An Adaptive Scheme for Fault-Tolerant Scheduling of Soft Real-Time Tasks in Multiprocessor Systems[⋆]

R. Al-Omari, Arun K. Somani, and G. Manimaran

Dependable Computing & Networking Laboratory
Dept. of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
{*romari,arun,gmani*}*@iastate.edu*

Abstract. The scheduling of real-time tasks with primary-backup based fault-tolerant requirements has been an important problem for several years. Most of the known scheduling schemes are non-adaptive in nature meaning that they do not adapt to the dynamics of faults and task's parameters in the system. In this paper, we propose an adaptive fault-tolerant scheduling scheme that has a mechanism to control the overlap interval between the primary and backup versions of tasks such that the overall performance of the system is improved. The overlap interval is determined based on the estimated primary fault probability and task's soft laxity. We also propose a new performance index that integrates schedulability (S) and reliability (R) into a single metric, called SR index. To evaluate the proposed scheme, we have conducted analytical and simulation studies under different fault and deadline scenarios, and found that the proposed adaptive scheme adapts to system dynamics and offers better SR index than that of the non-adaptive schemes.

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]. Multiprocessors and multicomputer systems are emerging as a powerful computing means for real-time applications due to their capability for high performance and reliability. In most real-time applications, there is a need for reliable execution of tasks even in the presence of faults. Reliable (fault-tolerant) execution of a task is usually achieved by scheduling multiple versions of the task [2]-[9].

Primary-Backup (PB) scheduling is an important fault-tolerant approach in which two versions of a task are scheduled on two different processors and an *acceptance test* is used to check the correctness of the result [3]-[8]. The three variants [8] of this approach are: primary-backup exclusive (PB-EXCL) [5, 7],

[⋆] This research was supported in part by the NSF under grant CCR-0098354.

primary-backup concurrent (PB-CONCUR) [8,9], and primary-backup overlap (PB-OVER) [8].

In PB-EXCL approach, the primary and backup versions of a task are excluded in time as well as in space (i.e., scheduled on two different processors in an non-overlapping manner). The backup version is executed only if the output of the primary fails the acceptance test, otherwise it (backup) is deallocated from the schedule. This approach uses less resources if faults rarely occur, because backups are mostly deallocated. Nevertheless, it requires the *execution interval* (defined as the interval between start time of primary and finish time of backup) of a task to be at least twice that of its computation time. In soft real-time systems, the execution interval of a task is critical in determining the utility of the task's output. The lower the execution interval, the higher the utility.

In PB-CONCUR approach, the primary and the backup versions of a task are executed concurrently. In this approach, the *consumption time* (defined as the total processor time used by a task for its execution) of a task is always twice that of its computation time irrespective of the fault rate in the system. Nevertheless, the execution interval for the task is always equal to its computation time. The consumption time is inversely proportional to the processor utilization. The lower the consumption time, the better the schedulability.

It is evident that the PB approaches offer a trade-off between the number of tasks scheduled to the total utility of their output to the system. It can be noted that the reduction in both consumption time and execution interval will contribute to better performance. In summary, PB-EXCL offers better performance when fault rate is low or tasks have high soft laxities (Laxity of a task is equal to its deadline minus current time minus its remaining execution time; Informally, the higher the laxity, the lesser the urgency), whereas PB-CONCUR offers better performance when the fault rate is high and tasks have low laxities. The PB-OVER is a flexible approach wherein the primary and the backup versions of a task are scheduled to overlap in execution. This approach has the potential to exploit the advantages of the other two approaches if the overlap interval is suitably adapted.

In this paper, we propose an adaptive PB-OVER based scheduling scheme that makes use of an estimate of the primary fault probability and task's laxity to control the degree of overlap between its (task) versions. There exists some adaptive fault-tolerant scheduling algorithms (e.g., [6]) wherein the task specifies to the scheduler the adaptiveness strategy. Clearly, such an algorithm does not enjoy the benefits of feedback-based adaptiveness wherein the scheduler adapts to the dynamics of the system.

2 System Model

Task Model

1. The system has soft real-time aperiodic tasks. Each task T_i has the attributes arrival time (a_i), ready time (r_i), worst case computation time (c_i), a relative soft deadline (d_i^s), and a relative firm deadline (d_i^f). Tasks are non-preemptable. **2.** Each task T_i has two versions, namely primary version (Pr_i)

and backup version (Bk_i). The versions have identical attributes. **3.** $d_i^s = k_1 \times c_i$ and $d_i^f = k_2 \times c_i$, where $k_2 \geq k_1$ and $k_2 \geq 2$.

Fault Model

Assumption 1: We assume that each processor, except the scheduler, may fail due to hardware fault which results in task's failure. The faults can be transient or permanent and are independent of each other.

Assumption 2: We assume that $(d_i^f - r_i) \; \forall \; T_i$ are much smaller than the mean time to failure ($MTTF$) of the system. This assumption is used to enhance the probability of successfully executing the backup of a task, if its primary fails.

Assumption 3: We assume that there exists fault-detection mechanisms such as fail-signal and acceptance test to detect processor and task failures, respectively. The scheduler will not schedule tasks to a known faulty processor.

3 Proposed Adaptive Scheduling Scheme

In a dynamic scheduling, tasks arrive at the scheduler, from where they are distributed to other processors for execution. The communication between the scheduler and the processors is through dispatch queues (Figure 1). The scheduler runs in parallel with the processors and periodically updates the schedules in the dispatch queues. In our work, we employ a fault monitor to periodically observe the fault rate in the system using a feedback as shown in Figure 1. The fault rate is then used to estimate the primary (task) fault probability in the system. The fault probability together with tasks' soft laxity is used to control the overlap interval between the primary and backup versions of each task.

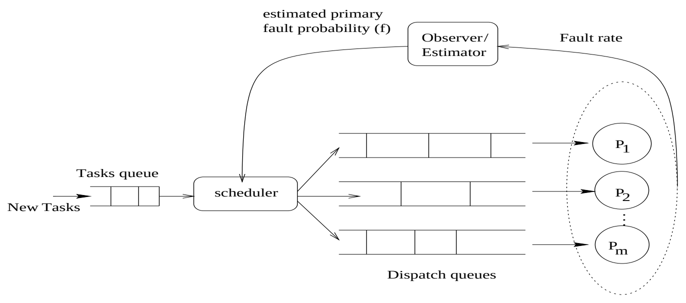


Fig. 1. An adaptive scheduler model

In this paper, we propose an adaptive PB-OVER scheduling scheme that makes use of an estimate of the primary fault probability and task's laxity to control the degree of overlap between the versions of a task. Two variations of the adaptive scheme are proposed by varying the adaptation mechanism. The adaptation can be done in a continuous manner which leads to an approach called PB-OVER continuous (PB-OVER-CONT), or it can be done in a discrete manner which leads to an approach called PB-OVER switch (PB-OVER-SWITCH). In PB-OVER-CONT, the overlap interval varies from no overlap to full overlap in a continuous manner as the fault probability varies from 0 to 1. In PB-OVER-SWITCH, the scheduler uses a threshold value of fault probability to switch from PB-EXCL to PB-CONCUR, i.e., if the probability is less than the threshold, the adaptive scheduler behaves like PB-EXCL, otherwise it behaves like PB-CONCUR. In PB-OVER-SWITCH, the threshold value is adapted with task's laxity.

3.1 Fault Monitoring System

A fault monitoring system periodically (with period p) monitors the completion of tasks in the system. For each period, the monitoring system counts the number of faulty primaries ($n^f(t)$) and the total number of primaries completed ($n(t)$) during that period.

According to the *frequency interpretation* probability concepts, the probability of an event (primary fail) is the proportion of the time that events of the same kind will occur in the long run. Hence, we define the primary fault probability (f) as the ratio of the number of faulty primaries to the total number of primaries executed in a given interval $[0,t]$: $f(t) = \frac{n^f(t)}{n(t)}$. This probability is calculated every p time units, where p is the sample period for the monitor.

3.2 Performance Index

The performance index that we use to compare the fault-tolerant approaches capture the trade-off between the processor utilization and task's value. This performance index is called the *schedulability-reliability* (SR) index.

For a task T_i, we define the processor utilization of the task ($UTIL_i$) as the ratio of worst case execution time (c_i) to the expected consumption time ($\overline{PT_i}$). $UTIL_i = c_i/\overline{PT_i}$. We also define a value function (VAL_i) that is used to credit the successful output from a task depending on its expected execution interval ($\overline{ET_i}$). In the value function (Equation 1), the task contributes a value of one if it finishes before its soft deadline, a monotonically decreasing value if it finishes between its soft and firm deadlines, a value of zero if it misses its firm deadline. The monotonically decreasing task value is inversely proportional to its expected execution interval.

$$VAL_i = \begin{cases} 1 & \text{for } \overline{ET_i} \leq d_i^s \\ \frac{d_i^s}{\overline{ET_i}} & \text{for } d_i^s \leq \overline{ET_i} \leq d_i^f \\ 0 & \text{for } \overline{ET_i} > d_i^f \end{cases} \tag{1}$$

The performance index of a soft real-time system is usually measured as the sum of the values contributed by the admitted tasks. For a given system capacity, there are two options: (i) admit a few tasks with a higher value for each task or (ii) admit a large number of tasks with a lower value for each task. Therefore, the SR_i index for a task (T_i) is:

$$SR_i = UTIL_i \times VAL_i. \tag{2}$$

The effect on the number of admitted tasks is inherently captured by the processor utilization of each task. The SR index for a set of n tasks that are admitted into the system is computed as $SR = \frac{\sum_{i=1}^{n} SR_i}{n}$.

4 Analysis of PB-based Fault-tolerant Approaches

To analyze and compare the fault-tolerant approaches, we make the following assumptions: (i) The primary fault probability (f) is estimated using the fault monitoring mechanism. (ii) The worst case execution time for the tasks is fixed and is equal to c. (iii) The relative soft deadline (d^s) for a task is c, and its relative firm deadline (d^f) is $2c$.

Primary-Backup EXCL*usive (*PB-EXCL*):* In this approach, the primary and the backup versions of a task are excluded in space as well as in time. The expected execution interval for a task T_i using this approach is: $\overline{ET_i} = 2cf + (1-f)c = c(1+f)$. The expected consumption time for the task is given by $\overline{PT_i} = 2cf + (1-f)c = c(1+f)$. The processor utilization is given by: $UTIL_i = \frac{c}{(1+f)c} = \frac{1}{(1+f)}$. Since $d_i^s = c$ and $d_i^f = 2c$, the value function for this approach is $VAL_i = \frac{c}{(1+f)c} = \frac{1}{1+f}$. The SR index for this approach is then $SR_i = \frac{1}{(1+f)^2}$ (refer Figure 2a for the plot).

Primary-Backup CONCUR*rent (*PB-CONCUR*):* In this approach, the primary and the backup versions of tasks are executed concurrently. In this approach, $\overline{ET_i}$ and $\overline{PT_i}$ are equal to c and $2c$, respectively. The processor utilization for this approach is given by $UTIL_i = \frac{c}{2c} = 0.5$. The value function for this approach is $VAL_i = \frac{c}{c} = 1$. The SR index for this approach is then $SR_i = 0.5$ (refer Figure 2a).

Primary-Backup OVER*lap (*PB-OVER*):* In this approach, the primary and the backup versions of a task can overlap in execution by an amount equal to γc. The $\overline{ET_i}$ and $\overline{PT_i}$ for the task using this approach are given by

$$\begin{aligned} \overline{ET_i} &= f(c + (1-\gamma)c) + (1-f)c = cf(1-\gamma) + c \\ \overline{PT_i} &= 2cf + (1-f)(c+\gamma c) = (1-\gamma)cf + (1+\gamma)c \end{aligned} \tag{3}$$

The processor utilization and the value function for this approach are respectively given by

$$UTIL_i = \frac{1}{1+f+\gamma(1-f)} \quad \textit{and} \quad VAL_i = \frac{1}{1+f(1-\gamma)} \tag{4}$$

The SR index for this approach is then

$$SR_i = \frac{1}{(1+f+\gamma(1-f))(1+f(1-\gamma))} \tag{5}$$

The most important property that is expected from PB-OVER is to combine the advantages of PB-EXCL and PB-CONCUR approaches. That is, when $f = 0$, the desirable values of $\overline{ET_i}$ and $\overline{PT_i}$ is c. When $f = 1$, the desirable values of $\overline{ET_i}$ and $\overline{PT_i}$ is c and $2c$, respectively.

5 Adaptive PB-OVER Fault-tolerant Approaches

To achieve the adaptive property, we propose two adaptive PB-OVER approaches by varying the adaptation mechanism: PB-OVER continuous (PB-OVER-CONT) and PB-OVER switch (PB-OVER-SWITCH).

5.1 Primary-Backup OVERlap CONTinuous (PB-OVER-CONT)

In PB-OVER-CONT, the overlap interval varies from no overlap to full overlap in a continuous manner as the fault probability varies from 0 to 1. From Equation (3), we found that this can be achieved by substituting $\gamma = f$ which results in

$$\overline{ET_i} = (-f^2 + f + 1)c \;\; and \;\; \overline{PT_i} = (-f^2 + 2f + 1)c \tag{6}$$

Using the assumption in Section 4 the processor utilization and the value function for this approach are respectively given by

$$UTIL_i = \frac{1}{-f^2 + 2f + 1} \;\; and \;\; VAL_i = \frac{1}{(-f^2 + f + 1)} \tag{7}$$

The SR_i index for this approach (Figure 2a) is then $SR_i = \frac{1}{(-f^2+2f+1)(-f^2+f+1)}$

5.2 Primary-Backup OVERlap SWITCH (PB-OVER-SWITCH)

In PB-OVER-SWITCH, the scheduler switches from PB-EXCL to PB-CONCUR depending on the value of f. If f is less than a threshold f_o, then the PB-OVER-SWITCH approach behaves like PB-EXCL, else it behaves like PB-CONCUR. The threshold f_o is the value of f at which the SR index of PB-EXCL is equal to that of PB-CONCUR. Thus, $\overline{ET_i}$ and $\overline{PT_i}$ of the task become

$$\overline{ET_i} = \begin{cases} c \times (1+f) & \text{for } 0 \le f \le f_o \\ c & \text{for } f_o < f \le 1 \end{cases} \;\; and \;\; \overline{PT_i} = \begin{cases} c \times (1+f) & \text{for } 0 \le f \le f_o \\ 2c & \text{for } f_o < f \le 1 \end{cases} \tag{8}$$

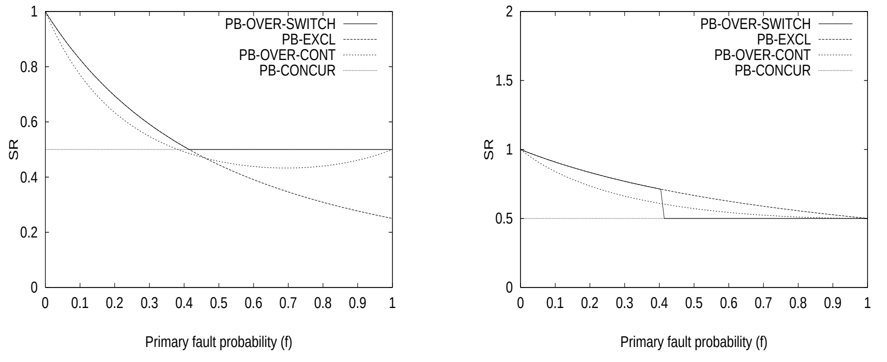
For the given assumption the value of f_o is the value of f that satisfies $\frac{1}{(1+f)^2} = 0.5$, which is $f = \sqrt{2} - 1$. The SR index for this approach (Figure 2a) is given by

$$SR_i = \begin{cases} \frac{1}{(1+f)^2} & \text{for } 0 \le f \le \sqrt{2} - 1 \\ 0.5 & \text{for } \sqrt{2} - 1 \le f \le 1 \end{cases} \tag{9}$$

5.3 Effect of Task's Soft Laxity on Performance

In this section, we first study the effect of task's soft laxity on the performance of the PB-based fault-tolerant approaches. Next, we propose a mechanism that allow the fault probability threshold (f_o) to be adapted with task's soft laxity.

In Sections 4, 5.1, and 5.2 (Figure 2a), we assumed that the task's soft deadline (d^s) is equal to c. This assumption allows the PB-EXCL approach to schedule only one version of a task (Pr) within its soft deadline and the other version (Bk) must be scheduled after the soft deadline. This degrades the performance of the PB-EXCL approach as f increases because the correct output is always produced by the backup which has less value. When tasks have a large soft deadline, then both the primary and the backup versions of the task can be scheduled in an exclusive manner within their soft deadline. Therefore, all PB-based fault-tolerant approaches offer the same output value (VAL) for the finished tasks which is equal to one and does not depend on f. Hence, the SR index will be 0.5, $\frac{1}{1+f}$, and $\frac{1}{-f^2+2f+1}$, respectively for the PB-CONCUR, PB-EXCL, and the PB-OVER-CONT approaches (Figure 2b).



(a) Small soft laxity (b) Large soft laxity

Fig. 2. SR index for the PB approaches

From Figure 2b, we notice that the PB-EXCL offers the best SR index for all values of f. The PB-SWITCH offers the best SR index only when $f < \sqrt{2} - 1$ and it offers the lowest SR index when $f > \sqrt{2}-1$. This is because the threshold value (f_o) that is used by the PB-SWITCH approach to switch between the PB-EXCL and PB-CONCUR approaches is constant ($f_0 = \sqrt{2} - 1$) and does not change with changing task's soft deadline. In Section 5.2, the threshold f_0 is defined as the value of f at which the SR index of PB-EXCL is equal to that of PB-CONCUR. Therefore, for the given task's deadline the value of f_0 is the value of f that satisfies $\frac{1}{(1+f)} = 0.5$, which is $f = 1$. By using this new threshold value ($f_0 = 1$), the PB-OVER-SWITCH always behaves like PB-EXCL approach which offers the best performance in this case.

It is evident from the above discussion that the threshold value (f_0) have to be adapted with the task's soft laxity to be able to determine the correct overlap interval that maximizes the performance of the system. To do so, the scheduler behaves as follows for each task (T_i) that arrives in the system:

1. Schedules the primary (Pr_i) version of the task.
2. If the primary was scheduled within the soft deadline, then

(a) Determine the overlap interval $(\sigma_i c_i)$ between the primary and the backup versions of the task so that both versions will be scheduled within the soft deadline.

$$\sigma_i = \frac{c_i - (d_i^s - ft(Pr_i))}{c_i} \tag{10}$$

where $ft(Pr_i)$ is the relative finish time of the primary version.

(b) If $\sigma_i < 0$ then, $\sigma_i = 0$.

3. Else if the primary was scheduled to finish after the soft deadline, then $\sigma_i = 1$.

Using this value (σ_i) the scheduler calculates the threshold value (f_0^i) that is used by the PB-OVER-SWITCH approach to select the correct overlap interval (i.e., if $(f < f_0^i) : \gamma = 0 \ ? \ \gamma = 1$) between the primary and the backup versions for this task (T_i). To calculate f_0^i for a given task T_i the scheduler uses the following equation:

$$f_0^i = (\sqrt{2} - 2)\sigma_i + 1 \tag{11}$$

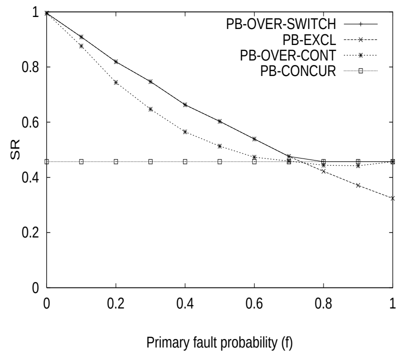
The above equation is derived from the fact that f_0 change from $\sqrt{2} - 1$ to 1, when the task's soft laxity changes from the case where the scheduler is only able to schedule one version of the task within its soft deadline to the case where it is able to schedule both versions in an exclusive manner within the soft deadline.

5.4 Performance Studies

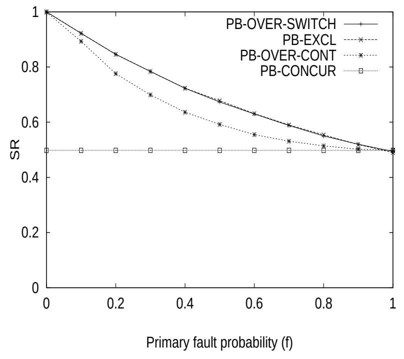
In this section, we compare the performance of the proposed adaptive fault-tolerant approaches with that of the non-adaptive PB-based fault-tolerant approaches, through simulation studies. The SR index has been used as the performance metric. For each point in the performance plots the system was simulated with 20,000 tasks. This number of tasks has been chosen to have a 99% confidence interval within ± 0.0035 around each value of SR.

Figures 3(a) and 3(b) show the effect of primary fault probability on the SR index for all PB approaches. Figure 3(a) shows the behavior of the PB approaches when f varies for tasks that have relative soft deadline equals to $2c_i$ and relative firm deadline equals to $5c_i$. Figure 3(b) shows the case for tasks that have relative soft deadline equals to $4c_i$ and relative firm deadline equals to $5c_i$.

From the figures, it can be seen that when the relative soft deadline is large (Figure 3(b)) the PB-OVER-SWITCH and the PB-EXCL approaches behave similar and offer the best SR index. This is because when d^s is large, the tasks have enough soft laxity to be scheduled in an exclusive manner within the soft deadline. The threshold value f_0 used by the PB-OVER-SWITCH approach is approximately equal to 1. From Figure 3(a), we can notice that the PB-EXCL approach has the highest SR index when $f < 0.75$, and the PB-CONCUR approach has the highest SR index when $f > 0.75$. The SR index offered by the PB-OVER-CONT lies between the PB-CONCUR and PB-EXCL for all values of primary fault probability. Since the PB-OVER-SWITCH behaves like PB-EXCL

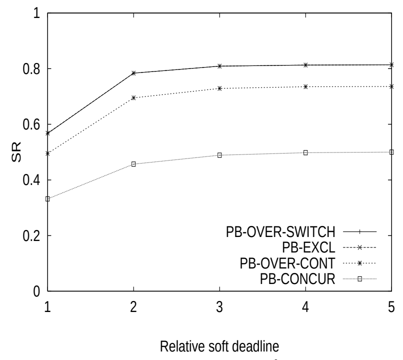


(a) $d^s = 2c_i$, and $d^f = 5c_i$

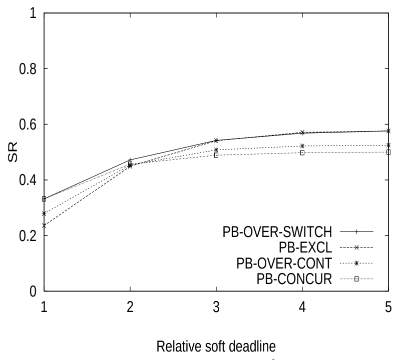


(b) $d^s = 4c_i$, and $d^f = 5c_i$

Fig. 3. Effect of primary fault probability (f) on SR index



(a) $f = 0.25$, and $d^f = 5c_i$



(b) $f = 0.75$, and $d^f = 5c_i$

Fig. 4. Effect of task's soft laxity on SR index

when $f \leq 0.75$ and like PB-CONCUR when $f > 0.75$, it offers the best SR index for all values of f.

Figures 4(a) and 4(b) show the effect of task's soft laxity on the SR index for all PB approaches. Figure 4(a) shows the behavior of the PB approaches when d^s varies for the system in which the primary fault probability is 0.25 and tasks' relative firm deadline is $5c_i$. Figure 4(b) shows the case for system in which the primary fault probability is 0.75 and tasks' relative firm deadline is $5c_i$.

From the figures, it can be seen that when the primary fault probability (f) is low (Figure 4(a)) the PB-OVER-SWITCH and the PB-EXCL approaches behave similar and offer the best SR index for all values of d^s. This is because when f is small, the PB-OVER-SWITCH switch to an overlap interval equal to zero. The SR index offered by the PB-OVER-CONT lies between the PB-CONCUR and PB-EXCL for all values of d^s. From Figure 4(b), we can notice that the PB-CONCUR and PB-OVER-SWITCH approaches have the highest SR index when $d^s \leq 2$. This is because, when f is high and tasks have small soft laxity, the threshold value that is used by the PB-OVER-SWITCH for each task is smaller than the fault probability in the system ($f = 0.75$). Thus, PB-OVER-SWITCH behaves like PB-CONCUR for this region. It can be noticed that when $2 \leq d^s \leq 3$, the PB-OVER-SWITCH approach has the highest SR index. This is because, in this interval, the tasks have medium values of soft deadline. Thus, PB-OVER-SWITCH maximizes the performance by adapting the task's fault threshold (f_o^i) based on soft deadline. Finally, in Figure 4(b), it can be noticed that the PB-EXCL and PB-OVER-SWITCH approaches have the highest SR index when $d^s > 3$.

6 Conclusions

In this paper, we have proposed an adaptive scheme for the PB-based fault-tolerant scheduling of tasks on multiprocessor real-time systems. The key idea used is to control the overlap interval between the primary and backup versions of a task based on an estimated value of primary fault probability and task's soft laxity. Two variants (PB-OVER-CONT and PB-OVER-SWITCH) of the adaptive scheme have been proposed and studied. We have also proposed a new metric, called schedulability-reliability (SR) index. Our studies show that the proposed PB-OVER-SWITCH adaptive scheme always performs better than the other PB-based approaches for the SR index performance metric. The feedback based adaptation mechanism, such as the one developed in this paper, opens up many avenues for further research in value-based scheduling in real-time systems.

References

1. K.G. Shin and P. Ramanathan, "Real-time computing: A new discipline of computer science and engineering," *Proc. of IEEE,* vol. 82, no. 1, pp. 6-24, Jan. 1994.
2. L. Chen and A. Avizienis, "N-version programming: A fault tolerant approach to reliability of software operation", In Proc. *IEEE Fault-Tolerant Computing Symposium,* pp. 3-9., 1978.
3. B. Randell, "System structure for software fault-tolerance," *IEEE Transactions on Software Engineering,* vol. 1, no. 2, pp. 220-232, June 1975.
4. S. Tridandapani, A. K. Somani, and U. Reddy, "Low overhead multiprocessor allocation strategies exploiting system spare capacity for fault detection and location," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 865-877, July 1995.
5. S. Ghosh, R. Melhem, and D. Mosse, "Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 8, no. 3, pp. 272-284, Mar. 1997.
6. O. González, H. Shrikumar, J.A. Stankovic, and K. Ramamritham, "Adaptive fault-tolerance and graceful degradation under dynamic hard real-time scheduling," In Proc. *IEEE Real-Time Systems Symposium,* 1997.
7. G. Manimaran and C. Siva Ram Murthy, " A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis," *IEEE Transactions on Parallel and Distributed Systems,* vol. 9, no. 11, pp. 1137-1152, Nov. 1998.
8. I. Gupta, G. Manimaran, and C. Siva Ram Murthy, "Primary-Backup based fault-tolerant dynamic scheduling of object-based tasks in multiprocessor real-time systems," in *Dependable Network Computing,* D.R. Avresky (editor), Kluwer Academic Publishers, 1999.
9. F. Wang, K. Ramamritham and J. A. Stankovic, "Determining redundancy levels for fault tolerant real-time systems," *IEEE Transactions on Computers,* vol. 44, no. 2, pp. 292-301, Feb. 1995.