1. No task has any non-preemptible section and the cast of preemption is negligible.
A2. Only processing requirements are significant; memory, I/O etc are negligible.
A3. All tasks are independent; there are no precedence constraints.

It becomes clearer with following example:

Task	Arrival Time	Execution Time	Absolute Deadline
T_1	0	10	30
T_2	4	3	10
T_3	5	10	25

Table 1 Task Parameters

When T_1 arrives, it is the only task waiting to run and thus start executing immediately. T_2 arrives at time 4; since it $d_2<d_{1,}$ it has higher priority than T_1 and preempts it. T_3 arrives at t=5; since $d_3<d_{2,}$ it has lower priority than so T_2 and must wait for T_2 to finish. When T_2 finishes at t=7, T_3 starts and runs till t=15 at which point T1 resume its execution.

3. Applications of EDF

EDF [12, ch 3] is an optimal uniprocessor-scheduling algorithm. That is, if EDF cannot feasibly schedule any task set on a uniprocessor system, than there is no algorithm that can schedule it. Test for schedulability for a task set as per EDF is given in [12].
EDF finds wide application in fields like telecommunication systems, defense systems and multimedia. Many real time operating systems like RTLinux, RTAI etc are supporting EDF [13, 14].

4. Feedback Control Earliest Deadline First (FC-EDF)

4.1 Concept of FC-EDF

A typical feed back control system is composed of a controller, a plant to be controlled, actuators, and sensors. It defines a controlled variable, the quantity of output that is to be measured and controlled. The set point represents the correct value of the controlled variable. The difference between the current value of the controlled variable and the set point is the error. A feedback system is best explained in [15, 11]

4.2 Overview of FC-EDF

The requirements of an ideal soft real time scheduling algorithm should be to

- Provide (soft) performance guarantees to admitted tasks i.e., maintain low miss ratio among admitted tasks.
- Achieve high system throughput.

The choice of controlled variable depends upon the system's primary objective where as the choice of manipulated variable depends upon the degree of dependence of controlled variable upon it. For most of the real-time systems, the performance heavily depends upon the fraction of tasks that miss their deadline. To satisfy these two requirements both versions of FC-EDF chooses System deadline miss ratio as Controlled variable. System deadline miss ratio is defined the ratio of tasks that miss their deadline. A small but nonzero (I have taken 0.1) value is taken as set point. In an unpredictable environment, it is impossible to achieve 100% utilization and 0% miss ratio all the time and hence a trade off between the two is unavoidable. Again out of the possible choices of manipulated variable like periods and deadline, the requested CPU utilization i.e., task load affects the deadline miss ratio the most. So I have chosen requested load as manipulated variable. Thus I choose the requested CPU utilization, i.e., total CPU utilization requested by all the accepted tasks in the system, as the manipulated variable. The logic behind it is that EDF can guarantee a miss ratio of 0% given the system is not overloaded and miss ratio increases as system load increases. Third, I need to design the mechanisms (i.e., actuators) to control the scheduler to manipulate requested utilization. An admission Controller and Service level Controller is included in the FC-EDF scheduler as the mechanisms to do so. The admission Controller can control the flow of workload flowing into system where as SLC adjusts the workload inside the system.

My approach is to regard a scheduling system a feedback control system the scheduler as the controller. I have used the most popular feed back control: PID i.e. Proportional Integral Derivative control. The PID controller periodically monitors the miss ratio (t) and computes the control ΔCPU (t) in terms of the requested CPU utilization with the following control formula

$\Delta CPU\ (t) = C_p\ error\ (t) + C_I \sum_{IW} error\ (t) + C_d\ (error\ (t) - error\ (t-DW))/DW$

Where,

error (t) = Miss $Ratio_s$- Miss Ratio (t),

SP, C_p, $C_{d,}$ C_I , IW and DW are tunable parameters of PID controllers and SP is sampling period.

4.3 Design of FC-EDF in general

This is based on Feedback Control EDF (FC-EDF) algorithm, which integrates the PID control with an EDF scheduler [1, 12, 15].

4.3.1 Task Model

My task model assumes all tasks have soft deadlines and all tasks are independent. Four sources are taken in all. Each source is characterized by a period, which is taken, as all tasks are periodic. Period of each source is taken as uniform poison distribution upon truly uniform variable. Execution time of each source is taken as a function of uniform distribution of period characterizing the source. Each task T_i submitted to the system is described with a tuple (N, P, D, AET, EET, DONE). Here N stands for the name of the task, P stands for the period of the task, D stands for its relative deadline i.e. deadline form the time of issue, AET stands for average execution time of the task, EET stands for estimated execution time of each task and DONE stands for execution done so far. Average execution time AET is calculated from Worst-case EET i.e. WCET and Best case execution time i.e. BCET. Following formulae are used:

$WCET_0 = P/10$

$WCET_1 = WCET_0/2$

$BCET_0 = WCET_0/3=P/30$

$BCET_1 = BCET_0/2$

$AETi = (WCETi + BCETi)/2$

$EETi = etf * AETi$

for *i* varing from 0 to1 for each task produced

Where,

P is period of the task and i is the service level of each task. etf (execution time factor) can be tuned to vary the accuracy of this estimation method. An etf value of greater than 1.0 means that estimated execution time is greater than average time whereas an etf value less than 1.0 means the other way around. Thus it is treated as tunable parameter. In all these experiments etf lies in interval [0.5,1.5].

In the design of deadline based Feedback EDF major changes are made in the task model used in simple EDF. Here I have introduced three service levels to enhance the dynamic nature of this algorithm. Here deadline of each task is chosen from a set of 3 values {d_{min}, d, d_{max}) where,

$d_{min} = P/ 2$

$d = P$

$d_{max} = P*2$

Another change brought in this algorithm is that all tasks are executed in bursts exactly in the same way as they are executed in a real system. So I have introduced a factor of CPU_BURST is included in the deadline-based simulation. The tasks are executed for less than or equal to this CPU_BURST time interval. This also adds to the dynamic nature of this scheme [7].

This is essentially useful for those systems time of completion can vary over a range without affecting the system performance substantially. This is also useful in case of many multimedia systems where as QoS specifications can be given in terms of intervals rather than a single value.

4.3.2 PID Controller

The PID controller is the core of this version of FC-EDF. It maps the miss ratio of the accepted tasks to the change in the requested utilization (i.e., error) to the change in requested CPU utilization (i.e., control signal) so as to drive the miss ratio back to set point.

The PID controller periodically monitors the miss ratio (t) and computes the control ΔCPU (t) in terms of the requested CPU utilization with the following control formula

$\Delta CPU\ (t) = C_p\ error\ (t)+C_l \sum_{IW} error\ (t)+C_d\ (error\ (t)-error(t-DW))/DW$

Where,
$error\ (t) = Miss\ Ratio_s - Miss\ Ratio\ (t)$, *SP*, C_p, C_d, C_l , *IW* and *DW* are tunable parameters of PID controllers and SP is sampling period. Detailed working of PID controller is given in [15].

4.3.3 Service Level Controller

The service Level Controller (SLC) changes the requested utilization in the system by adjusting the service levels of the accepted tasks.

4.3.4 Cost function used in SLC

The pivotal point on which the design of my scheme's cost function is based is which task is to be chosen for adjusting the load internally and what is amount of CPU load to be tuned. The amount of CPU load to be adjusted is described in the next few lines. If the task T_i having service level as i, has total execution time as ET_i and it has already run for ET_{done} and the current deadline of T_i is $d_{current}$ and I am changing its service level to let say j, than

If dj>di and $d_{current}$ >dj then no change in $d_{current}$

Else $d_{current}$= dj - (di - $d_{current}$)

And total change in load ΔCPU' = (ET_i - ET_{done})/d_{ii} - (ET_i - ET_{done})/d_{ij}

4.4.5 Admission Controller

The Admission Controller (AC) is brought into action if and only if the Service Level Controller is not able to completely adjust the load by itself. It controls the flow of workload into the system. When a new Task T_i is submitted to the system, AC decides whether it should be accepted or not. Given the total CPU requirement CPU (t) and CPU requirement of the incoming task, the Admission controller admits T_i with service level k if k is the highest level that satisfies CPU (t)+ET_i < 1.00; T_i is rejected if it cannot be admitted even with lowest level.

5. System Model

The analytical model [15] is used in the simulation of the two versions of feed back EDF algorithms. The basic Executor is simple EDF system. The workload consists of a set of 20 independent periodic tasks. 4 sources are used to produce 5 tasks each. As the system schedules the tasks, it also decrements the relative deadline of each task in the ready queue. A task is said to be missed when its relative deadline becomes 0. It is than immediately aborted and it starts attempts to reenter in the ready queue. Miss ratio is computed as the ratio of missed tasks to the Task set strength. The simulator consists of 9 components.
The following table shows the parameters used in both algorithms based on FC-EDF.

Parameter	Value
Set Point	0.1
SP	20
IW	10(SP)
DW	1(SP)
C_p	0.05
Cl	0.01
Cd	0.001
CPU BURST	0.30
etf	[0.5,1.5]

Table 2 System parameters used in simulation experiments

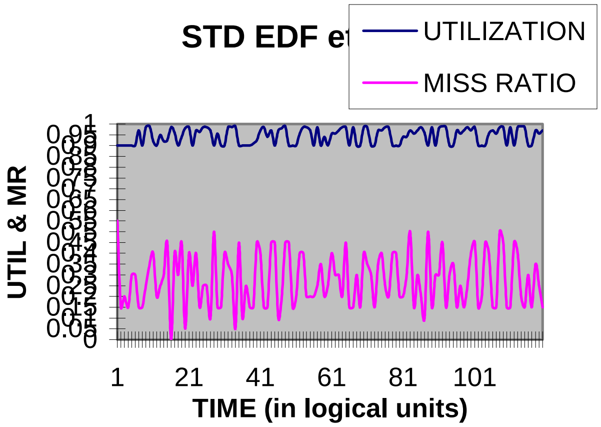
6. Results

Following are the graphs of results from simulation of standard EDF, FC-EDF based on the execution time and FC-EDF based on deadline. Dynamic simulation is done with CPU BURST taken as 0.30 and a total of 10 readings are taken. The final graphs are drawn with the average data. Graphs 1a, 1b, 1c shows the behavior of the system based on standard EDF algorithm for values of etf as 0.7, 1.0, 1.5 respectively. Graphs 2a, 2b, 2c shows the behavior of the system based on FC-EDF algorithm using execution time, for values of etf as 0.7, 1.0, 1.5 respectively. Graphs 3a, 3b, 3c shows the behavior of the system based on FC-EDF algorithm using deadline, for values of etf as 0.7, 1.0, 1.5 respectively.
Graph 1a, 2a and 3a show the dynamic simulation of standard EDF, FC- EDF using execution time and FC-EDF using deadline for etf = 0.7. In case of simple EDF system load remains at a constant value of 0.93333 and mr varies from 0 to 0.75. mr higher at those values of time which are multiples of two or more source periods as in t=20, 40, 60. In case of 1b, the system starts at normal assessment and mr=0; as the load increases the mr also increases and this trend continues till t =10. At t=10, mr goes to 0.25, then PID controller comes into action and adjusts the load internally to 0.875 and

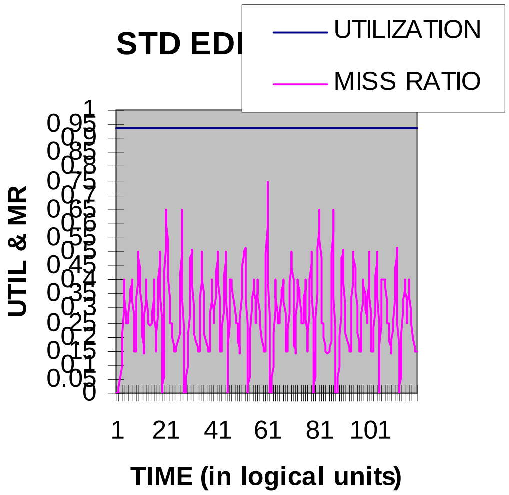
brings the mr down to 0.15. Again at t=40 and t=70 mr is reduced by calling SLC first and then AC to shed tasks to decrease the load. For e.g. at t=70, mr shoots up to 0.75, then first SLC is called and then AC is called to adjust the load to 0.85 and as a result mr is reduced to 0.

In the case of 3a, the variance between mr and set point is high. The system starts with normal approach. As load increases to 0.975, the mr shoots up to 0.25. Then immediately, SLC comes in to play and it decreases the load to 0.875, which in turn reduces the mr to 0.1.

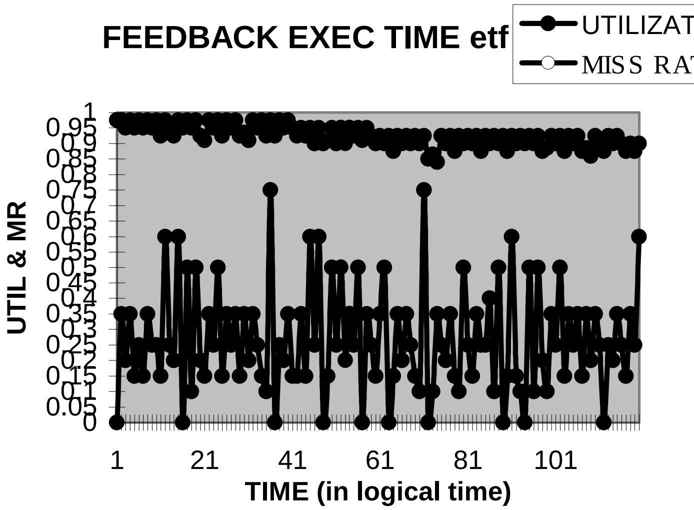
The result clearly shows that standard EDF is worst of the three. For etf=0.7, FC-EDF based on execution time performs best where as for etf=1.0 and 1.5, deadline based EDF-FC performs best.



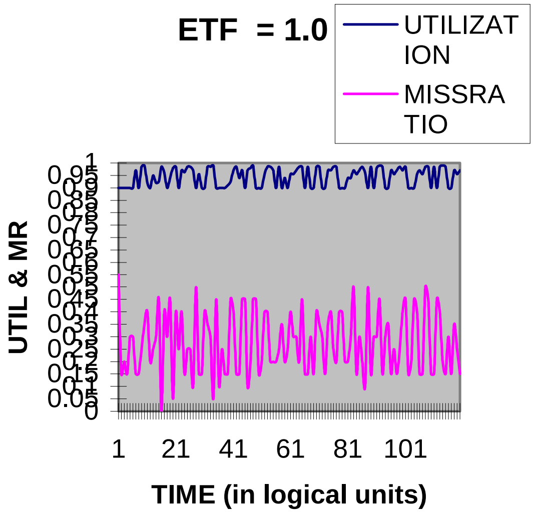
Graph 1c Standard EDF with etf = 1.5



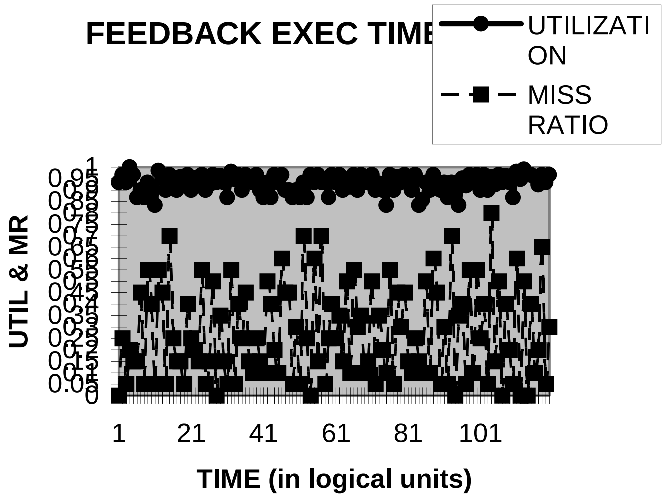
G 1a Standard EDF with etf = 0.7



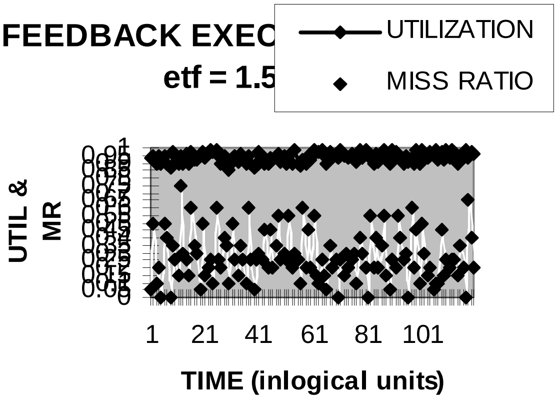
G 2a F back Exec Time Based EDF etf = 0.7



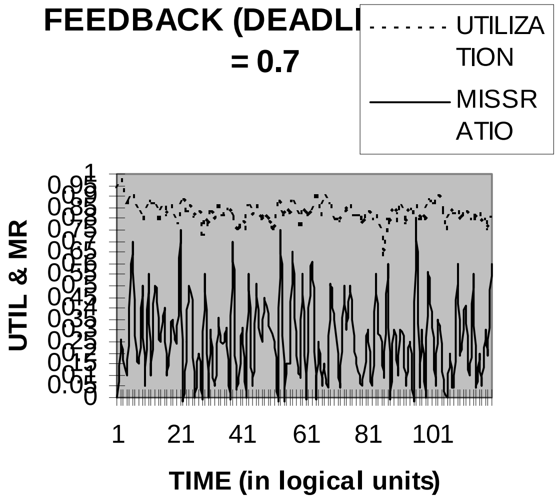
G 1b Standard EDF with etf = 1.0



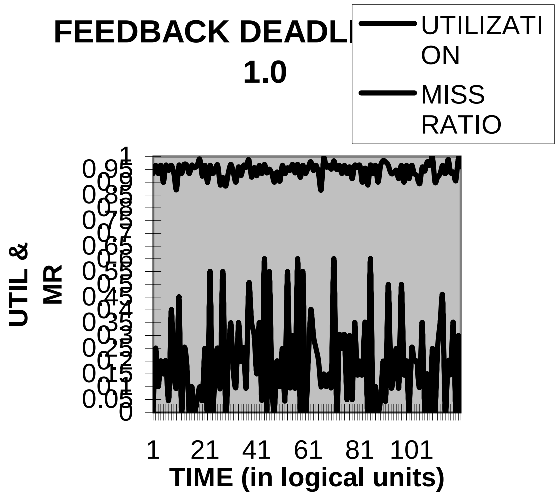
G 2b F back Exec Time Based EDF etf = 1.0



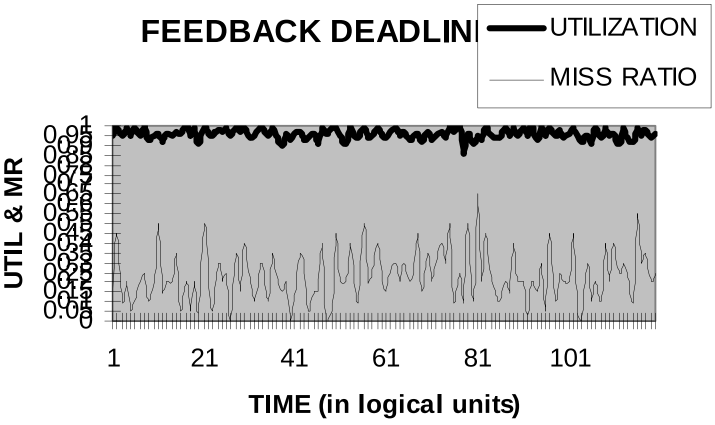
G 2c Feed Exec Time Based Edf etf = 1.5



G 3a F back Deadline Based EDF etf = 0.7



G 3b Feedback Deadline Based Edf etf = 1.0



G 3c Feedback Deadline Based Edf etf = 1.5

Conclusions

The [1] work was an excellent step in providing a flexible solution to most of the real time problems that have unpredictable characteristics but it has some shortcomings. First of all, the task model chosen is based on uniform distribution, which means that all the tasks produced from a source have identical parameters i.e., period, worst-case execution time, best-case execution time etc. Secondly, the tasks here are truly based on static priorities i.e., they lack a truly dynamic setup.

In the deadline based FC-EDF, I have taken an exponentially distribution over a truly generated random variable. Moreover, the task model is designed to give true dynamic nature to the tasks.

Further more in this paper, I have systematically reviewed the concepts of Earliest Deadline First real time scheduling algorithm and of feedback system in general. In the end I have compared two different EDF based real time scheduling algorithms and simple EDF algorithm in terms of CPU utilization and system miss ratio. Of all the three algorithms, deadline based FC-EDF performs best when task characteristics predictability is low.

Another important outcome from the results is that system miss ratio, which is the most important aspect of soft RTOS systems is directly controlled by the scheduler. Although a dynamic simulation is used with drastic changing system load conditions, both versions of FC-EDF i.e., execution time based and deadline based FC-EDF are able to maintain satisfactory deadline miss ratio and high system utilization.

Future scope

RTOS field is a never-ending field. The work done so far is just a tip of iceberg. I have tried to present simulation experiments that verify the advantage of feedback control scheduling in highly unpredictable environments. Another area of improvement is in the PID controller. Here in these experiments I have used 1_{st} degree control. The higher degree control used, greater reduction would occur in variance of system deadline miss ratio but it would increase the complexity and cost of algorithm.

Moreover, the deadline based FC-EDF can find vast application in multimedia application where task deadlines are not strict and one can tune the deadline without losing much on its QoS.

In the deadline based FC-EDF I have used three values of deadline i.e. d_{min}, d, $d_{max.}$ If I extend this to a range of deadline i.e., [d_{min}, d_{max}] than it will lead to better performance for many digital control systems which are usually robust.

Further improvement in this algorithm is to have three level feedback control where in CPU load can be adjusted by changing the execution time of the task, or by changing the deadline of the task or a combination of the two methods. This will give much more flexibility to this algorithm but it would make the system too complicated and would add to both the space and time complexity.

References

1. C.L`. Liu and J.W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” Journal of the ACM, vol 20(1), pp 46-61, 1973
2. T.W. Kuo and A. K. Mok, “Incremental reconfiguration and load adjustment in adaptive real-time systems,” IEEE Trans., Computers, vol 46, pp-1313-1324, Dec 1997
3. M. Hamdaoui and P. Ramanathan, “Evaluating dynamic failure probability for streams with (m,k)-firm deadlines,” IEEE Trans, Computers, vol 46, pp1313-1324, Dec 1997.
4. S. Chakraverty and C.P. Ravikumar, "A stochastic framework for cosynthesis of real-time systems," Proc of LCTES-2000, Canada, June 2000, pp 51-59.
5. J.A. Stankovic, C. Lu, Sang H. Son and G. Tao, “The case for feedback control for real-time scheduling”, in Proc. EuroMicro Conf. on Real-time Systems, June 1999.
6. D.C. Ku and G.D. Micheli, "Relative scheduling under timing constraints: algorithms for high-level synthesis of digital circuits," IEEE Trans. Computer Aided Design, vol11,no6, pp 696-718, June 1992.
7. J.W.S. Liu et al, "Algorithms for scheduling imprecise computations,"IEEE Computer, vol 24, no 5, May 1991.
8. H.S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms,"IEEE Trans Software Engg., vol 3, pp85-93, Jan 1977.
9. W.W. Chu and L.M.Y Lan,"Task allocation and precedence relations for distributed real time systems,"IEEE Trans on Comput. Vol 36, no 6, pp667-670, June 1987.
10. A.C Shaw,"Reasoning about time in higher level language software,"IEEE Trans Software Engg, vol 15, pp 875-889, 1989.
11. Olis Rukin, “The design of automatic control systems”, Artech House, 1993
12. C.M.KRISHNA and KANG G. SHIN ‘s REAL TIME SYSTEMS International Editions 1997 published by McGRAW-HILL.
13. World Wide Web (Site of Educational Os i.e., EDU/RTOS)
14. Modern Operating System by Tanenbaum, 2001 Indian low cost Edition.
15. Chenyang Lu John A., Stankovic Gang, Tao Sang H.Son: “ Design and Evaluation of a Feedback Control EDF Scheduling Algorithm“ IEEE RTAS June 1999.

Glossary

➢ EDF	: Earliest Deadline First algorithm
➢ RM	: Rate Monotony algorithm.
➢ RTS	: Real Time Scheduling Algorithm
➢ TSS	: Time Sharing Scheduling Algorithm
➢ PID	: Proportional Integral Control
➢ P_{lcm}	: LCM of Periods of all the Tasks in the Task Set
➢ P_i	: Period of task i.
➢ e_i	: Execution time of task i.
➢ SP	: Sampling Period
➢ SLC	: Service Level controller
➢ AC	: Admission Controller
➢ PID	: Proportional Integral Control
➢ u	: CPU utilization