Fault-Tolerant Rate-Monotonic Scheduling Algorithm in Uniprocessor Embedded Systems

Hakem Beitollahi
Hakem.Beitollahi@esat.kuleuven.be

Geert Deconinck
Geert.Deconinck@esat.kuleuven.be

Katholieke Universiteit Leuven, Electrical Engineering, Kasteelpark Arenberg 10, Leuven, Belgium

Abstract

The general approach to fault tolerance in uniprocessor systems is to use time redundancy in the schedule so that any task instance can be re-executed in presence of faults during the execution. In this paper a scheme is presented to add enough and efficient time redundancy to the Rate-Monotonic (RM) scheduling policy for periodic real-time tasks. This scheme can be used to tolerate transient faults during the execution of tasks. For performance evaluation of this idea a tool is developed.

1. Introduction

Transient faults in real-time systems are generally tolerated using time redundancy, which involves the retry or re-execution of any task running during the occurrence of transient faults [2]. Several studies have done for using time redundancy in embedded real-time systems for tolerating faults. Pandya and Malek in [3] have used time redundancy for tolerating a single fault. In the event of faults, all unfinished tasks are re-executed. In [4], authors have presented static and dynamic allocation strategies to provide fault-tolerance. Two algorithms have proposed to reserve time for the recovery of periodic real-time tasks on a uniprocessor [4]. In [1] authors have provided exact schedulability tests for fault-tolerant task sets. In their paper, time redundancy has been employed to provide a predictable performance in the presence of failures. However no study has been done about adding appropriate and efficient time redundancy into the schedule, which is the main contribution of this paper.

In recent years, Rate-Monotonic (RM) scheduling policy has been used to schedule real-time tasks in a variety of critical applications. However, RM does not provide mechanisms for managing time redundancy, so that real-time tasks will complete within their deadlines even in the presence of faults. The goal of this paper is to add appropriate and efficient time redundancy to the RM scheduling policy for schedule periodic tasks.

2. Determining Efficient Value of Time Redundancy for Rate-Monotonic Policy

The general approach to fault-tolerance in uniprocessor systems is to make sure there is enough slack in the schedule to allow for re-executing of any task instance, if a fault occurs during its execution [2]. In the RM policy with utilization less than 100%, there is a natural amount of slack in uniprocessor. But this natural slack is not enough for re-executing faulty tasks. To have an efficient fault-tolerant mechanism in the schedule it is necessary that additional slack time is added to the schedule. The recovery mechanism ensures that the reserved slack can be used for task re-executing before its deadline, without causing other tasks to miss deadlines.

The added slack is distributed throughout the schedule such that the amount of slack available over an interval of time (L) is proportional to the length of the interval (enough to enable the re-execution of any task). The ratio of slack S available over an interval of time L is thus constant and can be imagined to be the utilization of a backup task B, where S/L is the backup utilization. Formally, if the backup utilization is U_B , and the backup time (slack time) available during an interval L is denoted by B_L, then $B_L = U_B L$

Increasing the value of time redundancy has a considerable impact on the efficiency of scheduling and recovering tasks.

In order to obtain an efficient value of time redundancy, we need only to obtain an appropriate value of U_B. So the main goal of this section is to determine efficient value of U_B for any pair of average task utilization (α) and mean time to fault (MTTF), i.e., any pair of (α, MTTF).

To solve the problem, we define the gain variable that depends on Schedulability_value and Lost_tasks_value, and can be calculated using the following equation:
$Gain = \delta \times Schedulability\ \_value - \gamma \times Lost\ \_tasks\ \_value$

The ultimate goal of the system is to maximize Schedulabilty_value and to minimize Lost_tasks_value.
In other words, the gain variable should to be maximized. If chart of the gain variable is plotted for different values of U_B, MTTF, and α, we get the desired point where the gain is maximized. Based on this point, the appropriate value of U_B is determined.

3. Simulation Results

A set of periodic tasks is generated for every run of the simulation. On each task set, T_i is generated following a uniform distribution with $10 \le T_i \le 100$ and c_i is generated following a uniform distribution in the range $0 < c_i \le 2\alpha T_i$, where α is average task utilization.

For each result, the simulation was run 1000 times and the individual values were averaged. Each simulation was run for 10,000 time units.

Figure 1 shows the Schedulability_value of the task sets using different values of α. In Figure 2 and 3 the percentage of task instances lost due to faults for different values of MTTF are shown. Figures 4 and 5 show the gain variable of the system for two different values of α, α = 0.1 and α = 0.2.

Table 1 shows the appropriate value of U_B for four pairs of α and MTTF.

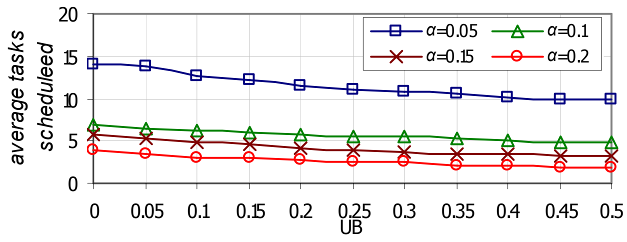

Figure 1. Average number of tasks scheduled for different values of α as function of U_B

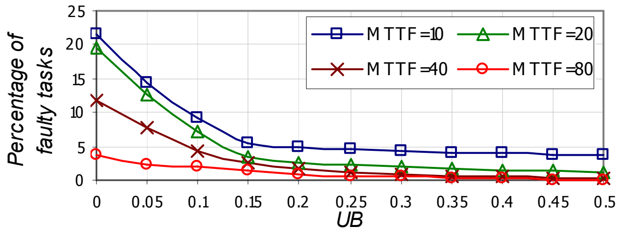

Figure 2. Percentage of faulty tasks for four different values of MTTF (α = 0.2)

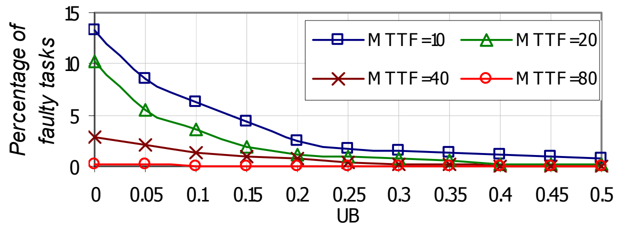

Figure 3. Percentage of faulty tasks for four different values of MTTF (α = 0.1)

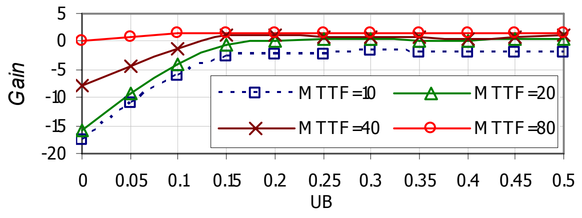

Figure 4. Gain variable for α = 0.2

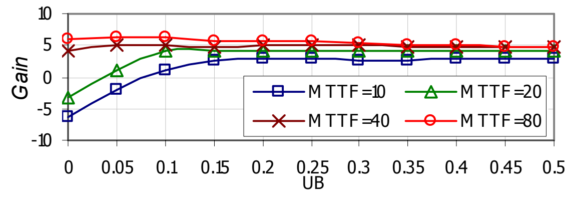

Figure 5. Gain variable for α = 0.1

Table 1. Appropriate value of U_B for pairs of (α, MTTF)

	MTTF=10	MTTF=20	MTTF=40	MTTF=80
$\alpha = 0.2$	0.3	0.25	0.15	0.1
$\alpha = 0.15$	0.25	0.2	0.15	0.1
$\alpha = 0.1$	0.2	0.1	0.05	0.05
$\alpha = 0.05$	0.1	0.05	0.05	0.05

4. Conclusions

In this paper, an approach has been presented for providing fault tolerance in the Rate-Monotonic (RM) scheduling policy. Here, a scheme has been presented to add appropriate and efficient time redundancy to the RM scheduling policy for periodic real-time tasks. The effect of time redundancy has been analyzed on both schedulability and recovery. Finally for sixteen pairs of average task utilization and mean time to fault an appropriate and efficient value of time redundancy has been determined.

References

[1] A. Burns, R. Davis, S. Punnekkat, "Feasibility Analysis of Fault-Tolerant Real-Time Task Sets", 8th Euromicro Workshop on Real-Time Systems, Jun 1996.

[2] D. Mossé, R. G. Melhem, S. Ghosh, "A Nonpreemptive Real-Time Scheduler with Recovery from Transient Faults and Its Implementation", IEEE Trans. Software Eng., 29(8):752-767, 2003.

[3] M.Pandya, M. Malek, "Minimum Achievable Utilization for Fault-tolerant Processing of Periodic Tasks" , IEEE Trans. On Reliability, 42(3):427-435, Sept 1993.

[4] S. Ramos-Thuel, J.K. Strosnider, "Scheduling Fault Recovery Operations for Time-Critical Applications", 4th IFIP Conference on Dependable Computing for Critical Applications, Jan 1995.