To Appear in IEEE Transactions on Computers
vol. 43, no. 10, pp. 1151–1162, October, 1994

A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods[1]

Kang G. Shin and Hagbae Kim

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109–2122

Abstact

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 (i) the TMR system can mask the effects of one faulty module while RSHW can recover from non-permanent faults, and (ii) 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 RSHWs 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 non-permanent faults, reconfiguration, nominal task-execution time, likelihoods, Bayes theorem.

[1]The work reported 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 view of the funding agencies.

1 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 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 non-permanent — 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 (i) the hardware cost resulting from the hasty elimination of modules with transient faults and (ii) 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 (non-adaptive) method is to determine the maximum number of RSHWs 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 (i) estimates the system state with the likelihoods of all possible states and (ii) 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 RSHWs 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 3 derives the optimal voting interval (X_v) 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 4 presents numerical results and compares two recovery methods of RSHW and RHWR. The paper concludes with Section 5.

2 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 non-coincident 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.vmp [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

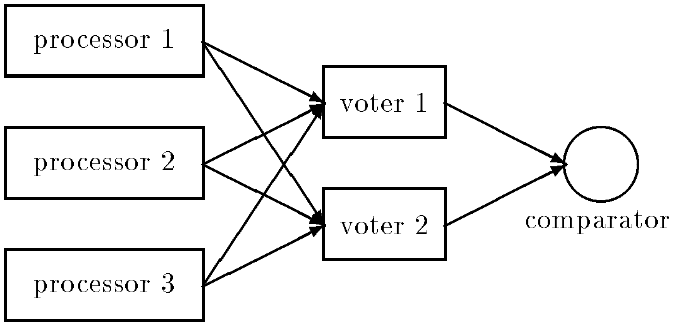

Figure 1: The structure of a TMR system with two voters and a comparator.

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 RSHWs irrespective of the nominal task-execution time and the system state which is defined by the number of faulty modules and the fault type(s). Taking into account the fact that the time overhead of an unsuccessful RSHW increases with the nominal task-execution time X, one can determine the MNR simply based on X, without estimating the system state. A more complex, but more effective, method is to decide between RSHW and RHWR based on the estimated system state. Since the system state changes dynamically, this decision is made by optimizing a certain criterion which is dynamically modified with the additional information obtained from each unsuccessful RSHW. In this adaptive method, the probabilities of all possible states will be used instead of one accurately-estimated state. Upon detection of a TMR failure, the expected cost of RSHW is updated and compared with that of RHWR. The failed task will then be re-executed, without replacing any module, either until RSHW recovers from the corresponding TMR failure or until the expected cost of RSHW becomes larger than that of task execution.[2] As the number of unsuccessful RSHWs increases, the possibility of permanent faults having caused the TMR failure increases, which, in turn, increases the cost of RSHW significantly.

[2]This procedure is described in the algorithm of Fig. 4.

Throughout this paper, we assume that the arrival of permanent faults and the arrival and disappearance of non-permanent faults are Poisson processes with rates λ_p, λ_n, and μ, respectively.

3 Optimal Recovery from a TMR Failure Using RSHW

3.1 The Optimal Voting Interval

Let X_i $(2 \leq i \leq n)$ be the nominal task-execution time measured in CPU cycles between the $(i-1)$-th and i-th 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 $1 \leq i \leq n$ let w_i represent the task-execution time from the beginning of the task to the first completion of the i-th voting possibly in the presence of some module failures, and let $W_i = E(w_i)$. Then $E(w_n) = W_n$ is the expected execution time of the task. Upon detection of a TMR failure, let p and q be the probabilities of recoverying 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, $1 \leq i \leq n$. Let $F_i(t)$ $(2 \leq i \leq n)$ 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 $F_1(t)$ 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,

$$w_1 = \begin{cases} X_1 & \text{with probability } 1 - F_1(X_1) \\ X_1 + w_1 & \text{with probability } F_1(X_1)p \\ X_1 + T_c + w_1 & \text{with probability } F_1(X_1)q. \end{cases}$$

where T_c is the setup time for system reconfiguration.

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

$$w_i = w_{i-1} + V_i + T_v \quad \text{for } 2 \leq i \leq n,$$

where V_i is defined as the actual task-execution time between the $(i-1)$-th and i-th votings,

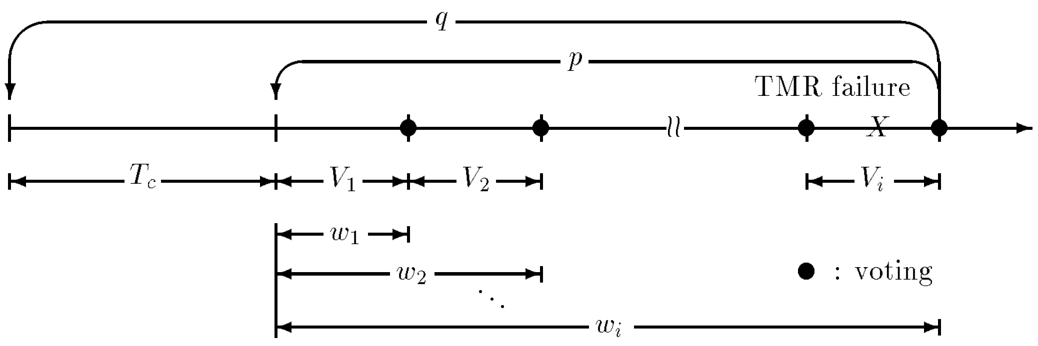

Figure 2: Graphical explanation for V_i and w_i for $1 \leq i \leq n$.

i.e., $V_1 = w_1$ 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 $2 \leq i \leq n$:

$$W_i = \frac{1}{1 - F_i(X_i)} \left(W_{i-1} + T_v + X_i + F_i(X_i) q T_c \right).$$

Applying this recursively $n-1$ times, we can get:

$$W_n = \sum_{j=1}^{n} \prod_{i=j}^{n} \left(\frac{F_j(X_j) q T_c + X_j + T_v}{1 - F_i(X_i)} \right). \tag{3.1}$$

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 $X_i = X_v = \frac{X}{n}$ for $1 \leq i \leq n$, where an optimal value of n must be determined by minimizing Eq. (3.1). Examples of n for a given X with typical values of p, q, T_v, and T_c are shown in Table 1. The voting points can be inserted by a programmer or a compiler.

3.2 Pre-determination of Non-adaptive RSHWs

In the first method, we determine *a priori* the maximum number of RSHWs (MNR), k_m, based on X without estimating the system state. The associated task will be re-executed up to k_m times. As X increases, the effect of an unsuccessful RSHW becomes more pronounced; that is, the possibility of successful recovery with RSHW (instead of RHWR) will decrease with X due to the increased rate of TMR failures, and the time overhead of an unsuccessful RSHW also increases with X while the time overhead of RHWR remains constant. So, k_m decreases as X increases.

Let $C_1(k, X)$ be the actual time/cost of task execution in the presence of up to k RSHWs for a task with the nominal execution time X, which can be expressed as:

$$\begin{aligned} C_1(k, X) &= Xp_s^1 + 2Xp_u^1p_s^2 + \cdots + kX\prod_{n=1}^{k-1} p_u^n p_s^k + \left(kX + \frac{T_c + X}{1 - F_1(X)}\right)\prod_{n=1}^{k} p_u^n, \\ &= \sum_{m=1}^{k-1} mX \prod_{n=1}^{m-1} p_u^n p_s^m + kX\prod_{n=1}^{k-1} p_u^n + \left(\frac{T_c + X}{1 - F_1(X)}\right)\prod_{n=1}^{k} p_u^n, \end{aligned} \tag{3.2}$$

where p_s^n (p_u^n) 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 $p_s^n + p_u^n = 1, \quad 1 \le n \le k$. In fact, p_s^n and p_u^n 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 RSHWs, p_s^n is monotonically decreasing in n:

$$p_s^1 > p_s^2 > \cdots > p_s^k \Longleftrightarrow p_u^1 < p_u^2 < \cdots < p_u^k.$$

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

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

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

$$C_2(X) = \frac{T_c + X}{1 - F_1(X)}. \tag{3.4}$$

Now, k_m can be determined as the integer that minimizes $C_1(k, X)$ subject to $C_1(k, X) < C_2(X)$. Example values of k_m for typical values of P and R are shown in Table 2.

3.3 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 non-permanent 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 non-permanent 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.[3]

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

[3] In fact, this problem can be eliminated by resynchronizing the processors after a transient fault is detected [21]. This, however, requires frequent voting and additional mechanisms for detecting errors in each processor and resynchronizing the processors.

of all possible states at time X_f evolved from each prior state. Let T_f^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 $(T_f^i \leq X_f)$ for prior state i. We want to calculate the probabilities of all possible states $\pi_n^i(X_f)$ at voting time X_f evolved from prior state i, which are actually conditional probabilities given the observed event $(T_f^i \leq X_f)$. They can be calculated from the probabilities of all types of TMR failures $\pi_m^i(T_f^i)$ at time T_f^i and the transition probabilities $P_{mn}(X_f - T_f^i)$ during the remaining task-execution time, $X_f - T_f^i$. The probabilities of all possible states are thus

$$\pi_n(X_f) = \sum_i \sum_m \frac{\pi_i(0)}{F_{T_f^i}(X_f)} \int_0^{X_f} P_{mn}(X_f - t)\pi_m^i(t)dF_{T_f^i}(t), \tag{3.5}$$

where subscripts i, m and n indicate the prior state, the state at time T_f^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 non-permanent-fault modules, and $3 - x - y$ nonfaulty modules.

Although there are ten different states, we only need to consider six of them by merging (i) $S(0,3)$ into $S(0,2)$, (ii) $S(1,2)$ into $S(1,1)$, and (iii) 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 $S(0,1)$,

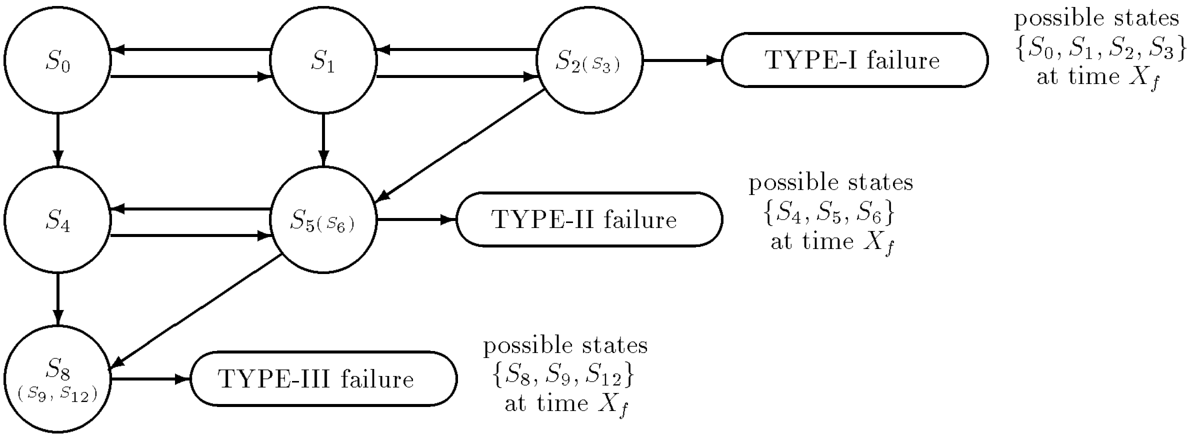

Figure 3: A simplified Markov-chain model for a TMR system.

$S(0,2)$, or $S(0,3)$ at time T_f^i, because a non-permanent 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_f^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 is $\{S_i : i = 0,1,2,4,5,8\}$; out of these, $\{S_1, S_2, S_5, S_8\}$ are the set of possible fault states transited from S_0, S_1, and S_2 at time T_f^0, T_f^1, and T_f^2, respectively. S_4 and S_5 may change to S_5 (or S_8) at T_f^4 (or T_f^5), 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_f^i 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, $T_f^i = \min_j \, [t_j^i]$, where the *pdf* of t_j^i is calculated by convolving the *pdf*'s of all sub-paths that make up path j. The *pdf* of a sub-path between two states S_{j_k} and $S_{j_{k+1}}$ is obtained by using the distribution of sojourn time t_{j_k} of S_{j_k} with several exits in the Markov chain model (Fig. 3):

$$f_{j_k j_{(k+1)}}(t) = \frac{\lambda_{j_k j_{(k+1)}}}{\sum\limits_{y \in \{E_{j_k}\}} \lambda_{j_k y}} \sum_{y \in \{E_{j_k}\}} \lambda_{j_k y} e^{-(\sum_{y \in \{E_{j_k}\}} \lambda_{j_k y})t}, \tag{3.6}$$

where $\{E_{j_k}\}$ represents the set of all outgoing arcs of S_{j_k}. Then, the *pdf* of t^i_j is

$$f_{t^i_j}(t) = f_{ij_1}(t) * f_{j_1j_2}(t) * \cdots * f_{j_fm}(t),$$

where path j is composed of sub-paths $\{ij_1, j_1j_2, \cdots, j_fm\}$ 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^i_m 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^i_f is, then, equal to $\sum_{j\in J^i_m} \text{Prob}(t^i_j = T^i_f)$, which is obtained by:

$$\pi^i_m(T^i_f) = \sum_{j\in\{J^i_m\}} \frac{f_{t^i_j}(T^i_f)}{\sum\limits_{y\in\{E^i\}} f_{t^i_y}(T^i_f)}, \tag{3.7}$$

where E^i is the set of all paths to all possible fault states evolved from S_i, i.e., $E^i = \bigcup\limits^{m} J^i_m$ 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 non-permanent faults, i.e., depending on whether or not a non-permanent fault, after having induced some error(s), is still active when a second non-permanent fault occurs. Likewise, the probabilities of S_4 and S_5 leading to Type-II failure are computed by the behavior of a non-permanent 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 $F_{T^i_f}(X)$ and the mean of T^i_f $(i = 0, 4)$ 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 $f_{T^i_f}$ and π^i_m, the transition probabilities P_{mn} from S_m to S_n during $X_f - T^i_f$ 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^i_f, 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, λ_p, of permanent faults is much smaller than both the appearance and disappearance rates of non-permanent faults. Using this observation, one can analyze the behavior of permanent faults separately from that of non-permanent faults.

The transition probabilities due to the occurrence of permanent faults are represented by $P_{mn}(X_f - T_f^i)$ for $S_m \in \{S(x_1, y)\}$, $S_n \in \{S(x_2, y) : x_2 > x_1\}$, that is, $P_{mn}(X_f - T_f^i) = 0$ for $S_m \in \{S(x_1, y)\}$, $S_n \in \{S(x_2, y) : x_2 < x_1\}$, because of the persistence of permanent faults. Although these probabilities depend upon $\pi_m^i(t)$ $\forall t$, $T_f^i \leq t \leq X_f$, they are approximated by using only the prior probabilities of source states, $\pi_m^i(T_f^i)$. 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_{1n} 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 sub-paths between two states. Let $T = X_f - T_f^i$, then

$$\begin{aligned}
P_{18}(T) &= \int_0^T F_{58}(T-t) f_{15}(t) dt \\
P_{15}(T) &= \left(\frac{2\lambda_p}{2\lambda_p + \mu} + \frac{\mu}{2\lambda_p + \mu} e^{-(\lambda_p + \mu)(X_f - T_f)} \right) \int_0^T (1 - F_{58}(T-t)) f_{15}(t) dt \\
P_{14}(T) &= \left(\frac{\mu}{2\lambda_p + \mu} - \frac{\mu}{2\lambda_p + \mu} e^{-(\lambda_p + \mu)(X_f - T_f)} \right) \int_0^T (1 - F_{58}(T-t)) f_{15}(t) dt.
\end{aligned}$$

The probability $\pi_1^i(T_f^i)$ for S_1 is thus reduced to $(1 - F_{15}(T))\pi_1^i(T_f^i)$. Likewise, transitions from other source states due to the occurrence of permanent faults can be derived. Consequently, the prior probabilities are transformed into $(1 - F_{25}(T))\pi_2^i(T_f^i)$, $(1 - F_{48}(T))\pi_4^i(T_f^i)$, and $(1 - F_{58}(T))\pi_5^i(T_f^i)$, respectively. Using these transformed prior probabilities, we will derive the transition probabilities based only on the behavior of non-permanent faults.

Considering only the behavior of non-permanent 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 (i) using the Laplace transform which reduces the linear differential equations of three states to algebraic equations in s, (ii) solving the algebraic equations, and (iii) transforming the solution back into the time domain.

The linear differential equation of $\{S_0, S_1, S_2\}$ with only the effects of non-transient faults is $\dot{\mathbf{\Pi}}(X_f) = \mathbf{T}(X_f - T_f^i)\mathbf{\Pi}(T_f^i)$, where

$$\mathbf{T} = \begin{bmatrix} -3\lambda_n & \mu & 0 \\ 3\lambda_n & -2\lambda_n\mu & 2\mu \\ 0 & 2\lambda_n & -2\mu \end{bmatrix}.$$

The Laplace transform of $\mathbf{T}$ is:

$$\mathbf{A} = \begin{bmatrix} s+3\lambda_n & -\mu & 0 \\ -3\lambda_n & s+2\lambda_n\mu & -2\mu \\ 0 & -2\lambda_n & s+2\mu \end{bmatrix}.$$

The solution requires the inverse of $\mathbf{A}$:

$$\mathbf{A}^{-1} = \frac{\begin{bmatrix} s^2+(2\lambda_n+3mu)s+2\mu^2 & \mu(s+2\mu) & 2\mu^2 \\ 3\lambda_n(s+2\mu) & s^2+(2\lambda_n+3\mu)s+6\lambda_n\mu & 2\mu(s+3\lambda_n) \\ 6\lambda_n^2 & 3\lambda_n(s+2\lambda_n) & s^2+(5\lambda_n+\mu)s+6\lambda_n^2 \end{bmatrix}}{s^3+(5\lambda_n+3\mu)s^2+(6\lambda_n^2+6\lambda_n\mu+2\mu^2)s.}$$

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

$$a_{ij} = \frac{c_{(ij)1}}{s} + \frac{c_{(ij)2}}{s+\alpha} + \frac{c_{(ij)3}}{s+\beta}.$$

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:

$$\mathbf{\Pi}'(T_f^i) = [A_0\pi_0(T_f^i), A_1\pi_1(T_f^i), A_2\pi_2(T_f^i)]^T,$$

where $A_0 = (1-F_{04}(T))$, $A_1 = (1-F_{15}(T))$, $A_2 = (1-F_{25}(T))$.
Thus, the i-th column of the 3×3 transition matrix $\mathbf{P}(T)$ reduces to:

$$\begin{bmatrix} (\frac{2\mu^2}{\alpha\beta} + k_{1i}(\alpha,\beta)e^{-\alpha T} + k_{1i}(\beta,\alpha)e^{-\beta T})A_{i-1} \\ (\frac{6\lambda_n\mu}{\alpha\beta} + k_{2i}(\alpha,\beta)e^{-\alpha T} + k_{2i}(\beta,\alpha)e^{-\beta T})A_{i-1} \\ (\frac{6\lambda_n^2}{\alpha\beta} + k_{3i}(\alpha,\beta)e^{-\alpha T} + k_{3i}(\beta,\alpha)e^{-\beta T})A_{i-1} \end{bmatrix},$$

where

$$\begin{aligned}
k_{11}(x,y) &= \frac{x^2+(2\lambda_n+3\mu)x+2\mu^2}{x(y-x)}, \\
k_{22}(x,y) &= \frac{x^2+(3\lambda_n+2\mu)x+6\lambda_n\mu}{x(y-x)}, \\
k_{33}(x,y) &= \frac{x^2+(5\lambda_n+\mu)x+6\lambda_n^2}{x(y-x)}, \\
k_{12}(x,y) &= \frac{\mu x+2\mu^2}{x(y-x)}, & k_{21}(x,y) &= \frac{3\lambda_n x+6\lambda_n\mu}{x(y-x)}, \\
k_{13}(x,y) &= \frac{2\mu^2}{x(y-x)}, & k_{31}(x,y) &= \frac{6\lambda_n^2}{x(y-x)} \\
k_{23}(x,y) &= \frac{2\mu x+6\lambda_n\mu}{x(y-x)}, & k_{32}(x,y) &= \frac{2\lambda_n x+6\lambda_n^2}{x(y-x)}
\end{aligned}$$

The above equations indicate that the coefficients of exponentials 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 as:

$$\begin{bmatrix} (\frac{\mu}{2\lambda_n+\mu} + \frac{2\lambda_n}{2\lambda_n+\mu} e^{-(2\lambda_n+\mu)T})A_4 & (\frac{\mu}{2\lambda_n+\mu} - \frac{\mu}{2\lambda_n+\mu} e^{-(2\lambda_n+\mu)T})A_5 \\ (\frac{2\lambda_n}{2\lambda_n+\mu} - \frac{2\lambda_n}{2\lambda_n+\mu} e^{-(2\lambda_n+\mu)T})A_4 & (\frac{2\lambda_n}{2\lambda_n+\mu} + \frac{\mu}{2\lambda_n+\mu} e^{-(2\lambda_n+\mu)T})A_5 \end{bmatrix},$$

where $A_4 = 1 - F_{48}(T)$ and $A_5 = 1 - F_{58}(T)$ also represent the effects of permanent-fault occurrences on the transitions to S_8. 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_2, 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 $F_{T_f^i}(X)$ be the probability of a TMR failure evolved from S_i during the execution time X, where $F_{T_f^i}$ is the probability distribution function of T_f^i. 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:

$$C_1(X) = X + \frac{T_c + X}{1 - F_{T_f^0}(X)} \left(\sum_{i \in \{2,5,8\}} \pi_i(0) + \sum_{i \in \{0,1,4\}} F_{T_f^i}(X) \pi_i(0) \right), \tag{3.8}$$

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 Eq. (3.4):

$$C_2(X) = \frac{T_c + X}{1 - F_{T_f^0}(X)}. \tag{3.9}$$

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 (π_i^{k+1}) are renewed from those of the k-th RSHW (π_i^k):

$$\pi_i^{k+1} = \frac{\pi_i^k \,\mathrm{Prob(a\ TMR\ failure\ during\ } X_f \mathrm{\ from\ } S_i)}{\mathrm{Prob(a\ TMR\ failure\ during\ } X_f)}, \tag{3.10}$$

where Prob(a TMR failure during X_f) $= \sum_i \pi_i^k$ Prob(a TMR failure during X_f from S_i) $= \sum_{i \in S_T} \pi_i^k F_{T_f^i}(X_f)$. From Eq. (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) non-permanent faults, we may wait for a certain period of time (called a *back-off time*) in order for the current non-permanent 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 Eq. (3.8). When re-execution starts after backing off for r units of time, the cost changes (due to the change of prior states):

$$\begin{aligned} C_1(r) &= X + r + \frac{T_c + X}{1 - F_{T_f^0}(X)} \left(\sum_{i \in \{2,5,8\}} \pi_i(r) + \sum_{i \in \{0,1,4\}} F_{T_f^i}(X) \pi_i(r) \right), \\ & \quad \text{where } \pi_i(r) = \sum_{j \in S_T} F_{ji}(r) \pi_i(0). \end{aligned}$$

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

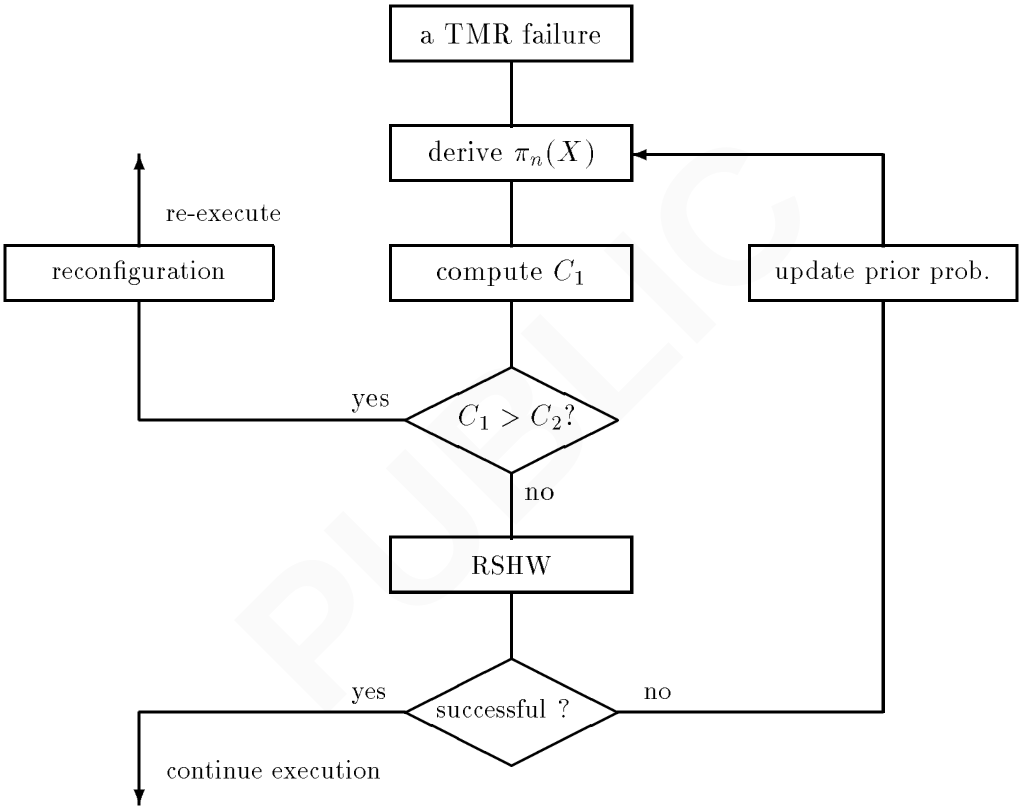

Figure 4: Algorithm to recover from a TMR failure by estimating the system state and comparing the costs of RSHW and RHWR.

X $(hr.)$	3	5	10	20	30	40
n	2	3	9	24	42	61

Table 1: n vs. X for $(p, T_c, T_v) = (0.9, 0.1X, 0.001)$

X/T_c $(hr.)$	$1/1$	$2/1$	$3/1$	$4/1$	$5/1$
k_m	4	3	2	2	1

Table 2: k_m vs. X for $(P, F, R) = (0.8, 0.1, 0.8)$

$\frac{1}{\lambda_n}$	$\frac{1}{\lambda_p}^*$	$\frac{1}{\mu}$	$\frac{1}{\mu}$	X^*
200	3000	0.0001	0.002	50

Table 3: Parameter values used in simulations, all measured in hours.

4 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 non-permanent fault does not disappear during the diagnosis, it will be treated as a permanent fault and replaced by a new, non-faulty spare. We assume that (A1) 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*:

$$\mathrm{OVR}(X) \equiv \frac{E - X}{X},$$

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 Table 3, where the symbol * 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 $T_c = 0.15X$, are plotted in Figs. 7–9. The OVRs of Methods 1 and 2 with the optimal number of votings are compared in Fig. 7. The difference between the OVRs of Method 1 and Method 2 increases significantly with X. When X is small, the OVRs 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 multi-voting 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 multi-voting policy. The number of RHWRs — 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 T_c — 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 $\frac{\lambda_n}{\lambda_p}$ from 5 to 25, where λ_n is fixed at 0.005 /hr, and $X = 50$ hours and $T_c = 7.5$ hours. The OVRs of both methods decrease with $\frac{\lambda_n}{\lambda_p}$, 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 λ_p decreases with λ_n fixed, and because the probability of successful RSHW increases with $\frac{\lambda_n}{\lambda_p}$.

We simulated the proposed and other schemes for 10^5 units of time with the fault parameters of Table 3 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.

5 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 non-permanent, so simple re-execution can recover from non-permanent 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.

Acknowledgement

The authors would like to thank Allan White, Chuck Meissner, and Felix Pitts of the NASA Langley Research Center, and Jim Smith of the Office of Naval Research for their technical and financial assistance.

References

[1] A. Avizienis and G. C. Gilley, "The STAR(self-testing and repairing) computer: An investigation of theory and practice of fault-tolerant computer design," *IEEE Trans. on Computers*, vol. C–20, no. 11, pp. 1312–1321, November 1971.

[2] M. Berg and I. Koren, "On switching policies for modular redundancy fault-tolerant computing systems," *IEEE Trans. on Computers*, vol. C–36, no. 9, pp. 1052–1062, September 1987.

[3] P. K. Chande, A. K. Ramani, and P. C. Sharma, "Modular TMR multiprocessor system," *IEEE Trans. on Industrial Electronics*, vol. 36, no. 1, pp. 34–41, February 1989.

[4] B. Cuchi, "Reliability and analysis of hybrid redundancy," in *Digest of Papers, FTCS–5*, pp. 75–79, 1975.

[5] P. T. de Sousa and F. P. Mathur, "Shift–out modular redundancy," *IEEE Trans. on Computers*, vol. C–27, no. 7, pp. 624–627, July 1978.

[6] A. L. Hopkins Jr., T. B. Smith III, and J. H. Lala, "FTMP–a highly reliable fault-tolerant multiprocessor for aircraft," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1221–1239, October 1978.

[7] M. Kameyama and T. Higuchi, "Design of dependent–failure–tolerant microcomputer system using triple–modular redundancy," *IEEE Trans. on Computers*, vol. C-29, no. 2, pp. 202–205, February 1980.

[8] D. L. Kiskis and K. G. Shin, "Embedding triple-modular redundancy into a hypercube architecture," in *Proc. of 3rd Conf. on HCCA*, pp. 337–345, Los Angeles, January 1988.

[9] I. Koren and Z. Koren, "Analysis of a class of recovery procedures," *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 703–712, August 1986.

[10] T.-H. Lin and K. G. Shin, "An optimal retry policy based on fault classification," submitted for publication, 1992.

[11] J.-C. Liu and K. G. Shin, "A RAM architecture for concurrent access and on–chip testing," *IEEE Trans. on Computers*, vol. C-40, no. 10, pp. 1153–1158, October 1991.

[12] J. Losq, "A highly efficient redundancy scheme: Self-purging redundancy," *IEEE Trans. on Computers*, vol. C–25, no. 6, pp. 569–578, June 1976.

[13] R. E. Lyons and W. Vanderkulk, "The use of triple–modular redundancy to improve computer reliability," *IBM J. Res. Develop.*, vol. 6, pp. 200–209, April 1962.

[14] S. R. McConnel, D. P. Siewiorek, and M. M. Tsao, "The measurement and analysis of transient errors in digital computer systems," in *Digest of Papers, FTCS-9*, pp. 67–70, June 1979.

[15] K. G. Shin and Y.-H. Lee, "Error detection process — model, design, and its impact on computer performance," *IEEE Trans. on Computers*, vol. C–33, no. 6, pp. 529–539, June 1984.

[16] K. G. Shin, T.-H. Lin, and Y.-H. Lee, "Optimal checkpointing of real–time tasks," *IEEE Trans. on Computers*, vol. C–36, no. 11, pp. 1328–1341, November 1987.

[17] K. G. Shin and J.-C. Liu., "Study on fault-tolerant processor for advanced launch system," *NASA Contractor Report*, June 1990.

[18] D. P. Siewiorek, V. Kini, and H. Mashburn, "A case study of C.mmp, Cm*, and C.vmp: Part I – experiences with fault tolerance in multiprocessor systems," *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1178–1199, October 1978.

[19] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Equipment Corporation, Bedford, MA, 1982.

[20] J. S. Upadhyaya and K. K. Saluja, "A watchdog processor based general rollback technique with multiple retries," *IEEE Trans. Software Eng.*, vol. SE-12, no. 1, , January 1986.

[21] J. F. Wakerly, "Transient failures in triple modular redundancy systems with sequential modules," *IEEE Trans. on Computers*, vol. 63, no. 5, pp. 570–573, May 1975.

[22] J. F. Wakerly, "Microcomputer reliability improvement using triple-modular redundancy," *IEEE Trans. on Computers*, vol. 64, no. 6, pp. 889–895, June 1976.

[23] X.-Y.Zhuo and S.-L. Li, "A new design method of voter in fault-tolerant redundancy multiple-module multi-microcomputer system," in *Digest of Papers, FTCS-13*, pp. 472–475, June 1983.

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_v	:	Time for voting on those variables changed during the previous voting interval.
$p_s^n(p_u^n)$	:	Probability of the n-th RSHW being successful (unsuccessful).
P	:	Probability of the first RSHW being successful.
R	:	Ratio of the probability of success at the $(n+1)$-th RSHW to that at the n-th RSHW.
k_m	:	Allowable maximum number of RSHWs.
X_f	:	Time of detecting a TMR/voting failure.
T_f^i	:	Time to a TMR system failure occurred first after starting the system in state S_i.
$F_{T_f^i}(X)$	:	Probability of a TMR failure from S_i during the execution time X ($f_{T_f^i} \equiv pdf$ of T_f^i).
t_i^j	:	Time of TMR failure occurrence via path j from S_i ($f_{t_f^i} \equiv pdf$ of t_f^i).
$S(x, y)$	:	State with x permanent faulty processor(s), y non-permanent faulty processor(s), and $(3 - x - y)$ nonfaulty processor(s) ($S_i = S(x, y)$ such that $i = 4x + y$).
J_m^i	:	Set of all paths to a fault state S_m from an initial state S_i ($E^i = \bigcup^m J_m^i$).
$\pi_i(0)$	:	Probability of a prior state before the first RSHW.
$\pi_m^i(T_f^i)$	:	Probability of a fault state S_m at time T_f^i from an initial state S_i.
$P_{mn}(T)$	:	Transition probability from S_m to S_n during T.
$C_1(k, X)$	:	Expected cost of RSHW with a nominal task-execution time X and MNRA k.
$C_1(X)(C_2(X))$	:	Expected cost of RSHW (RHWR) for X.
$F_{j_k j_{(k+1)}}$	:	Distribution of time to move to $S_{j_{(k+1)}}$ from S_{j_k}.
$\{E_{j_k}\}$	:	Set of all sub-paths emanating from S_{j_k}.
$\lambda_n(\lambda_p)$	:	Occurrence rate of non-permanent (permanent) faults.
$\frac{1}{\mu}$	:	Active duration of a non-permanent fault.

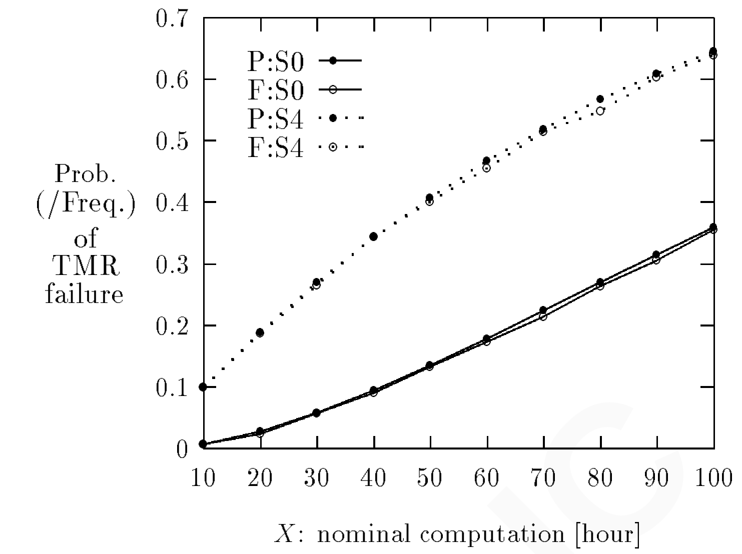

Figure 5: Probability/Frequency of a TMR failure obtained from the Markov-chain model (P:S0=from S_0 and P:S4=from S_4)/from simulations (F:S0=from S_0 and F:S4=from S_4).

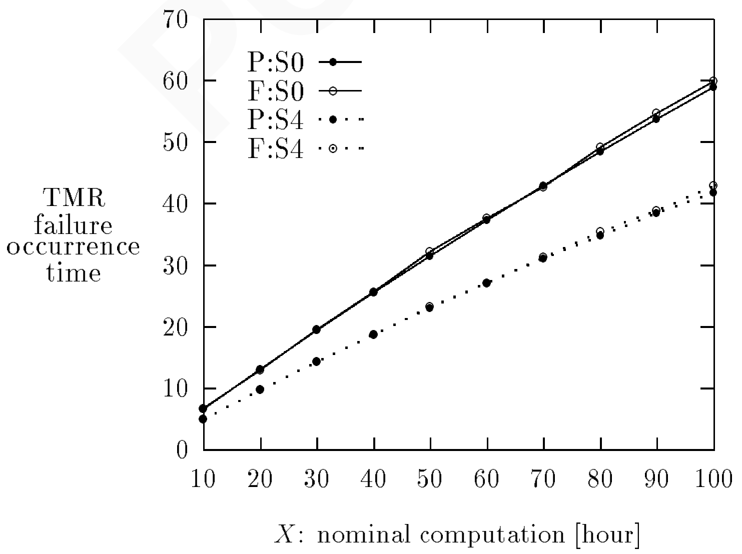

Figure 6: Mean TMR failure time ($E[T_f^0]$) obtained from analysis (P:S0=from S_0 and P:S4=from S_4), and from simulations (F:S0=from S_0 and F:S4=from S_4).

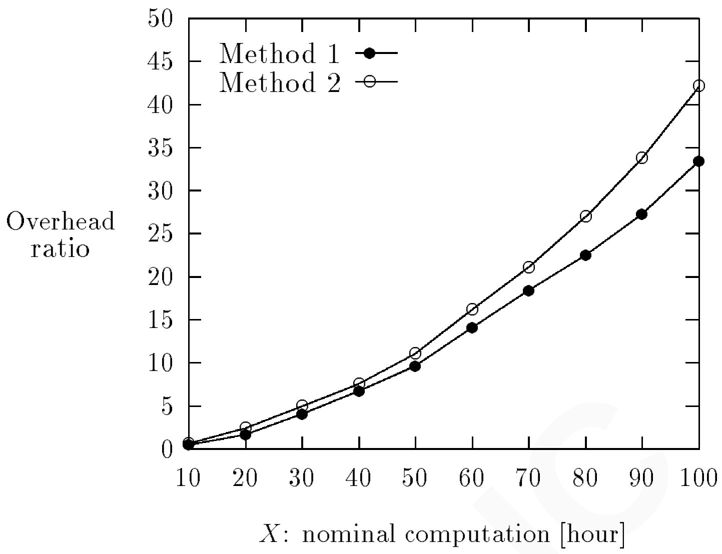

Figure 7: Overhead ratios [%] vs. X for RSHW and RHWR, with the optimal number of votings for $T_v = 0.0005$ hour: $(13, 34, 61, 87, 110, 133, 164, 181, 198, 216)$.

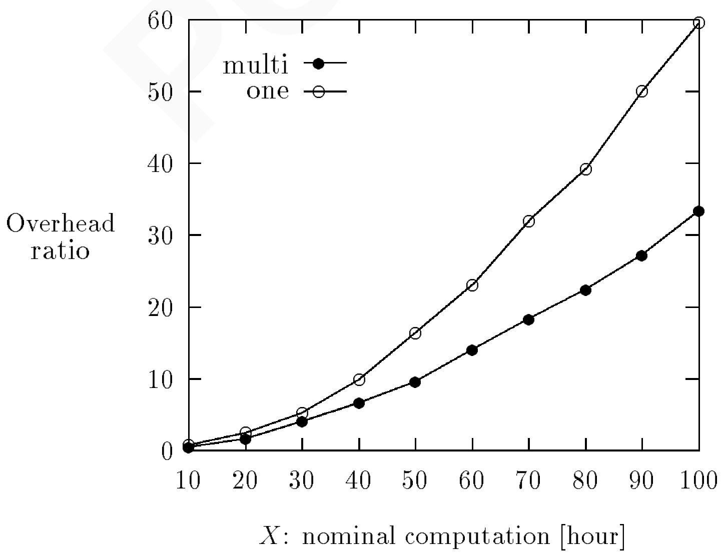

Figure 8: Overhead ratios [%] vs. X for one voting and multi-votings with the optimal number of votings.

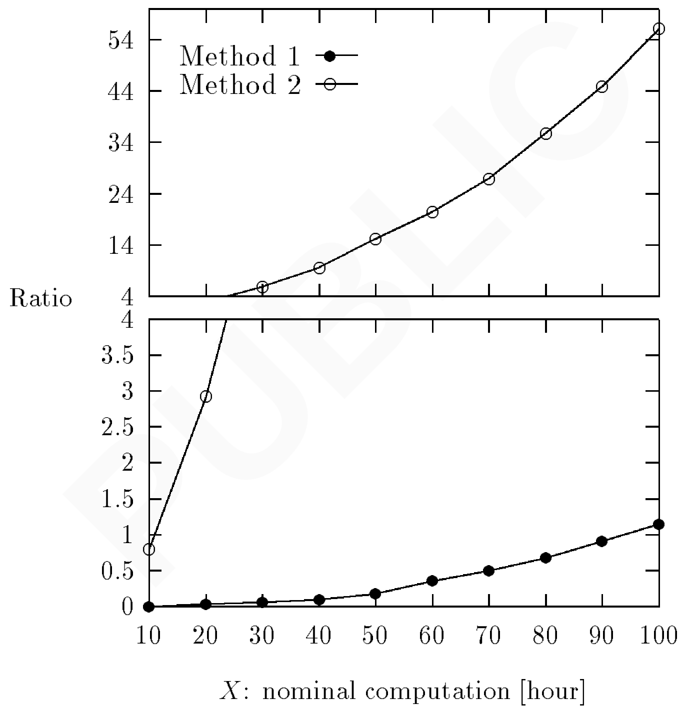

Figure 9: Ratio [%] of the number of reconfigurations to the total number of simulation runs.

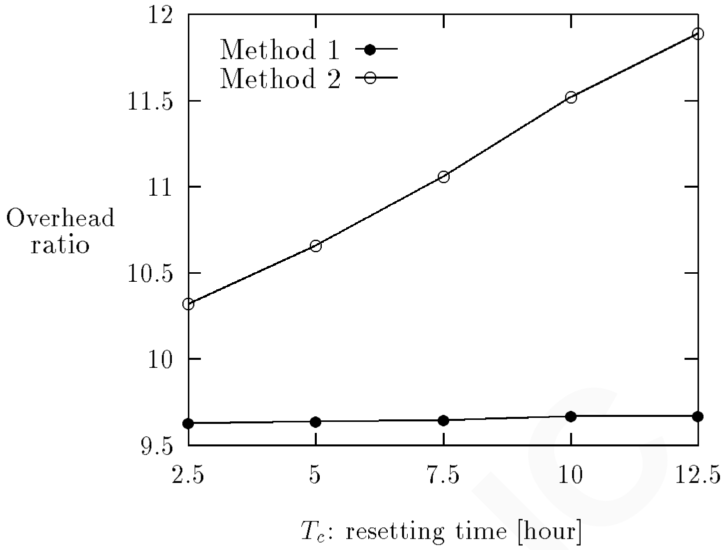

Figure 10: Overhead ratios [%] vs. T_c for RSHW and RHWR.

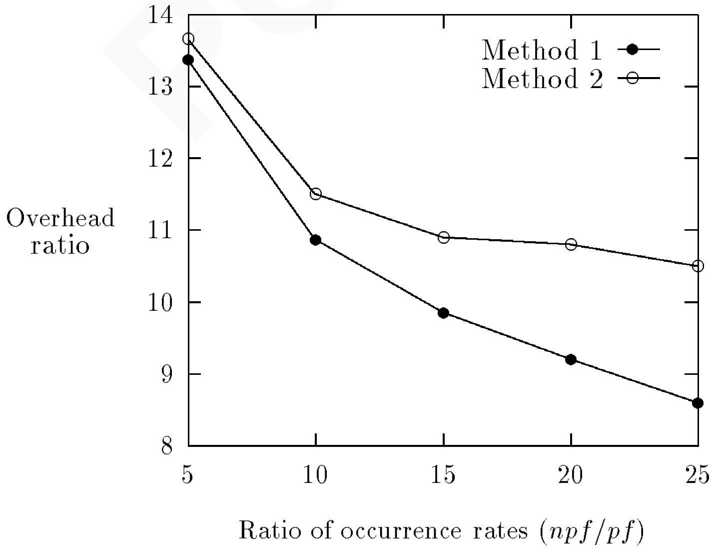

Figure 11: Overhead ratios [%] for different occurrence rates of non-permanent and permanent faults.