A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods

Kang G. Shin, Fellow, IEEE, and Hagbae Kim, Member, IEEE

Abstract—Failure to establish a majority among the processing modules in a triple modular redundant (TMR) system, called a TMR failure, is detected by using two voters and a disagreement detector. Assuming that no more than one module becomes permanently faulty during the execution of a task, Re-execution of the task on the same Hardware (RSHW) upon detection of a TMR failure becomes a cost-effective recovery method, because 1) the TMR system can mask the effects of one faulty module while RSHW can recover from nonpermanent faults, and 2) system reconfiguration—Replace the faulty Hardware, reload, and Restart (RHWR)—is expensive both in time and hardware.

We propose an adaptive recovery method for TMR failures by "optimally" choosing either RSHW or RHWR based on the estimation of the costs involved. We apply the Bayes theorem to update the likelihoods of all possible states in the TMR system with each voting result. Upon detection of a TMR failure, the expected cost of RSHW is derived with these likelihoods and then compared with that of RHWR. RSHW will continue either until it recovers from the TMR failure or until the expected cost of RSHW becomes larger than that of RHWR. As the number of unsuccessful RSHW's increases, the probability of permanent fault(s) having caused the TMR failure will increase, which will, in turn, increase the cost of RSHW. Our simulation results show that the proposed method outperforms the conventional reconfiguration method using only RHWR under various conditions.

Index Terms—Spatial and time redundancy, TMR failure, permanent and nonpermanent faults, reconfiguration, nominal task-execution time, likelihoods, Bayes theorem.

I. INTRODUCTION

Fault tolerance is generally accomplished by using redundancy in hardware, software, time, or combination thereof. There are three basic types of redundancy in hardware and software: static, dynamic, and hybrid. Static redundancy masks faults by taking a majority of the results from replicated tasks [13]. Dynamic redundancy takes a two-step procedure for detection of, and recovery from, faults [2]. The effectiveness of this method relies on selecting a suitable number of spares, a fault-detection scheme, and a switching operation. Hybrid redundancy is a combination of static and dynamic redundancy [4]. A core based on static hardware redundancy, and several spares are provided to tolerate faults. Such redundant systems could provide very high reliability depending on the number of spares used under the assumption of perfect coverage and switching operation. However, new faults may occur during the detection of existing faults, and the switching operation becomes very complex as the number of spares increases. In order to reduce the complexity of switching operation and enhance reliability at low cost, self-purging [12] and shift-out [5] schemes were developed, where faulty modules were removed but not replaced by standby spares. In these schemes, the additional operation required to select nonfaulty spare(s) is not needed, thus making the switching operation simpler. But it is difficult to implement either a threshold voter or a shift-out checking unit which requires comparators, detectors, and collectors.

Triple Modular Redundancy (TMR) has been one of the most popular fault-tolerance schemes using spatial redundancy. In the Fault-Tolerant MultiProcessor (FTMP) [6], computations are done on triplicated processors/memories connected by redundant common serial buses, and its quad-redundant clocks use bit-by-bit voting in hardware on all transactions over these buses. C.vmp [18] is also a TMR system which traded performance for reliability by switching between TMR mode with voting and independent modes under program control. In [22], an optimal TMR structure to recover from a transient fault was shown to extend significantly the lifetime of a small system in spite of its requirement of reliable voter circuits. The authors of [3] proposed a modular TMR multiprocessor to increase reliability and availability by using a retry mechanism to recover transient faults, and switching between TMR and dual-processor modes to isolate a permanent fault. A simple multiple-retry policy (retry a pre-specified number of times)—also used to discriminate a permanent fault—was employed there. This policy can tolerate multiple faults only by treating them as a sequence of single faults with repair between fault occurrences, thus requiring frequent voting for effective fault detection. A TMR failure caused by near-coincident faults in different modules must also be detected and recovered. The effect of dependent faults inducing a TMR failure was eliminated by periodic resynchronization at an optimal time interval [7]. However, the fault model of [7] and [22] did not include the possibility of permanent faults for which resynchronization is no longer effective.

In addition to the use of spatial redundancy with fault masking or reconfiguration, time redundancy can be applied effectively to recover from transient faults. Such recovery techniques are classified into instruction retry [10], program

Manuscript received October 15, 1992; revised May 10, 1993. This work was supported in part by the Office of Naval Research under Contract No. N00014-91-J-1115 and by the NASA under Grant No. NAG-1-1120. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122 USA; e-mail: kgshin@eecs.umich.edu. IEEE Log Number 9403024.
rollback [16], program reload and restart with module replacement. Several researchers attempted to develop an optimal recovery policy using time redundancy, mainly for simplex systems. Koren [9] analyzed instruction retries and program rollbacks with such design parameters as the number of retries and intercheckpoint intervals. Berg and Koren [2] proposed an optimal module switching policy by maximizing application-oriented availability with a pre-specified retry period. Lin and Shin [10] derived the maximum allowable retry period by simultaneously classifying faults and minimizing the mean task-completion time.

The main intent of this paper is to develop an approach of combining time and spatial redundancy by applying time redundancy to TMR systems. (Note that spatial redundancy is already encapsulated in the TMR system.) When a TMR failure—failure to establish a majority due to multiple-module faults—is detected at the time of voting, or when a faulty module, even if its effects are masked, is identified, the TMR system is conventionally reconfigured to replace all three or just the faulty module with fault-free modules. If the TMR failure had been caused by transient faults, system reconfiguration or Replacement of HardWare and Restart (RHWR), upon detection of a TMR failure, may not be desirable due to its high cost in both time and hardware. To counter this problem, we propose to, upon detection of a TMR failure, Re-execute the corresponding task on the Same HardWare (RSHW) without module replacement. Instruction retry intrinsically assumes almost-perfect fault detection, for which TMR systems require frequent voting, thereby inducing high time overhead. However, the probability of system crash due to multiple-channel faults is shown in [17] to be insignificant for general TMR systems, even when the outputs of computing modules are infrequently voted on as long as the system is free of latent faults. Unlike simplex systems, program rollback is not adequate for TMR systems due to the associated difficulty of checkpointing and synchronization. So, we consider re-execution of tasks on a TMR system with infrequent voting. For example, since more than 90% of faults are known to be nonpermanent—as few as 2% of field failures are caused by permanent faults [14]—simple re-execution may be an effective means to recover from most TMR failures. This may reduce 1) the hardware cost resulting from the hasty elimination of modules with transient faults and 2) the recovery time that would otherwise increase, i.e., as a result of system reconfiguration. Note that system reconfiguration is time-consuming because it requires the location and replacement of faulty modules, program and data reloading, and resuming execution.

We shall propose two RSHW methods for determining when to reconfigure the system instead of re-executing a task without module replacement. The first (nonadaptive) method is to determine the maximum number of RSHW’s allowable (MNR) before reconfiguring the system for a given task according to its nominal execution time without estimating the system (fault) state—somewhat similar to the multiple-retry policy applied to a general rollback recovery scheme in [20]. By contrast, the second (adaptive) method 1) estimates the system state with the likelihoods of all possible states and 2) chooses the better of RSHW or RHWR based on their expected costs when the system is in one of the estimated states. RHWR is invoked if either the number of unsuccessful RSHW’s exceeds the MNR in the first method or the expected cost of RSHW gets larger than that of RHWR in the second method. For the second method, we shall develop an algorithm for choosing between RSHW and RHWR upon detection of a TMR failure. We shall also show how to calculate the likelihoods of all possible states, and how to update them using the RSHW results and the Bayes theorem.

The paper is organized as follows. In the following section, we present a generic methodology of handling TMR failures, and introduce the assumptions used. Section III derives the optimal voting interval \( (X, X) \) for a given nominal task-execution time \( X \). The MNR of the first method and the optimal recovery strategy of the second method are computed for given \( X \). We derive the probability density function (pdf) of time to the first occurrence of a TMR failure, the probabilities of all possible types of faults at that time, transition probabilities up to the voting time, the costs of RSHW and RHWR, and the problem of updating likelihoods of the system state and the recovery policy after an unsuccessful RSHW. Section IV presents numerical results and compares two recovery methods of RSHW and RHWR. The paper concludes with Section V.

II. DETECTION AND RECOVERY OF A TMR FAILURE

Detection and location of, and the subsequent recovery from, faults are crucial to the correct operation of a TMR system, because the TMR system fails if either a voter fails at the time of voting or faults manifest themselves in multiple modules during the execution of a task. The fault occurrence rate is usually small enough to ignore coincident faults which are not caused by a common cause, but noncoincident fault arrivals at different modules are not negligible and may lead to a TMR failure.

Disagreement detectors which compare the values from the different voters of a TMR system can detect single faults, but may themselves become faulty. FTMP [6], JPL-STAR [1], and C-vmf [18] are example systems that use disagreement detectors. In FTMP, any detected disagreement is stored in error latches which compress fault-state information into error words for later identification of the faulty module(s). System reconfiguration to resolve the ambiguity in locating the source of a detected error is repeated depending on the source of the error and the number of units connected to a faulty bus. Two fault detection strategies—hard failure analysis (HFA) and transient failure analysis (TFA)—are provided according to the number and persistency of probable faulty units. These strategies may remove the unit(s) with hard failures or update the fault index (demerit) of a suspected unit. Frequent voting is required to make this scheme effective, because any faulty module must be detected and recovered before the occurrence of a next fault on another module within the same TMR system.

Voting in a TMR system masks the output of one faulty module, but does not locate the faulty module. One can, however, use a simple scheme to detect faulty modules and/or
voter. Assuming that the probability of two faulty modules producing an identical erroneous output is negligibly small, the output of a module-level voter becomes immaterial when multiple modules are faulty [8]. A TMR failure can then be detected by using two identical voters and a self-checking comparator as shown in Fig. 1. These voters can be implemented with conventional combinational logic design [23]. The comparator can be easily made self-checking for its usually simple function: for example, a simple structure made of two-rail comparators in [11] for each bit can be utilized for its high reliability and functionality. This TMR structure can also detect a voter fault. When a TMR failure or a voter fault occurs, the comparator can detect the mismatch between the two voters that results from either the failure to form a majority among three processing modules, or a voter fault. (Note that using three voters, instead of two, would not make much difference in our discussion, so we will focus on a two-voter TMR structure.)

If the comparator indicates a mismatch between two voters at the time of voting, an appropriate recovery action must follow. Though RHWR has been widely used, RSHW may prove more cost-effective than RHWR in recovering from most TMR failures. To explore this in-depth, we will characterize RSHW with the way the MNR is determined. The simplest is to use a constant number of RSHW's irrespective of the nominal task-execution time and the system state which is measured in CPU cycles between the (i−1)th and ith voting, in the absence of any TMR/voter failure. As shown in Fig. 2, for 1 ≤ i ≤ n let wi represent the task-execution time from the beginning of the task to the first completion of the ith voting possibly in the presence of some module failures, and let \( W_i = E(w_i) \). Then \( E(w_i) = W_n \) is the expected execution time of the task. Upon detection of a TMR failure, let \( p \) and \( q \) be the probabilities of recovering a task with RSHW and RHWR, respectively, where \( p + q = 1 \). Assuming that the time overhead of reconfiguration is constant \( T_c \), \( W_n \) is expressed as a recursive equation in terms of \( W_i \), 1 ≤ i ≤ n.

Let \( F_i(t) \) (2 ≤ i ≤ n) be the probability of a TMR failure in \( t \) units of time from the system state at the time of the \( (j-1) \)th voting, and let \( F_i(t) \) be that from the beginning of the task. The probability of a recovery attempt (i.e., RSHW or RHWR) being successful depends upon \( F_i(t) \). When a TMR failure is detected at the time of first voting (i.e., it occurred during the execution of the task portion corresponding to \( X_1 \)), the system will try RSHW (or RHWR) with probability \( p \) (or \( q \)) to recover from the failure. This process is renewed probabilistically for the variable \( w_1 \) which is the actual task-execution time corresponding to the nominal task-execution time \( X_1 \). Thus,

\[
\begin{align*}
w_i &= \begin{cases} X_i & \text{with probability } 1 - F_i(X_i) \\ X_i + w_i & \text{with probability } F_i(X_i)p \\ X_i + T_c + w_i & \text{with probability } F_i(X_i)q \end{cases} \\
w_i &= \begin{cases} X_i & \text{with probability } 1 - F_i(X_i) \\ X_i + w_i & \text{with probability } F_i(X_i)p \\ X_i + T_c + w_i & \text{with probability } F_i(X_i)q \end{cases}
\end{align*}
\]

where \( T_c \) is the setup time for system reconfiguration.

Let \( T_c \) be the time overhead of voting that is in practice negligible. The above equation is also renewed for all \( w_i \)'s (2 ≤ i ≤ n) after each successful recovery. Hence,

\[
w_i = w_{i-1} + V_i + T_c, \quad \text{for } 2 \leq i \leq n,
\]

where \( V_i \) is defined as the actual task-execution time between the (i−1)th and ith voting, i.e., \( Y_i = w_i \) and

\[
V_i = \begin{cases} X_i & \text{with probability } 1 - F_i(X_i) \\ X_i + w_i & \text{with probability } F_i(X_i)p \\ X_i + T_c + w_i & \text{with probability } F_i(X_i)q \end{cases}
\]

From the above equations, the following recursive expressions are derived for 2 ≤ i ≤ n:

\[
W_i = \frac{1}{1 - F_i(X_i)}(W_{i-1} + T_c + X_i + F_i(X_i)qT_c).
\]

1This procedure is described in the algorithm of Fig. 4. In this case, the possibility of permanent faults having caused the TMR failure increases, which, in turn, increases the cost of RSHW significantly.

Throughout this paper, we assume that the arrival of permanent faults and the arrival and disappearance of nonpermanent faults are Poisson processes with rates \( \lambda_p, \lambda_n \), and \( \mu \), respectively.

III. OPTIMAL RECOVERY FROM A TMR FAILURE USING RSHW

The Optimal Voting Interval

Let \( X_i \) (2 ≤ i ≤ n) be the nominal task-execution time measured in CPU cycles between the (i−1)th and ith voting, and let \( X_1 \) be that between the beginning of the task and the first voting, in the absence of any TMR/voter failure. As shown in Fig. 2, for 1 ≤ i ≤ n let \( w_i \) represent the task-execution time from the beginning of the task to the first completion of the ith voting possibly in the presence of some module failures, and let \( W_i = E(w_i) \). Then \( E(w_i) = W_n \) is the expected execution time of the task. Upon detection of a TMR failure, let \( p \) and \( q \) be the probabilities of recovering a task with RSHW and RHWR, respectively, where \( p + q = 1 \). Assuming that the time overhead of reconfiguration is constant \( T_c \), \( W_n \) is expressed as a recursive equation in terms of \( W_i \), 1 ≤ i ≤ n.

Let \( F_i(t) \) (2 ≤ i ≤ n) be the probability of a TMR failure in \( t \) units of time from the system state at the time of the (j−1)th voting, and let \( F_i(t) \) be that from the beginning of the task. The probability of a recovery attempt (i.e., RSHW or RHWR) being successful depends upon \( F_i(t) \). When a TMR failure is detected at the time of first voting (i.e., it occurred during the execution of the task portion corresponding to \( X_1 \)), the system will try RSHW (or RHWR) with probability \( p \) (or \( q \)) to recover from the failure. This process is renewed probabilistically for the variable \( w_1 \) which is the actual task-execution time corresponding to the nominal task-execution time \( X_1 \). Thus,

\[
w_i = \begin{cases} X_i & \text{with probability } 1 - F_i(X_i) \\ X_i + w_i & \text{with probability } F_i(X_i)p \\ X_i + T_c + w_i & \text{with probability } F_i(X_i)q \end{cases}
\]

where \( T_c \) is the setup time for system reconfiguration.

Let \( T_c \) be the time overhead of voting that is in practice negligible. The above equation is also renewed for all \( w_i \)'s (2 ≤ i ≤ n) after each successful recovery. Hence,

\[
w_i = w_{i-1} + V_i + T_c, \quad \text{for } 2 \leq i \leq n,
\]

where \( V_i \) is defined as the actual task-execution time between the (i−1)th and ith voting, i.e., \( Y_i = w_i \) and

\[
V_i = \begin{cases} X_i & \text{with probability } 1 - F_i(X_i) \\ X_i + w_i & \text{with probability } F_i(X_i)p \\ X_i + T_c + w_i & \text{with probability } F_i(X_i)q \end{cases}
\]

From the above equations, the following recursive expressions are derived for 2 ≤ i ≤ n:

\[
W_i = \frac{1}{1 - F_i(X_i)}(W_{i-1} + T_c + X_i + F_i(X_i)qT_c).
\]

1This procedure is described in the algorithm of Fig. 4.
Applying this recursively \( n - 1 \) times, we can get:

\[
W_n = \sum_{j=1}^{n} \left( \frac{F_j(X_j)qT_e + X_j + T_v}{1 - F_j(X_j)} \right). \tag{3.1}
\]

The optimal voting frequency is derived by minimizing \( W_n \) with respect to \( n \) and \( X_i, 1 \leq i \leq n \), subject to

\[
\sum_{i=1}^{n} X_i = X.
\]

If all inter-voting intervals are assumed to be identical then the constant voting interval is

\[
X_i = \frac{X}{n},
\]

where \( p_s^0(p_u^0) \) and \( F_1(X) \) denote the probability of the \( n \)th RSHW becoming successful (unsuccessful) and the probability of a TMR failure during \( X \) after system reconfiguration, respectively, where \( p_s^0 + p_u^0 = 1, 1 \leq n \leq k \). In fact, \( p_s^0 \) and \( p_u^0 \) cannot be determined without knowledge of the system state after the \((n-1)\)th unsuccessful RSHW, which is too complicated to derive a priori. We will approximate these probabilities using the following useful properties of a TMR system. Since the probability of permanent faults having caused the TMR failure increases with the number of unsuccessful RSHW’s, \( p_s^0 \) is monotonically decreasing in \( n \):

\[
\frac{p_s^0}{p_u^0} < \cdots < p_k \iff p_i^0 < p_u^0 < \cdots < p_k^0.
\]

Though \( p_i^0 \) and \( R(n) \equiv p_s^{n-1}/p_u^n \) depend upon \( X \) and fault parameters, it is assumed for simplicity that \( p_i^0 \) is given a priori as a constant \( P \) and \( R(n) \) is a constant \( R \) for all \( n \). \( C_1(k,X) \) of (3.2) is then modified in terms of \( P \) and \( R \):

\[
C_1(k,X) = \sum_{m=1}^{k-1} mX \prod_{n=1}^{m-1} \left( 1 - PR^{n-1} \right) PR^{m-1} + \frac{T_e + X}{1 - F_1(X)} \prod_{n=1}^{k} \left( 1 - PR^{n-1} \right). \tag{3.3}
\]

The cost of RHWR, denoted by \( C_2(X) \), is derived by using recursive equations:

\[
C_2(X) = \frac{T_e + X}{1 - F_1(X)}. \tag{3.4}
\]

The outcome, of one RSHW, regardless whether it is successful or not, is used to update the likelihoods of states in one of which (called a prior state) the RSHW started. The possible states upon detection of a TMR failure can be inferred from the posterior states which are the updated prior states using the Bayes theorem.

**Adaptive RSHW**

In this method, the system chooses, upon detection of a TMR failure, between RSHW and RHWR based on their expected costs. RSHW will continue either until it becomes successful or until the expected cost of the next RSHW becomes larger than that of RHWR. The system state is characterized by the likelihoods of all possible states because one can observe only the time of each TMR failure detection, which is insufficient to accurately estimate the system state. The outcome of one RSHW, regardless whether it is successful or not, is used to update the likelihoods of states in one of which (called a prior state) the RSHW started. The possible states upon detection of a TMR failure can be inferred from the posterior states which are the updated prior states using the RSHW result and the Bayes theorem.
Unlike a simplex model, there are too many possible states and events to analyze a TMR system accurately. We will thus use the simplified Markov-chain model in Fig. 3 to derive the state probabilities and transition probabilities in a TMR system. The model consists of six states which are distinguished by the number of permanent faults and that of nonpermanent faults, where two- and three-fault states are merged into one state due to their identical effects in our analysis. In Fig. 3, the transitions over the bidirectional horizontal lines result from the behavior of nonpermanent faults and the transitions over the unidirectional vertical lines are caused by the occurrence of permanent faults. Note that even occurrences of near-coincident faults can be represented by sequential occurrences with slightly different interarrival times. The model, thus, includes only transitions between neighboring states—any transition from a state due to multiple faults occurs in two steps through one of its neighboring states.

Some faults may disappear without affecting the execution of a task. This happens when the latency of a fault is greater than its active duration, i.e., it will not manifest itself. Note that the occurrence of an error in a module during the task execution may produce an erroneous output for the task, even if the fault which had induced the error disappeared before producing the final output of the task. In other words, a transient fault may have permanent effects on task execution.

The optimal recovery algorithm based on the adaptive method in Fig. 4 can be illustrated as follows. Upon detection of a TMR failure, the first step is to derive the probabilities of all possible states at time $X_f$ evolved from each prior state. Let $T_j^i$ be the time when the TMR system moved to the failure state from prior state $i$ during $[0, X_f]$, where $X_f$ is the time of detecting a TMR failure (i.e., a voting time). Occurrence of a TMR failure is then represented by an event $(T_j^i \leq X_f)$ for prior state $i$. We want to calculate the probabilities of all possible states $\pi^i_n(X_f)$ at voting time $X_f$ evolved from prior state $i$, which are actually conditional probabilities given the observed event $(T_j^i \leq X_f)$. They can be calculated from the probabilities of all types of TMR failures $\pi^i_n(T_j^i)$ at time $T_j^i$ and the transition probabilities $P_{mn}(X_f - T_j^i)$ during the remaining task-execution time, $X_f - T_j^i$. The probabilities of all possible states are thus

$$\pi^i_n(X_f) = \sum_i \sum_m \pi^i_n(0) \int_0^{X_f} P_{mn}(X_f - t) \pi^i_n(t) dF_{T_j^i}(t),$$

where subscripts $i, m$ and $n$ indicate the prior state, the state at time $T_j^i$, and the state at the time, $X_f$, of detecting a TMR failure, respectively. As mentioned earlier, a voting failure may result from a voter fault or multiple-module faults. Multiple-module faults can be classified based on the number of modules with permanent faults: Type-I, Type-II, and Type-III failures represent zero, one, and more than one permanent-fault module, respectively, where all possible states of each type are listed in Fig. 3. Let $S(x, y)$ be the state with $x$ permanent-fault modules, $y$ nonpermanent-fault modules, and $3 - x - y$ nonfaulty modules.

Although there are ten different states, we only need to consider six of them by merging 1) $S(0, 3)$ into $S(0, 2)$, 2) $S(1, 2)$ into $S(1, 1)$, and 3) both $S(2, 1)$ and $S(3, 0)$ into $S(2, 0)$. This merger of states simplifies the model of a TMR system without losing model accuracy, because:

- by modifying the transition rates, one can make the simplified Markov-chain model in Fig. 3 represent a TMR system very accurately, and
- the merger is based on a realistic assumption that simultaneous occurrence of faults in different processor modules is highly unlikely.

Moreover, the merger does not change the analysis of a TMR failure because merged states have similar effects on the TMR failure as compared to the original states. For example, the merged states induce the same type of TMR failure, where the “type” is determined by the number of permanent-fault modules. There are four possible states, $\{S(0, 0), S(0, 1), S(0, 2), S(0, 3)\}$ at time $X_f$, which led to Type-I failures (i.e., it was $S(0, 1)$, $S(0, 2)$, or $S(0, 3)$ at time $T_j^i$, because a nonpermanent fault might disappear after inducing error(s)). Type-II and Type-III failures have three possible states, $\{S(1, 0), S(1, 1), S(1, 2)\}$ and $\{S(2, 0), S(2, 1), S(3, 0)\}$, respectively, at time $T_j^i$ and $X_f$.

For notational simplicity, let state $S_i \equiv S(x, y)$ where $i = 4x + y$. Then, the set of all possible states after the merging

![Fig. 3. A simplified Markov-chain model for a TMR system.](image-url)
is \(\{S_i : i = 0, 1, 2, 4, 5, 8\}\); out of these, \(\{S_1, S_2, S_3, S_4\}\) are the set of possible fault states transited from \(S_0\), \(S_1\), and \(S_2\) at time \(T_j^1\), \(T_j^2\), and \(T_j^3\), respectively. \(S_4\) and \(S_5\) may change to \(S_6\) (or \(S_8\)) at \(T_j^2\) (or \(T_j^3\)), and \(S_8\) remains unchanged due to the persistence of a permanent fault.

Let a path denote the transition trajectory between a pair of states. Since there are usually more than one path between a given pair of nodes, each of these paths is assigned an ID number. From the simplified model in Fig. 3, \(T_j\) is the minimum-time path from \(S_i\) to any type of TMR failure. Let \(t_j^i\) be the time taken from \(S_i\) to a TMR failure via path \(j\). Then, \(T_j = \min_j[t_j^i]\), where the pdf of \(t_j^i\) is calculated by convolving the pdf’s of all subpaths that make up path \(j\). The pdf of a subpath between two states \(S_j\) and \(S_{j+1}\) is obtained by using the distribution of sojourn time \(t_j\) of \(S_j\) with several exits in the Markov chain model (Fig. 3):

\[
f_{ij}(t) = \int_{E_j} f_{ij}^t(t) \, dt
\]

where \(\{E_j\}\) represents the set of all outgoing arcs of \(S_j\).

Then, the pdf of \(t_j^i\) is

\[
f_{ij}^t(t) = f_{i,j}(t) \cdot f_{ij}(t) \cdot \cdots \cdot f_{jm}(t),
\]

where path \(j\) is composed of subpaths \(\{j_1, j_2, \ldots, j_m\}\) and \(S_m\) must be one of possible fault states: \(S_m \in \{S_1, S_2, S_4, S_5, S_8\}\). (When the inter-arrival time of events such as fault occurrence, fault disappearance, and fault latency, is not exponentially distributed, we need a semi-Markov chain model in place of a Markov chain model.) Let \(J_n\) represent the set of all paths to a fault state \(S_m\) from \(S_i\). The likelihood of a fault state \(S_m\) at time \(T_j\) is, then, equal to \(\sum_{j \in J_n} \text{Prob}(t_j^i = T_j^i)\), which is obtained by:

\[
\pi^i_m(T_j^i) = \sum_{j \in J_n} \frac{f_{ij}(t_j^i)}{\sum_{y \in E^i} f_{ij}(t_j^i)},
\]

where \(E^i\) is the set of all paths to all possible fault states evolved from \(S_i\), i.e., \(E^i = \bigcup_{m=1}^m J_m\), and \(m \in \{1, 2, 4, 5, 8\}\). The probabilities of \(S_1\) and \(S_2\) leading to Type-I failure are computed based on the behavior of nonpermanent faults, i.e., depending on whether or not a nonpermanent fault, after having induced some error(s), is still active when a second nonpermanent fault occurs. Likewise, the probabilities of \(S_4\) and \(S_5\) leading to Type-II failure are computed by the behavior of a nonpermanent fault, if it had occurred earlier than permanent fault(s). When an intermittent fault is considered, the fault state must be divided by fault active and fault benign states as in [15], which makes the problem too complicated to be tractable. The numerical examples of \(F_{T_j^i}(X)\) and the mean of \(T_j^i\) (\(i = 0, 4\)) for several \(X\) are given in Figs. 5 and 6, in which analytic results are compared against the results obtained from Monte-Carlo simulations.

In addition to \(F_{T_j^i}\) and \(\pi^i_m\), the transition probabilities \(P_{mn}\) from \(S_m\) to \(S_n\) during \(X_f - T_j^i\) must be derived in order to obtain the likelihood of every possible state at the time of voting (failure detection), \(X_f\). Although the matrix algebra using the transition matrix or Chapman-Kolmogrov theorem can be applied to give accurate expressions, we will use a simplified method for computational efficiency at an acceptably small loss of accuracy. For the transition probabilities from \(T_j\), we need not consider subsequent errors but can focus on only those states useful in choosing between RSHW and RHWR.

Observe that the occurrence rate, \(\lambda_p\), of permanent faults is much smaller than both the appearance and disappearance rates of nonpermanent faults. Using this observation, one can analyze the behavior of permanent faults separately from that of nonpermanent faults. The transition probabilities due to the occurrence of permanent faults are represented by \(P_{mn}(X_f - T_j)\) for \(S_m \in \{S_1, S_2, S_4, S_5, S_8\}\), that is, \(P_{mn}(X_f - T_j) = 0\) for \(S_m \in \{S_1, S_2, S_4, S_5, S_8\}\), because of the persistence of permanent faults. Although these probabilities depend upon \(\pi^i_m(T_j)\) \(\forall t\), \(T_j \leq t \leq X_f\), they are approximated by using only the prior probabilities of source states, \(\pi^i_m(T_j)\). This approximation causes only a very small deviation from the exact values because the occurrence rate of permanent faults is usually very small as compared to the other rates. For example, consider \(P_{mn}\) for \(n \geq 4\), i.e., transitions from \(S_1\) due to the occurrence of permanent fault(s). The corresponding transition probabilities are derived from the model in Fig. 3 in terms of the pdf’s of subpaths between two states. Let \(T = X_f - T_j\), then

\[
P_{18}(T) = \int_0^T F_{58}(T - t) f_{15}(t) \, dt
\]

\[
P_{13}(T) = \left(\frac{2\lambda_p}{2\lambda_p + \mu} + \frac{\mu}{2\lambda_p + \mu} e^{-(\lambda_p + \mu)(X_f - T_j)}\right) \cdot \int_0^T (1 - F_{58}(T - t)) f_{15}(t) \, dt
\]

\[
P_{14}(T) = \left(\frac{\mu}{2\lambda_p + \mu} - \frac{\mu}{2\lambda_p + \mu} e^{-(\lambda_p + \mu)(X_f - T_j)}\right) \cdot \int_0^T (1 - F_{58}(T - t)) f_{15}(t) \, dt.
\]

The probability \(\pi^i_m(T_j)\) for \(S_1\) is thus reduced to \((1 - F_{15}(T))\pi^i_m(T_j)\). Likewise, transitions from other source states due to the occurrence of permanent faults can be derived. Consequently, the prior probabilities are transformed into \((1 - F_{25}(T))\pi^i_m(T_j), (1 - F_{58}(T))\pi^i_m(T_j)\), and \((1 - F_{58}(T))\pi^i_m(T_j)\), respectively. Using these transformed prior probabilities, we will derive the transition probabilities based only on the behavior of nonpermanent faults.

Considering only the behavior of nonpermanent faults divides the above model into a two-state model \(\{S_4, S_5\}\) and a three-state model \(\{S_0, S_1, S_2\}\), as shown in Fig. 3. The transition matrix of the three-state model \(\{S_0, S_1, S_2\}\) is derived by 1) using the Laplace transform which reduces
the linear differential equations of three states to algebraic
equations in s, 2) solving the algebraic equations, and 3)
transforming the solution back into the time domain.

The linear differential equation of \( \{S_0, S_1, S_2\} \) with only the
effects of nontransient faults is

\[
A(X_f) = T(X_f - T_j)A(T_j),
\]

where

\[
T = \begin{bmatrix}
-3\lambda_n & \mu & 0 \\
3\lambda_n & -2\lambda_n\mu & 2\mu \\
0 & 2\lambda_n & -2\mu
\end{bmatrix}
\]

The Laplace transform of \( T \) is:

\[
A = \begin{bmatrix}
s + 3\lambda_n & -\mu & 0 \\
-3\lambda_n & s + 2\lambda_n\mu & -2\mu \\
0 & -2\lambda_n & s + 2\mu
\end{bmatrix}
\]

The solution requires the inverse of \( A \) (found at the bottom
of the page).

Let the roots of \( s^2 + (5\lambda_n + 3\mu)s + 6\lambda_n^2 + 6\lambda_n\mu + 2\mu^2 \)
be \( \alpha \) and \( \beta \), then \( a_{ij} \), the \( ij \)th element of \( A \), can be obtained by
partial fraction expansion:

\[
a_{ij} = \frac{c(ij)_1}{s + \alpha} + \frac{c(ij)_2}{s + \alpha} + \frac{c(ij)_3}{s + \beta}.
\]

Since \( c(ij)_2 \) and \( c(ij)_3 \) are conjugates, \( c(ij)_2 = k_{ij}(\alpha, \beta) \) if
\( c(ij)_3 = k_{ij}(\beta, \alpha) \). The effect of permanent faults changes the
initial probabilities of \( \{S_0, S_1, S_2\} \) to:

\[
\Pi'(T_j) = [A_0\pi_0(T_j), A_1\pi_1(T_j), A_2\pi_2(T_j)]^T,
\]

where \( A_0 = (1 - F_{81}(T)) \), \( A_1 = (1 - F_{12}(T)) \), \( A_2 = (1 - F_{23}(T)) \). Thus, the \( it \)th column of the \( 3 \times 3 \) transition
matrix \( P(T) \) reduces to:

\[
\begin{bmatrix}
\frac{2\lambda_n^2}{\alpha\beta} + k_{11}(\alpha, \beta)e^{-\alpha T} + k_{11}(\beta, \alpha)e^{-\beta T}A_{i-1} \\
\frac{6\lambda_n\beta}{\alpha\beta} + k_{22}(\alpha, \beta)e^{-\alpha T} + k_{22}(\beta, \alpha)e^{-\beta T}A_{i-1} \\
\frac{6\lambda_n\alpha}{\alpha\beta} + k_{33}(\alpha, \beta)e^{-\alpha T} + k_{33}(\beta, \alpha)e^{-\beta T}A_{i-1}
\end{bmatrix},
\]

where

\[
k_{11}(x, y) = \frac{x^2 + (2\lambda_n + 3\mu)x + 2\mu^2}{x(y - x)},
\]

\[
k_{22}(x, y) = \frac{x^2 + (3\lambda_n + 2\mu)x + 6\lambda_n\mu}{x(y - x)},
\]

\[
k_{33}(x, y) = \frac{x^2 + (5\lambda_n + \mu)x + 6\lambda_n^2}{x(y - x)},
\]

The above equations indicate that the coefficients of exponen-
tials in \( A_0, A_1, \) and \( A_2 \) include the effects of the occurrence
of permanent fault(s) on the prior probabilities. Likewise, the
transition matrix of a two-state model for \( \{S_4, S_5\} \) can be
derived from the matrix found at the bottom of the page where
\( A_4 = 1 - F_{85}(T) \) and \( A_5 = 1 - F_{58}(T) \) also represent the
effects of permanent-fault occurrences on the transitions to
\( S_6 \). These transition matrices and probabilities (resulting from
the occurrence of permanent faults) can describe all possible
transitions in the simplified model of Fig. 3.

When the TMR system is in \( S_0, S_5 \) or \( S_8 \) at time \( X_f \),
RSHW will be unsuccessful again due to multiple active faults
(in more than one module). If it is not in those states at
time \( X_f \) due to disappearance of active fault(s) after inducing
some error(s), the system moves to a recoverable state by
RSHW. Let \( F_{cI}(X) \) be the probability of a TMR failure
evolved from \( S_i \) during the execution time \( X \), where \( F_{cI} \) is the
probability distribution function of \( T_f \). Since exact knowledge
of the system state is not available, we estimate the state
probabilities, which are then used to calculate the expected
cost of a single RSHW as follows:

\[
C_1(X) = X + \frac{T_c + X}{1 - F_{cI}(X)} \left( \sum_{i \in \{3, 5\}} \pi_i(0) + \sum_{i \in \{0, 1, 4\}} F_{cI}(X)\pi_i(0) \right),
\]

where \( \pi_i(0) \) is the probability that the state before starting
one RSHW (upon detecting a TMR failure) is \( S_i \), i.e., the
probabilities of the present states become those of the prior
states for the next RSHW. The expected cost of RHWR is
obtained similarly to (3.4):

\[
C_2(X) = \frac{T_c + X}{1 - F_{cI}(X)}.
\]

When RSHW is unsuccessful or a voting failure occurs
again, the (prior) state probabilities are updated with the
additional information obtained from the RSHW using the Bayes theorem. The observed information tells us that a TMR failure has occurred again during the current execution. (Note that the TMR failure detection time during the current execution is \(X_f\).) As a result, the prior probabilities of all possible fault states for the \((k + 1)\)th RSHW \((\pi_k^{k+1})\) are renewed from those of the \(k\)th RSHW \((\pi_k^k)\):

\[
\pi_{k+1} = \frac{\pi_k^k \text{Prob}(\text{a TMR failure during } X_f \text{ from } S_k)}{\text{Prob}(\text{a TMR failure during } X_f)}
\]

(3.10)

where \(\text{Prob}(\text{a TMR failure during } X_f) = \sum_{i \in S} \pi_k^k \text{Prob}(\text{a TMR failure during } X_f \text{ from } S_k) = \sum_{i \in S} \pi_k^k F_{ij}(X_f)\). From (3.10), one can see that the probability of the TMR system being in a permanent-fault state increases with each unsuccessful RSHW, which, in turn, increases the chance of adopting RHWR over RSHW upon detection of next TMR failure. Using the above updated state probabilities, we can get the conditional probabilities of all states upon detection of a TMR voting failure.

When RSHW is successful, one can likewise update the probabilities of possible states, which will then be used to guess the prior state of the next voting interval.

When the hardware cost is high and the time constraint is not stringent, one may do the following. Since the fault occurrence rate is much smaller than the disappearance rate of (existing) nonpermanent faults, we may wait for a certain period of time (called a back-off time) in order for the current nonpermanent fault(s) to disappear before task re-execution. An optimal back-off time is determined by minimizing the expected time overhead. When a task is re-executed without any back-off, the cost of one RSHW is equal to (3.8). When re-execution starts after back-off for \(r\) units of time, the cost changes (due to the change of prior states):

\[
C_1(r) = X + r + \frac{T_r + X}{1 - F_{T_r}(X)} \left( \sum_{i \in \{2,5,8\}} \pi_i(r) + \sum_{i \in \{0,1,4\}} F_{T_j}(X) \pi_i(0) \right),
\]

where \(\pi_i(r) = \sum_{j \in S} F_{ij}(r) \pi_i(0)\).

The optimal back-off time is obtained by minimizing \(C_1(r)\) with respect to \(r\).

IV. NUMERICAL RESULTS AND DISCUSSION

A system with three replicated processing modules, two voters, and a comparator is simulated to compare the proposed method (called Method 1) with an alternative which is based on RHWR (called Method 2). Upon detection of a TMR failure, Method 1 will decide between RSHW and RHWR according to their respective costs. Method 2, however, will reconfigure the TMR entirely with a new healthy TMR or partially with healthy spare modules following an appropriate diagnosis. If a nonpermanent fault does not disappear during the diagnosis, it will be treated as a permanent fault and replaced by a new, nonfaulty spare. We assume that (A1)

![Fig. 4. Algorithm to recover from a TMR failure by estimating the system state and comparing the costs of RSHW and RHWR.](image)

**Table III**

| Parameter Values Used in Simulations, All Measured in Hours |
|-----------------|-----------------|-----------------|-----------------|
| \(\frac{1}{X}\)  | \(\frac{1}{X_f}\) | \(\mu\)          | \(\rho\)        | \(X^*\)         |
| 200             | 3000            | 0.0001          | 0.002           | 50              |

an unlimited number of tasks with the same nominal task-execution time are available to keep the running module busy, which simplifies the description of system workload, and (A2) there are an unlimited number of spares available. The performances of these two methods are characterized by the overhead ratio:

\[
\text{OVR}(X) \equiv \frac{E - X}{X},
\]

where \(E\) is the real execution time (including the RSHW and/or RHWR overheads) of a task whose nominal execution time is \(X\).

We ran simulations under the fault generation process with the parameters as given in Table III, where the asterisk (*) indicates a parameter varied while the others are fixed, in order to observe the effects of the parameter on OVR in both methods. Since fault occurrence/disappearance rates are difficult to estimate on-line, some experimental data or numerical data based on a model reflecting the maturity of design/fabrication process, the environmental effects, operating conditions, and the number and ages of components, can be used [19].

In Figs. 5 and 6, the probabilities of a TMR failure and the failure times from \(S_0\) and \(S_4\) are computed from the Markov-chain model and simulations, and are then compared. The simulation and modeling results are very close to each other. The modeling analyses proved to be very effective in determining when and how to choose between RSHW and RHWR under various conditions, as shown in Figs. 7–11.

The results obtained while varying \(X\) from 10 to 100 hours with \(T_r = 0.15X\), are plotted in Figs. 7–9. The OVR’s of Methods 1 and 2 with the optimal number of votings are compared in Fig. 7. The difference between the OVR’s of Method 1 and Method 2 increases significantly with \(X\). When
Fig. 5. Probability/Frequency of a TMR failure obtained from the Markov-chain model (P:S0=from S0 and P:S4=from S4) and from simulations (F:S0=from S0 and F:S4=from S4).

Fig. 6. Mean TMR failure time (E[T]) obtained from analysis (P:S0=from S0 and P:S4=from S4), and from simulations (F:S0=from S0 and F:S4=from S4).

Fig. 7. Overhead ratios [%] vs. X for RSHW and RHWR, with the optimal number of votings for Tp=0.0005 hour: (13, 34, 61, 87, 110, 133, 164, 181, 198, 216).

X is small, the OVR's of the two methods are too small to distinguish, which is due mainly to the small probability of a TMR failure. Fig. 8 compares the multivoting policy (with the optimal number of votings) and one voting policy. Generally, the overhead of a TMR system with infrequent voting increases significantly as X increases, because the probability of a TMR failure increases with X; e.g., if there is no voting during the task execution, a TMR failure means the waste of the entire nominal execution time, X. As X increases, the OVR of a one-voting policy increases more rapidly than that of multivoting policy. The number of RHWR's—which is represented by the percentage of RHWR from the total number of simulations in Fig. 9—will determine the hardware cost of spares used. The increase in this percentage is much larger in Method 2 than Method 1, since the number of TMR failures increases with X, and Method 1 can recover from most TMR failures with RSHW.

The second comparison is made while varying Tp—the resetting time for system reconfiguration—from 2.5 to 12.5
hours for $X = 50$ hours, and the results are plotted in Fig. 10. A larger resetting time generally results in a larger OVR. Increasing $T_c$ greatly affects the performance of Method 2. But, it has little influence on the OVR of Method 1, since the system recovers from most TMR failures with RSHW, which has nothing to do with $T_c$.

The third comparison in Fig. 11 is made while varying $\lambda / \lambda_0$ from 5 to 25, where $\lambda_0$ is fixed at 0.005/hr, and $X = 50$ hours and $T_c = 7.5$ hours. The OVR's of both methods decrease with $\lambda / \lambda_0$, but the magnitude of decrease in Method 1 is larger than that in Method 2. This is because the probability of a TMR failure decreases as $\lambda_0$ decreases with $\lambda_0$ fixed, and because the probability of successful RSHW increases with $\lambda / \lambda_0$.

We simulated the proposed and other schemes for $10^5$ units of time with the fault parameters of Table III for each comparison (of the mean overhead ratios of different schemes). The fault parameters are assumed not to change during the simulation. Since the estimation of system states depends upon the fault parameters, they must be estimated first. This problem can be solved by assuming the parameters to be time-varying and estimating them on-line with certain adaptive methods which, in turn, require more samples.

APPENDIX

LIST OF SYMBOLS

$X$: Nominal task-execution time in the absence of failures, i.e., the amount of pure computation for a task measured in CPU cycles without including repetition of part of the task due to failures.

$X_i$: Nominal execution time for the task between the $i$th and $(i-1)$th voting.

$W_n(X)$: Expected execution time of a task whose nominal execution time is $X$.

$w_i$: Actual execution time from the beginning of the task to the first completion of the $i$th voting, where $W_i = E(w_i)$.

$V_i$: Actual execution time during the interval $[X_{i-1}, X_i]$.

$p(q)$: Probability of recovering a task with RSHW (RHWR), $p + q = 1$.

$T_c$: Resetting time in case of system reconfiguration.

$T_p$: Time for voting on those variables changed during the previous voting interval.

V. CONCLUSION

In this paper, we have proposed a strategy for recovering TMR failures using two different methods that determine when and how to apply RHWR. Both methods are shown to outperform the conventional method based solely on reconfiguration. This finding is consistent with the fact that most faults are nonpermanent, so simple re-execution can recover from nonpermanent faults and the TMR structure can mask the effects of one faulty module.

The distinct characteristic of the proposed strategy is that it uses the estimated state of a TMR system even with incomplete observation of system states. Detection of a TMR failure and/or an unsuccessful RSHW does not always call for reconfiguration (RHWR) but requires us to derive and compare the expected costs of reconfiguration and one additional RSHW. Most TMR failures are represented by using a simplified Markov-chain model, and the TMR failure time and the probability of another unsuccessful RSHW are also analyzed with the model. One can therefore conclude that combining time and spatial redundancy appropriately can be effective in handling component failures.
\[ p^\alpha_i(p^\alpha_i) : \text{Probability of the } n\text{th RSHW being successful (unsuccessful).} \]
\[ P : \text{Probability of the first RSHW being successful.} \]
\[ R : \text{Ratio of the probability of success at the } (n+1)\text{th RSHW to that at the } n\text{th RSHW.} \]
\[ k_m : \text{Allowable maximum number of RSHW's.} \]
\[ X_f : \text{Time of detecting a TMR/voting failure.} \]
\[ T^*_f : \text{Time to a TMR system failure occurred first after starting the system in state } S_i. \]
\[ F^*_f(X) : \text{Probability of a TMR failure from } S_i \text{ during the execution time } X \text{ (} f^*_f \text{ is pdf of } T^*_f). \]
\[ t^*_f : \text{Time of TMR failure occurrence via path } j \text{ from } S_i \text{ (} f^*_f \text{ is pdf of } t^*_f). \]
\[ S(x,y) : \text{State with } x \text{ permanent faulty processor(s), } y \text{ nonpermanent faulty processor(s),} \]
\[ (3-x-y) \text{ nonfaulty processor(s) } (S_i = S(x,y)) \text{ such that } i = 4x+y. \]
\[ J^*_m : \text{Set of all paths to a fault state } S_m \text{ from an initial state } S_i \text{ (} E^i = \bigcup J^*_m). \]
\[ \pi_m(0) : \text{Probability of a prior state before the first RSHW.} \]
\[ P_m(T^*_f) : \text{Probability of a fault state } S_m \text{ at time } T^*_f \text{ from an initial state } S_i. \]
\[ C_1(k,X) : \text{Expected cost of RSHW with a nominal task-execution time } X \text{ and MNRA } k. \]
\[ C_1(X)(C_2(X)) : \text{Expected cost of RSHW (RHWR) for } X. \]
\[ F_{f_{j+1}} \text{ : Distribution of time to move to } S_j(k+1) \text{ from } S_j. \]
\[ \{E_j\} : \text{Set of all subpaths emanating from } S_j. \]
\[ \lambda_m(\lambda_p) : \text{Occurrence rate of nonpermanent (permanent) faults.} \]
\[ \frac{1}{\mu} : \text{Active duration of a nonpermanent fault.} \]

**ACKNOWLEDGMENT**

The authors would like to thank A. White, C. Meissner, and F. Pitts of the NASA Langley Research Center, and J. Smith of the Office of Naval Research for their technical and financial assistance.

**REFERENCES**

Kang G. Shin (S'75-M'78-SM'83-F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively. He is a Professor of Electrical Engineering and Computer Science for the Computer Science and Engineering Division, The University of Michigan, Ann Arbor, MI. He also chaired the CSE Division for three years beginning 1991. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the U.S. Air Force Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA. He has also been applying the basic research results of real-time computing to manufacturing-related applications ranging from the control of robots and machine tools to the development of open architectures for manufacturing equipment and processes. Recently, he has initiated research on the open-architecture Information Base for machine tool controllers.

Dr. Shin has authored/coauthored over 270 technical papers (about 130 of these in archival journals) and several book chapters in the area of distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. In 1987, he received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called HARTS, to validate various architectures and analytic results in the area of distributed real-time computing. He was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems, a Program Co-Chair for the 1992 International Conference on Parallel Processing, and served numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991-93, is a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED COMPUTING, and an Area Editor on International Journal of Time-Critical Computing Systems.

Hagbae Kim (S'90-M'94) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, in 1990 and 1994, respectively. Currently, he is a Research Associate at NASA Langley Research Center, Hampton, VA. His current research interests include real-time control systems, fault-tolerant computing, reliability modeling, and probability and stochastic processes.