

A Feedback-Based Adaptive Algorithm for Combined Scheduling with Fault-Tolerance in Real-Time Systems

Suzhen Lin and G. Manimaran

Dept. of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011, USA
{linsz, gmani}@iastate.edu

Abstract. In this paper, we propose a feedback-based combined scheduling algorithm with fault tolerance for applications that have both periodic tasks and aperiodic tasks in real-time uniprocessor systems. Each periodic task is assumed to have a primary copy and a backup copy. By using the rate monotonic scheduling and deferrable server algorithm, we create two servers, one for serving aperiodic tasks and the other for executing backup copies of periodic tasks. The goal is to maximize the schedulability of aperiodic tasks while keeping the recovery rate of periodic tasks close to 100%. Our algorithm uses feedback control technique to balance the CPU allocation between the backup server and the aperiodic server. Our simulation studies show that the algorithm can adapt the parameters of the servers to recover the failed periodic tasks.

Keywords: Real-time systems, feedback-based scheduling, deferrable server algorithm, fault-tolerance.

1 Introduction

Real-time systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced [1]. Most real-time systems involve both periodic tasks and aperiodic tasks. Usually, periodic tasks are more important than aperiodic tasks. Due to the critical nature of tasks in a real-time system, it is essential that every periodic task admitted in the system completes its execution even in the presence of faults. Therefore, fault tolerance is an important requirement in such systems.

To address the fault tolerance problem, the primary/backup technique is used [2]. Each periodic task is assumed to have a primary copy and a backup copy. If the primary copy fails, the backup copy will be scheduled and then executed.

In order to schedule both periodic and aperiodic tasks in real-time systems, the simplest approach is to create a periodic server, with a certain computation time and period, whose purpose is to serve one or more aperiodic tasks each time it is invoked. In our paper, the concept of deferrable server algorithm [3] is used. We create two deferrable servers, one for serving aperiodic tasks and the other for executing backup copies of failed periodic tasks.

In scheduling systems, faults cause the performance of the system unpredictable. Control theory is one of the successful areas in addressing performance in the presence

of uncertainty [4]. To the best of our knowledge, ours is the first work that uses feedback control theory to address the problem of combined scheduling with fault tolerance.

In this paper, we propose a feedback-based combined scheduling for uniprocessor real-time systems. By feeding back the performances, we adjust the utilization capacity for backup deferrable server. Specifically, we adjust the period of the backup deferrable server. By adjusting the period, the utilization capacity and priority are both adjusted. Since the CPU resource is limited, we cannot adjust the utilization for a server without changing the utilization of the other server. If the utilization capacity of the backup server is increased (decreased), then the utilization capacity of the aperiodic server must be decreased (increased). Our goal is to maximize the schedulability of aperiodic tasks while keeping the recovery rate of periodic tasks close to the desired value (100%).

The rest of the paper is organized as follows. In Section 2, the related work is discussed. In Section 3, the feedback-based fault-tolerant scheduling algorithm is proposed. In Section 4, we validate the result through simulation. Finally, Section 5 concludes the paper.

2 Background

In this section, we will introduce the system model for our algorithm, feedback control technique, and deferrable server [3] concept.

2.1 System Model

Figure 1 shows the system model. In our model, we make following assumptions.

- The system is uniprocessor real-time system.
- Tasks arrive at the system are periodic and aperiodic tasks. Each periodic task T_i is denoted by $T_i = \langle c_i, p_i \rangle$, where c_i and p_i are the computation time and period of T_i respectively. Each aperiodic task T_j is denoted by $T_j = \langle a_j, c_j, d_j \rangle$, where a_j , c_j and d_j are the arrival time, computation time and deadline of task T_j respectively.
- We assume a primary-backup scheduling for periodic tasks [6][7][8] wherein a backup copy of a task is executed if its primary fails the acceptance test.
- We assume transient faults for the periodic tasks and the faults are independent. The failure of aperiodic tasks is not considered as they are not very critical.
- There exists a fault-detection mechanism such as acceptance tests to detect both processor failures (transient) and software failures.
- We use the concept of deferrable server to serve aperiodic tasks and the backup copies. We have n periodic tasks: T_1, T_2, \dots, T_n , and we also create an aperiodic deferrable server ($T_{as} = \langle c_{as}, p_{as} \rangle$) to serve aperiodic tasks and another backup deferrable server ($T_{bs} = \langle c_{bs}, p_{bs} \rangle$) to serve backup copies of periodic tasks. c_{as} and p_{as} are the computation time budget and period of the aperiodic deferrable server respectively. c_{bs} and p_{bs} are the computation time budget and period of the backup deferrable server respectively. When a failure is detected in a primary copy, the backup copy is put into the backup task queue of the backup deferrable server.
- The scheduling algorithm used is rate monotonic scheduling (RMS) [5] algorithm. The RMS algorithm schedules periodic tasks and the server instances.

- Feedback control technique is used to adjust the utilization allocated to the aperiodic server and the backup server. The adjustment is based on the fault rate and recovery rate of the periodic tasks.

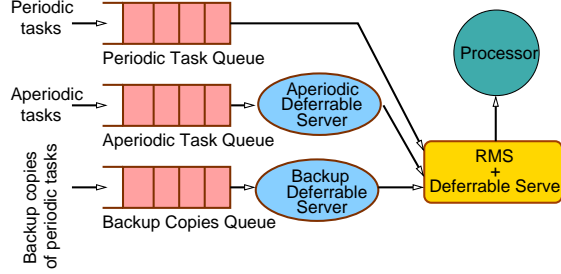


Fig. 1. System model

We define *miss ratio* (MR) of periodic tasks and aperiodic tasks and *recovery rate* (RR) of periodic tasks for the model. Miss ratio of periodic tasks is the ratio of the number of periodic tasks that miss their deadlines to the number of periodic tasks admitted into the system. Similarly, miss ratio of aperiodic tasks is defined for aperiodic tasks. Recovery rate of periodic tasks is the ratio of the number of recovered tasks (backup copies that meet their deadlines) to the number of failed primary copies. The desired MR of the periodic tasks is zero and the desired recovery rate is 100%.

Deferrable Server Algorithm: In the deferrable server algorithm, a periodic task known as a *deferrable server* [3] is created to serve aperiodic tasks. When the server is invoked but no aperiodic tasks are outstanding, the server does not execute but defers its assigned time slot. When an aperiodic task arrives, the server is invoked to execute aperiodic tasks and maintains its priority. The computation time budget for the server is replenished at the beginning of each period of the server.

For periodic tasks and deferrable server, we use the following schedulability checks. Assume that the tasks are ordered in non-increasing order of priority, that is, we have $T_1, T_2, \dots, T_m, T_s, T_{m+1}, \dots, T_n$, where T_s is the server. For schedulability check of each periodic task T_j that have higher priorities than the server, Equation 1 is used. For the server, Equation 2 is used, where c_s and p_s are computation time budget and period of the server respectively. For each task T_j that have lower priorities than the server, Equation 3 is used.

$$\sum_{i=1}^j \frac{c_i}{p_i} \leq j(2^{1/j} - 1) \quad (1)$$

$$\sum_{i=1}^m \frac{c_i}{p_i} + \frac{c_s}{p_s} \leq (m+1)(2^{1/(m+1)} - 1) \quad (2)$$

$$\sum_{i=1}^j \frac{c_i}{p_i} + \frac{c_s}{p_s} + \frac{c_s}{p_j} \leq (j+1)(2^{1/(j+1)} - 1) \quad (3)$$

2.2 Feedback Control

Figure 2 shows a typical control system, consisting of a controller, a plant to be controlled (controlled system), sensors, and actuators [9]. The system defines four variables: (1) exogenous variables are inputs from outside of the system, e.g., set points

(desired values of the output values) and disturbance. (2) regulated variables are the output values that the system wants to regulate. (3) measured variables are values that the sensors measure. (4) control variables are the inputs of the actuators. The actuators will actuate the plant based on the control variables. Besides, The system also defines the error, which is the difference between the set points and the feedback information.

The system works as follows: The sensors periodically monitor the regulated variables and get the error to feed to the controller. The controller computes the required control, using the control function of the system, based on the error. The actuators change the control (manipulated) variables to control the system.

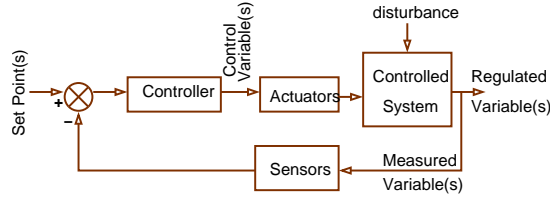


Fig. 2. Control system

3 Proposed Combined Scheduling using feedback

The goal of the proposed algorithm is to maximize the schedulability of aperiodic tasks while keeping the recovery rate of primary copies of periodic tasks close to 100%. Since aperiodic tasks are less important than periodic tasks, we give the aperiodic deferrable server a period larger than all periodic tasks, that is, the aperiodic deferrable server has lowest priority. According to Section 2.1, the CPU utilization allocated to the backup deferrable server is $u_{bs} = \frac{c_{bs}}{p_{bs}}$, the utilization that the periodic tasks need is $u_p = \sum_{i=1}^n \frac{c_i}{p_i}$, and the utilization allocated to aperiodic tasks is $u_{as} = \frac{c_{as}}{p_{as}}$.

The problem is how to allocate CPU utilization to the periodic tasks, aperiodic deferrable server and backup deferrable server. Since there is no way to know exactly how many tasks and which tasks will fail until the failures happen, we need to estimate u_{bs} to allocate resource. When an application starts, we guarantee enough capacity to the periodic tasks, allocate small capacity to the backup deferrable server, and the remaining capacity is allocated to the aperiodic deferrable server. When faults occur, we increase u_{bs} . u_{bs} can be increased by increasing c_{bs} or decreasing p_{bs} . Since decreasing p_{bs} can not only increase u_{bs} but also increase the priority of the backup deferrable server, we adjust the period of the backup deferrable server to change u_{bs} . The failure information can be obtained by measuring the recovery rate of failed periodic tasks in a past interval. The remaining utilization is assigned to the aperiodic deferrable server with lowest priority. When u_{bs} changes, we change the capacity of the aperiodic deferrable server to achieve suitable CPU utilization. The priority of the aperiodic deferrable server is fixed to be the lowest priority, thus p_{as} is fixed and c_{as} will be adjusted.

3.1 Admission Test

During the adjusting of the utilization of servers, the situation may happen. At the beginning, the period of the backup deferrable server is larger than the aperiodic deferrable

server; Later, the period of the backup deferrable server may become smaller than the aperiodic deferrable server. In the admission test, when $p_{bs} > p_{as}$, Equation 4 is used, otherwise, Equation 5 is used.

$$\sum_{i=1}^n \frac{c_i}{p_i} + \frac{c_{as}}{p_{as}} + \frac{c_{bs}}{p_{bs}} + \frac{c_{as}}{p_{bs}} \leq (n+2)(2^{1/(n+2)} - 1) \quad (4)$$

$$\sum_{i=1}^n \frac{c_i}{p_i} + \frac{c_{bs}}{p_{bs}} + \frac{c_{as}}{p_{as}} + \frac{c_{bs}}{p_{as}} \leq (n+2)(2^{1/(n+2)} - 1) \quad (5)$$

All the utilization adjustments must satisfy Equation 4 or Equation 5. That is, every time we change the period of the backup deferrable server, we need to use Equation 4 or Equation 5 to decide the value of c_{as} .

3.2 Feedback Control Mechanism

The system architecture is shown in Figure 3. The measured variable is recovery rate of failed periodic tasks. The set point is desired value of the recovery rate. The control variable is the utilization of backup sever. The regulated variable is the recovery rate. However, we notice that if we assign the set point to a desired value of 100%, the period of the backup deferrable server will not change when the fault rate decreases. To avoid such undesired situation, we also measure the failure rate of the periodic tasks and feedback this information to the controller to adjust the period of the backup deferrable server. The controller algorithm is shown in Equation 6. In Equation 6, when the recovery rate at time instant $k-1$ (RR_{k-1}) is less than the set point ($RR_s = 100\%$), the term $-k_r(RR_s - RR_{k-1})$ contributes a negative part to the period at time instant k (p_{bsk}) and hence the period of the backup deferrable server will decrease. This means the utilization allocated to the backup deferrable server increases. Therefore backup copies will get more chances to be executed, and the recovery rate will increase. In Equation 6, we also compare the measured failure rate (FR_{k-1}) with the average failure rate (FR_a) in the past t intervals. If the failure rate is less than FR_a , the term $k_f(FR_a - FR_{k-1})$ will contribute a positive part to the period. This will increase the period, and hence the utilization allocated to the backup deferrable server will decrease.

$$p_{bsk} = p_{bs(k-1)} - k_r(RR_s - RR_{k-1}) + k_f(FR_a - FR_{k-1}) \quad (6)$$

After getting p_{bs} , c_{as} can be calculated; when $p_{bsk} > p_{as}$, we use Equation 7, otherwise, we use Equation 8. The subscript k is time instant and U_{rms} is the utilization bound of RMS.

$$c_{ask} = \frac{U_{rms} - u_p - u_{bsk}}{\frac{1}{p_{as}} + \frac{1}{p_{bsk} c_{bs}}} \quad (7)$$

$$c_{ask} = (U_{rms} - u_p - u_{bsk} - \frac{1}{p_{as}}) \times p_{as} \quad (8)$$

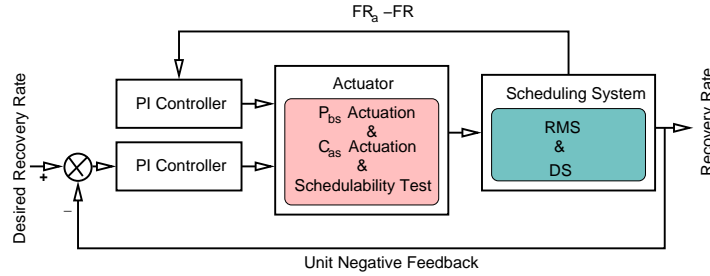


Fig. 3. System architecture

Note that when we use controller to adjust the parameters of the two servers, negative values of p_{bsk} and c_{ask} should not appear. There may be two reasons of getting negative values: One is that the parameter of the controller is too large which results in the task parameters going to negative values; The other reason is that the fault rate is very high, and the system resource does not have enough capacity to execute all the tasks. We assume low fault rate, so the latter case will not appear. To deal with the formal case, we need to choose the controller parameters carefully such that the task parameters will not go out of their reasonable ranges. When faults occur, the period of backup deferrable server decreases. According to Equation 5, if the period decreases to a value that c_{as} has to be zero such that Equation 5 can still be satisfied, p_{bs} can not be decreased any more. From Equation 5, we get Equation 9.

$$\frac{c_{bs}}{p_{bs}} + \frac{c_{as}}{p_{as}} = U_{rms} - u_p - \frac{c_{bs}}{p_{as}} \quad (9)$$

We require that $c_{as} \geq 0$, then we have Equation 10.

$$p_{bs} \geq \frac{c_{bs}}{U_{rms} - u_p - \frac{c_{bs}}{p_{as}}} \quad (10)$$

In order to let p_{bs} satisfy Equation 10, we must be careful with selecting k_r and k_f . If these two values are too large, it will cause large fluctuation and unreasonable task parameters before the system becomes stable. Thus, we need to confine the value of k_r and k_f such that p_{bs} will be greater than $p_{bst} = \frac{c_{bs}}{U_{rms} - u_p - \frac{c_{bs}}{p_{as}}}$. The maximum change in p_{bs} is $p_{bsmax} = p_{bso} - p_{bst}$, where p_{bso} is the original period assigned to the backup deferrable server. Thus we have Equation 11.

$$-k_r \Delta RR + k_f \Delta FR \leq p_{bsmax} \quad (11)$$

ΔRR and ΔFR are the change in RR and FR respectively in the corresponding situation. Assume $\Delta RR = -1$ and $\Delta FR = 1$, we have a conservative restriction on k_r and k_f as shown in Equation 12, where $k_r > 0$ and $k_f > 0$.

$$k_r + k_f \leq p_{bsmax} \quad (12)$$

When we choose the parameters for the controller, Equation 12 must be satisfied.

4 Simulation Studies

The simulation studies were conducted in two parts. The first part shows the effect of feedback adjustment by injecting fault at the beginning. The second part shows the steady state performance for different fault rate.

The periodic tasks used for the simulation are generated as follows:

- The computation time (c_i) is uniformly chosen between 10 and 20.
- The period of task T_i is uniformly chosen between $(6 * c_i)$ and $(8 * c_i)$.
- The backup copies have identical characteristics of their primary copies.

The aperiodic tasks used for the simulation are generated as follows:

- The computation time (c_i) of task T_i is uniformly chosen between 10 and 20.
- The deadline of T_i is uniformly chosen between $(r_i + 30 * c_i)$ and $(r_i + 40 * c_i)$.
- The inter-arrival time of tasks is exponentially distributed with mean $\theta = 50$.

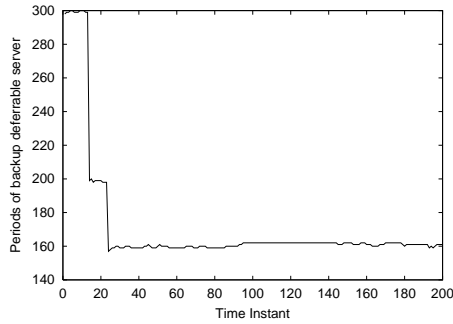


Fig. 4. Periods of backup deferrable server

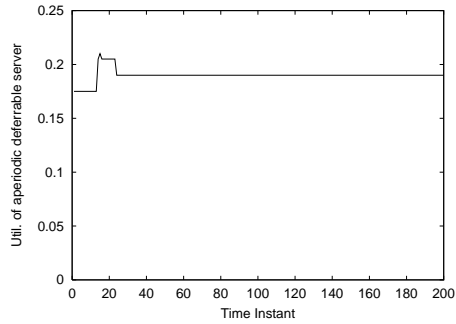


Fig. 5. Utilization of aperiodic deferrable server

4.1 Part 1: Fault Injection At Time 0

In the first part of the simulation, we generate three periodic tasks and inject fault at time instant 0. Fault rate for task T_1 , T_2 and T_3 are 0.01, 0.02 and 0.01 respectively.

Figure 4 shows that the period for the backup server decreases due to the fault injection. Then the period curve becomes flat after the fluctuation at the beginning.

Figure 5 shows the utilization allocated for the aperiodic deferrable server. At the beginning, the utilization is lower, this is due to the preassigned budget. Then the utilization goes up since the utilization of the backup server has not gone up very much. Then, when the utilization of the backup server increases more, the utilization of the aperiodic deferrable server decreases.

Figure 6 shows the utilization allocated for the backup deferrable server. The curve increases at the beginning and then becomes stable due to the injection of fault.

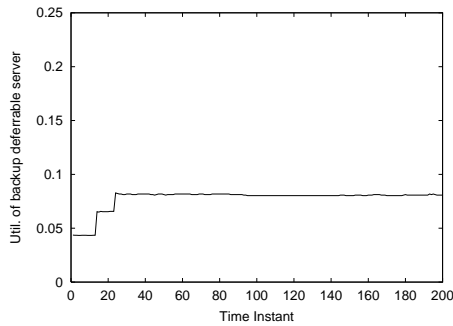


Fig. 6. Utilization of backup deferrable server

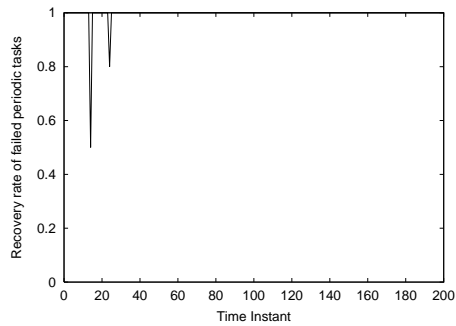


Fig. 7. Recovery rate of failed periodic tasks

Figure 7 shows the recovery rate of the failed periodic tasks. It reaches 100% soon. This means failed periodic tasks can be recovered after the curve reaches 100%.

4.2 Part 2: Steady State Performances

In the second part of the simulation, we measure the system performances for different average fault rate of tasks. In the steady state, the recovery rate is 100%. We plot the utilization of the aperiodic server instead of the schedulability of aperiodic tasks.

Figure 8 shows the periods of the backup deferrable server for different fault rates. The period of the backup server decreases when the fault rate increases.

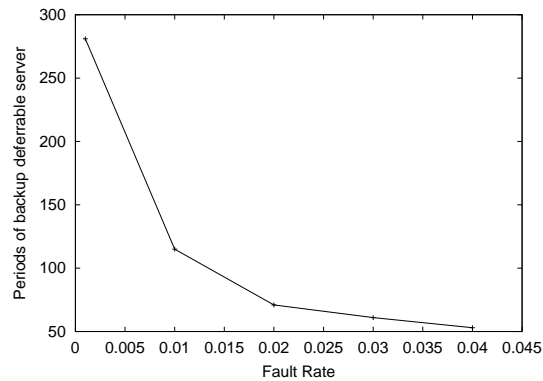


Fig. 8. Periods of backup deferrable server

Figure 9 shows the final values of the utilization allocated to the aperiodic deferrable server. The utilization decreases when the fault rate increase, since the aperiodic deferrable server needs to give some utilization to the backup server so that the recovery rate of the backup copies will reach 100%.

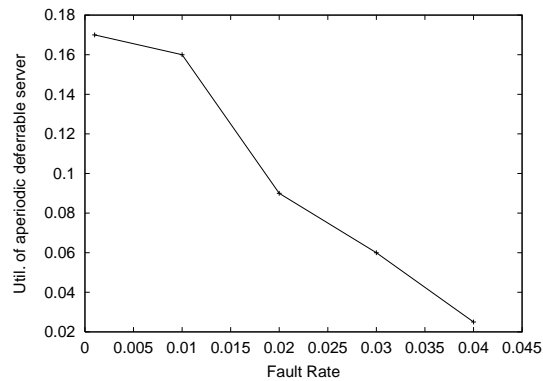


Fig. 9. Utilization of aperiodic deferrable server

Figure 10 shows the final values of the utilization allocated to the backup deferrable server. The utilization increases when the fault rate increase since more utilization needs to be allocated to the backup server when the fault rate increases.

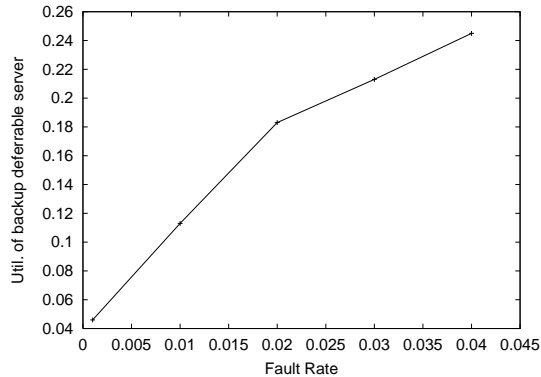


Fig. 10. Utilization of backup deferrable server

5 Related Work

In [10][11], authors proposed a new methodology for automatically adapting the rate of a periodic task set by using feedback control technique. The rate adaptation is good to achieve correct behavior of the system and high resource utilization. In [12][13], the authors present a feedback control EDF scheduling algorithm for real-time uniprocessor systems. The adaption is applied on tasks' service level. In [14][15][16], authors present a closed-loop scheduling algorithm based on execution time estimation. [17] assumes that each task consists of n sequential replicable sub-tasks. The controller adjusts the number of the replicas of sub-tasks to achieve low MR and high resource utilizations when the system is overloaded.

The above papers adopt the feedback control technique, but they do not address the issue of fault tolerance, which is important for periodic tasks in real-time systems. Our paper is the first one which addresses the combined scheduling with fault-tolerant issue.

There are several papers [6][7][8] which use PB approach to address fault tolerant scheduling problems in real-time systems, however, these papers do not use feedback technique, thus do not have the flexibility to allocate suitable utilization for tasks.

6 Conclusions

In this paper, we proposed a feedback-based fault-tolerant scheduling algorithm for real-time uniprocessor systems. The system has both periodic tasks and aperiodic tasks. Each periodic task can have a primary copy and a backup copy. The rate monotonic scheduling algorithm and deferrable server algorithm are used to schedule tasks. Two deferrable servers are created, one for aperiodic tasks and one for the backup copies of periodic tasks. The recovery rate and failure rate of periodic tasks are fed back to the controller, and the utilization capacity of the backup deferrable server is adjusted using feedback control theory. Suitable utilization capacity is allocated to backup deferrable server and the remaining utilization capacity is used for the aperiodic deferrable server. The simulation studies show that the algorithm can guarantee 100% recovery rate for periodic tasks with a balanced utilization for backup tasks.

References

1. K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems", in Proc. IEEE, vol.82, no.1, pp.55-67, Jan. 1994.
2. D. K. Pradhan, "Fault Tolerant Computing: Theory and Techniques", Prentice Hall, NJ, 1986.
3. J. K. Strosnider, J. P. Lehoczky and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments", *IEEE Trans. on Computers*, vol.44, no.1, pp.73-91, Jan. 1995.
4. Katsuhiko Ogata, "Modern Control Engineering", Prentice Hall, Upper Saddle River, New Jersey, 2002.
5. C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", *Journal of ACM*, vol.20, no.1, pp.45-61, Jan. 1973.
6. A. L. Liestman and R. H. Campbell, "A fault-tolerant scheduling problem", *IEEE Trans. Software Engineering*, vol.12, no.11, pp.1089-1095, Nov. 1988.
7. S. Ghosh, R. Melhem, and D. Mosse, "Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems", *IEEE Trans. on Parallel and Distributed Systems*, vol.8, no.3, pp.272-284, Mar. 1997.
8. G. Manimaran and C. Siva Ram Murthy, "A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis", *IEEE Tran. on Parallel and Distributed Systems*, vol.9, no.11, pp.1137-1152, Nov. 1998.
9. C. Siva Ram Murthy and G. Manimaran, "Resource Management in Real-Time Systems and Networks", MIT Press, April 2001.
10. G. Buttzaao, G. Lipari and L. Abeni, "Elastic task model for adaptive rate control", in Proc. *IEEE Real-Time Systems Symposium*, pp.286-295, 1998.
11. G. Buttazzo and L. Abeni, "Adaptive workload management through elastic scheduling", *Real-Time Systems*, vol.23, no.1-2, pp.7-24, July-September, 2002.
12. C. Lu, J. A. Stankovic, G. Tao, and S.H. Son, "Design and evaluation of feedback control EDF scheduling algorithm", In Proc. *IEEE Real-Time System Symposium*, pp.56-67, 1999.
13. J. A. Stankovic, Chenyang Lu, S. H. Son, and G. Tao, "The case for feedback control real-time scheduling", in Proc. *Euromicro Conference on Real-Time Systems*, pp.11-20, 1999.
14. D. R. Sahoo, S. Swaminathan, R. Al-Omari, M. V. Salapaka, G. Manimaran, and A. K. Somani, "Feedback control for real-time scheduling", in Proc. *American Controls Conference*, vol.2, pp.1254-1259, 2002.
15. R. Al-Omari, G. Manimaran, M. V. Salapaka, and A. K. Somani, "Novel algorithms for open-loop and closed-loop scheduling of real-time tasks based on execution time estimation", in Proc. *IEEE Intl. Parallel and Distributed Processing Symposium*, pp.7-14, 2003.
16. S. Lin, S. Sai Sudhir and G. Manimaran, "ConFiRM-DRTS: A certification framework for dynamic resource management in distributed real-time systems," in Proc. *Intl. Workshop on Parallel and Distributed Real-Time Systems*, pp.110-117, 2003.
17. B. Ravindran, P. Kachroo, and T. Hegazy, "Adaptive resource management in asynchronous real-time distributed systems using feedback control functions", in Proc. *Intl. Symposium on Autonomous Decentralized Systems*, pp.39-46, 2001.