Design and Analysis of an Optimal Instruction-Retry Policy for TMR Controller Computers

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

Abstract—An instruction-retry policy is proposed to enhance the fault-tolerance of triple modular redundant (TMR) controller computers by adding time redundancy to them. A *TMR failure* is said to occur if a TMR system fails to establish a majority among its modules' outputs due to multiple faulty modules or a faulty voter. Either multiple consecutive TMR failures the active period of which exceeds a certain time limit or the exhaustion of spares as a result of frequent system reconfigurations may result in failure to meet the timing constraints of one or more tasks, called the *dynamic failure*, during a given mission. An optimal instruction-retry period is derived by minimizing the probability of dynamic failure upon detection of either a masked (by the TMR) error or a TMR failure. We also derive the minimum number of spares needed to keep below the pre-specified level the probability of dynamic failure for a given mission by using the derived optimal retry period.

Index Terms—Real-time control systems, controller computer, internal and external faults, common-cause faults, TMR failures and masked errors, retry, reconfiguration, dynamic failure, hard deadlines.

1 Introduction

A digital computer in the feedback loop of a real-time control system reads sensors and operator's input, and calculates control/output commands for actuators and display devices. The controller computer of a critical real-time control system such as a nuclear reactor and aircraft should be equipped with one or more fault-tolerance mechanisms to meet its stringent reliability requirement. The reliability of a controller computer depends on both the timeliness and correctness of its computation results. Thus, the timing constraint or the deadline imposed by the controlled process—called the *control system deadline*—is a key to the design of a controller computer, for which one must determine the type and degree of fault-tolerance.

Using the control system deadline[1] information obtained from the controlled process [13], we propose in this paper an optimal recovery policy to enhance system reliability by adding time redundancy to controller computers equipped with a minimum degree of spatial redundancy. Specifically, instruction retry is used "optimally" (in the sense of minimizing a certain cost) for triple modular redundant (TMR) controller computers.

A TMR system is a typical example of static redundancy which can tolerate one faulty module without any delay [3], [5], [6], [15], [16]. The TMR system can tolerate even multiple faults, if they occur sequentially with a relatively long inter-occurrence interval, by using appropriate detection, identification, and replacement of a faulty module (whose error was "masked") before a new fault occurs to another module within the TMR. Detect–diagnose–reconfigure is a conventional recovery policy for handling multiple faults in TMR systems [5], [15]. Alternatively, the system can also recover from the masked error induced by a transient fault by retrying the failed operation a fixed number of times on the same hardware [3]. Note that these two policies can tolerate only a subset of multiple faults. That is, *TMR failures*—failure to establish a majority of module outputs due to multiple faulty modules or a faulty voter—caused by coincident/common-cause faults require a different recovery method. Note that a harsh environment with electromagnetic interferences (EMI), such as lightning, high-intensity radiated fields (HIRF), or nuclear electromagnetic pulses (NEMP), may cause (near) coincident faults in all modules.

A TMR system uses a minimum degree of spatial redundancy to mask one faulty module (in general, $2n + 1$ modules needed to mask up to n faulty module outputs), and more than 90% of field failures are reported to be caused by transient faults [11]. Most TMR failures can thus be recovered by

1) using the capability of a TMR system that can mask one (permanent/transient) faulty module, and
2) retrying instructions on the same redundant hardware in case of a TMR failure resulting from additional transient fault(s).

This may reduce the hardware cost by avoiding the premature retirement of modules with transient faults, and reduce the time overhead of recovery and the probability of dynamic failure resulting from spares exhaustion as well as control system deadline misses. Note that system reconfiguration is more time-consuming than a simple retry [7]. Reconfiguration and (cold) restart generally consist of

1. The control system deadline is usually a *random* variable because of its dependence on the control system state which is a random variable.

• *H. Kim is with the Department of Electrical Engineering, Yonsei University, 134 Shinchon-Dong, Sudaemoon-Ku, Seoul 120-749, Korea.*
• *K.G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Sciences, the University of Michigan, Ann Arbor, MI 48109-2122. E-mail: kgshin@eecs.umich.edu.*

Manuscript received June 1994; revised Nov. 10, 1995.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96157.

1) switching power and bus connections,
2) running built-in-test (BIT) on the spare module,
3) loading programs and data,
4) initializing the software (even when warm spares are used, thus unneeding 1 and 2, this is still time-consuming), and
5) there are only a limited number of spares available during each mission.

As the simplest form of time redundancy, instruction retry has been proposed and analyzed by several researchers. The authors of [2] specified the retry period *a priori* in an ad hoc manner. The retry period was also derived by minimizing an average task-oriented measure (mean execution time per instruction) [8], or mean task-completion time by using a Bayesian decision approach [9] or the maximum likelihood principle [10], under the assumption that an infinite number of spares are available. The retry periods derived in these papers were intended for use in simplex systems.

In contrast to the above cited approaches, we derive in this paper the optimal retry period, r_{opt}, of a TMR controller computer by minimizing the probability of missing control system deadlines or dynamic failure upon detection of a masked error or a TMR failure. A retry will terminate if it becomes successful or the retry period expires, whichever occurs first. (Since the time required for repeated execution of an instruction cannot be cascaded into a single continuous duration, a retry period should be discrete, i.e., we define the "retry period" as a number of retry attempts throughout the discussion to follow.) If retry during a given period cannot recover the system from a TMR failure, the system will be reconfigured. Clearly, whether or not retry is successful depends upon the retry period as well as the *error latency* defined as the time interval from the occurrence of an error to its detection. The retry period should be large enough for the transient fault(s) inducing the detected error/failure to die away. Retry becomes unsuccessful when retry is used for permanent fault(s), and/or when the error latency is so large that more part of the program (in addition to the retried instruction) was contaminated. This may in turn increase the recovery time. We assume the use of a detection scheme with high (but not necessarily perfect) coverage for both masked errors and TMR failures as required by a usual retry policy. For example, any error in a module can be caught by using a simple disagreement detector [12], [17]. One can ultimately adopt a detection scheme like a Totally Self Checking Circuit (TSCC) in [4], which has the capability of detecting a masked error as well as a TMR failure and is both self-testing and fault-secure.

The probability of dynamic failure, P_{dyn}, depends on the mission lifetime, the number of spares, the control system deadline, and the retry period r.[2] We calculate P_{dyn} as a function of these parameters and derive r_{opt} by minimizing P_{dyn} in each case of masked error or TMR failure. Using the optimal retry period, we also determine the minimum number of spares needed to attain the largest acceptable P_{dyn} for a given mission.

This paper is organized as follows. Section 2 describes the characteristics of a real-time control system, the basic assumptions used, and the required property and structure of a detection scheme adopted. In Section 3, we numerically derive the optimal retry period by minimizing P_{dyn} in both cases of TMR failure and masked error. The effects of the number of available spares on P_{dyn} are also analyzed there. Section 4 presents representative numerical examples. The paper concludes with Section 5.

2. $r = 0$ means system reconfiguration without retry.

2 Notation, Assumptions, and Models

A real-time control system executes missions between maintenances, and usually no repair is assumed during a single mission. System diagnosis and the subsequent repair, if needed, are performed during a maintenance period between missions. As shown in Fig. 1, a mission lifetime generally consists of many *task periods* during each of which a sequence of instructions are executed to generate a control command or display output. During a mission, a *dynamic failure* is said to occur due to several consecutive TMR failures whose total period exceeds a limit, called the *control system deadline*—the maximum delay in the feedback loop the controlled process can tolerate without losing system stability or leaving its allowed state space [13]. This delay could be as long as the time of executing several consecutive tasks or task invocations. Using the control system deadline, one can derive the deadline of each task.

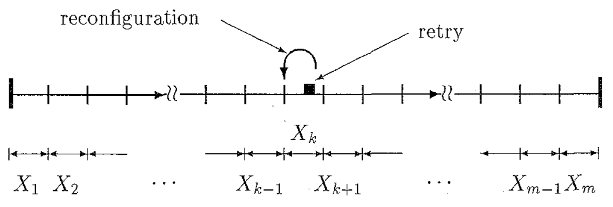


Fig. 1. Time index of a mission lifetime: $X_M = \sum_{i=1}^{m} X_i$.

The main computational load of a controller computer consists of a set of periodic tasks that are executed repetitively, each time with a different input. A dynamic failure may occur due to either missing the control system[3] or exhausting spares as a result of frequent system reconfigurations. Note that the latter may occur because there are only a limited number of spares aboard, depending on the weigh, volume, cost, MTTF of each spare, and the given mission. To study the effects of retry/reconfiguration on the probability of dynamic failure, we need to introduce the following variables:

- X_M: The lifetime of a mission consisting of m computational tasks.
- X_i: The execution time of the ith task of the mission, i.e., $X_M = \sum_{i=1}^{m} X_i$. $X = \frac{X_M}{m}$ is the average execution time of a task in the mission, or it is the execution time of a single task invocation if the mission consists of m invocations of a periodic task.

3. That is, missing the update of the control command for a period longer than the deadline due to consecutive TMR failures or generating incorrect execution results for several tasks.

- Δx: The inter-voting interval such that $X = K\Delta x$, where K is the number of times (intermediate) computation results are voted on during the average execution time, X, of a task or a task invocation.
- N: The number of spares available during a single mission.
- D: The control system deadline characterized by a probability density function $f_D(t)$.
- $r = \{r_e, r_t\}$: The maximum retry period allowed for a masked error (r_e) or a TMR failure (r_t).

Throughout the paper, we will classify faults to be *external* or *internal*, depending on whether their causes are inside or outside the system. Internal (system component) faults reside inside the system inducing errors, while external faults are caused by environmental interferences. One can observe that external faults are likely to be transient because adverse environmental conditions are generally temporary/transient and environmental disruptions result in functional error modes without actually damaging system components [1]. A harsh environment resulting from lightning, HIRF, or NEMP is likely to cause coincident or 'common mode' faults/errors. In other words, external faults are likely to result in multiple nonpermanent fault modules, which will in turn cause TMR failures.

We assume that all faults arrive according to time-invariant Poisson processes with rate λ_{ip} (λ_{in}) for permanent (nonpermanent) internal faults such that the total internal fault arrival rate $\lambda_i = \lambda_{ip} + \lambda_{in}$, and rate λ_{ep} (λ_{en}) for permanent (nonpermanent) external faults such that the total external fault arrival rate $\lambda_e = \lambda_{ep} + \lambda_{en}$. The active duration of a nonpermanent internal (external) fault is assumed to be exponentially distributed with mean $\frac{1}{\lambda_{in}}$ ($\frac{1}{\lambda_{en}}$). We also assume occurrences of *internal* faults in one module to be independent of those in other modules.

As mentioned earlier, retry could be effective only if the corresponding masked error/TMR failure is detected upon its occurrence. To achieve immediate and accurate detection of such errors/failures, we employ a special detection scheme like a Totally Self Checking Circuit (TSCC) [4], which detects the existence of a masked error and/or a TMR failure and is both self-testing and fault-secure. The TSCC provides two output indications distinguishing a masked error (and/or a fault in the error checking circuit) from a TMR failure (an output-information error generating a stop signal). However, it may not detect the incorrect output(s) before the completion of one instruction, thus making retry inapplicable. If a more complicated recovery operation such as rollback with checkpoints is applied to complement retry, an error/failure detected late can also be recovered by using only time redundancy, i.e., a nonzero error latency is allowed. However, in our approach to using retry or reconfiguration, any error/failure detected late is assumed to be recovered only through reconfiguration.

Let c_e and c_t be the detection coverages of a masked error and a TMR failure, respectively. Then, the probability that the detection scheme detects late (or misses) a masked error (or TMR failure) is $(1 - c_e)$ (or $(1 - c_t)$). There are two cases in which the detection scheme may fail to find a masked error/TMR failure:

1) a short-lived fault inducing the masked error/TMR failure disappears after contaminating one or more tasks,
2) a permanent or long-lived transient fault keeps generating incorrect outputs to lead to a dynamic failure.

Although an undetected permanent or long-lived transient fault can cause serious damages, the probability of not capturing such a long-lived fault is so small as to be ignored because consecutive instructions with incorrect results are likely to be detected before the control system deadline which is usually larger than the execution time of one or more tasks. Furthermore, the probability of missing a short-lived fault, which may not induce any error, is not negligible, but its effect is not significant to independent periodic tasks. Thus, we will only deal with late detection of errors/failures due to an imperfect detection scheme while assuming that long-active faults are detected eventually.

3 Optimal Retry Policy

In our model of a TMR system, the key variables in determining the optimal retry period r_{opt} are the mission lifetime $X_M = mX$, the number of spares N, and the control system deadline D.

When a masked error/TMR failure is detected, there are several state-transition scenarios as shown in Fig. 2. Suppose a masked error/TMR failure occurs during X_i. If retry is chosen to recover from this error/failure ($I_{r_e} = I_{r_t} = 1$), it may terminate when the fault that had caused the error/failure disappears (a successful retry), or the retry period expires (an unsuccessful retry), whichever occurs first. In the case of a TMR failure, a successful retry results when the system is in one of two possible states: *fault-free* state due to disappearance of all existing (nonpermanent) faults and *one masked-error* state due to the existence of one still-active faulty module. A successful retry for a masked error moves the controller computer to fault-free state. Unsuccessful retries may lead to a dynamic failure in case of a TMR failure,[4] or may trigger a system reconfiguration in both cases of masked error and TMR failure. Note that retry for a masked error is performed only on the faulty module while retaining the execution results from the other two healthy modules. The occurrence of a new fault in another module before the disappearance of the current fault or initiation of a system reconfiguration will result in a TMR failure. The system may be reconfigured immediately without retry ($I_{r_e} = I_{r_t} = 0$) or after an unsuccessful retry. If no spares are available, or $N = 0$ for a masked error and $N < 3$ for a TMR failure, a dynamic failure occurs. System reconfiguration increases the probability of exhausting spares during the remaining mission even if it could prevent an immediate dynamic failure. When the control system deadline is tight, system reconfiguration may also lead to a dynamic failure due to its setup and restart delays, during which another TMR failure may occur.

4. Mainly due to missing a control system deadline.

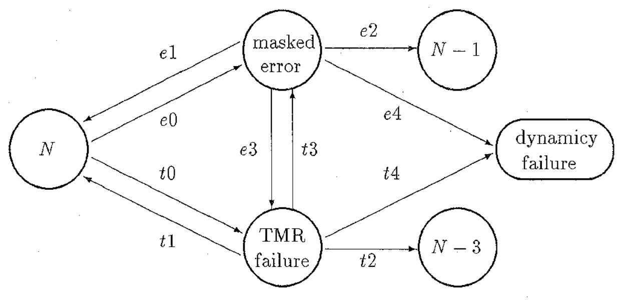


Fig. 2. A system state diagram. In case of a masked error (*e*0): *e*1 = recovery by retry, *e*2 = recovery by immediate reconfiguration ($I_{r_e} = 0$) or after unsuccessful retry ($I_{r_e} = 1$), *e*3 = a TMR failure due to occurrence of another faulty module during retry, *e*4 = spares exhaustion during reconfiguration or missing the control system deadline. In case of a TMR failure (*t*0): *t*1 = restoration to a fault-free state by retry, *t*2 = recovery by reconfiguration, *t*3 = restoration to one masked-error state by retry, *t*4 = the same as *e*4.

All of the above phenomena are captured in the Markov-chain of Fig. 3, each state of which is distinguished by the number of spares available. In this model, the probabilities p_{E_s} and p_{T_s} account for replacing the faulty module(s) that had caused a masked error and a TMR failure, respectively, and p_{mh} represents the probability of dynamic failure due to missing the control system deadline during the execution of a task. These probabilities determine the probability of dynamic failure over the mission lifetime. They all depend upon whether retry is used or not (represented by a pair of indicator functions (I_{r_e}, I_{r_t})) and how long retry lasts (i.e., values of (r_e, r_t)), or how many times the failed instruction is retried.

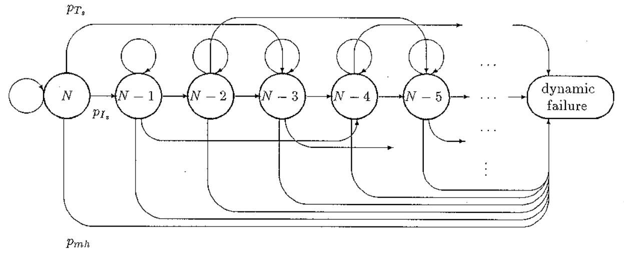


Fig. 3. A modified Markov-chain model based on the number of spares upon occurrence of masked errors or TMR failures: $p_0 = 1 - p_{E_s} - p_{T_s} - p_{mh}$, $p_{E_s}(p_{T_s})$ = probability of reconfiguration due to a masked error (TMR failure), p_{mh} = probability of dynamic failure due to missing the control system deadline.

$P_{dyn}(r, m, X, N, D)$ is obtained simply by adding the probability of exhausting spares $P_{es}(r, m, X, N)$ and the probability of missing the control system deadline $P_{mh}(r, m, X, D)$ during the mission lifetime $X_M = mX$, i.e.,

$$P_{dyn}(r, m, N, D, X) = P_{es}(r, m, N, X) + P_{mh}(r, m, D, X). \quad (3.1)$$

N spares can withstand j and k reconfigurations resulting from TMR failures and masked errors during m invocations of tasks, respectively, such that $3j + k \le N$. Thus,

$$P_{es}(r, m, X, N) =$$
$$\sum_{3j+k>N} \frac{m!}{j!k!(m-j-k)!} p_{T_s}^j(r, X) p_{E_s}^k(r, X)\left[1 - p_{T_s}(r, X) - p_{e_s}(r, X)\right]^{m-j-k}, \quad (3.2)$$

where $p_{E_s}(r, X)$ ($p_{T_s}(r, X)$) is the probability of reconfiguration due to a masked error (TMR failure) during the average task-execution time X with a retry period $r = \{r_e, r_t\}$. Suppose that in (3.2) of a multinomial function, $m \gg 1$, $p_{T_s} + p_{E_s} \ll 1$, but mp_{T_s} and mp_{E_s} remain constant, say $mp_{T_s} = a_1$ and $mp_{E_s} = a_2$. Then, (3.2) can be approximated as $\frac{1}{j!k!} a_1^j a_2^k (1 - a_1 - a_2)^{m-j-k}$, where $m(m-1) \cdots (m-j-k+1) \simeq m^{k+j}$, if m is allowed to become large enough and if j and k are fixed. Hence, as $m \to \infty$, $p_{T_s} + p_{E_s} \to 0$, and $j + k \ll m$, we obtain

$$\frac{1}{j!k!} a_1^j a_2^k (1 - a_1 - a_2)^{m-j-k} \to \frac{1}{j!k!} a_1^j a_2^k e^{-a_1 - a_2}.$$

Thus, in situations where the multinomial law applies with $m \gg 1$, $p_{T_s} + p_{E_s} \ll 1$, but $mp_{T_s} = a_1$ and $mp_{E_s} = a_2$ are finite constants, we can use the following approximation for (3.2):

$$\frac{1}{j!k!} \left(mp_{T_s}(r_t, X)\right)^j \left(mp_{E_s}(r_i, X)\right)^k e^{-m\left[p_{T_s}(r_t, X) + p_{E_s}(r_i, X)\right]}.$$

Equation (3.2) is based on the assumption of at most one masked error or TMR failure during X, which is reasonable because masked errors and TMR failures occur very rarely. $P_{mh}(r, m, D, X)$ is also derived from the probability, $p_{mh}(r, X, D)$, of missing the control system deadline D during X when retry is applied for a period r. For the successful completion of a mission, every task/invocation must not miss its deadline, which is represented by $\prod_{i=1}^{m} [1 - p_{mh}(r, X, D)]$. Thus, $P_{mh}(r, m, X, D)$ is computed as:

$$P_{mh}(r, m, X, D) = 1 - \prod_{i=1}^{m} \left[1 - p_{mh}(r, X, D)\right] = 1 - \left[1 - p_{mh}(r, X, D)\right]^m,$$
$$\simeq mp_{mh}(r, X, D), \quad \text{if } p_{mh} \ll 1. \quad (3.3)$$

We now analyze quantitatively the effects of retries for masked errors and TMR failures on P_{dyn} separately, and then combine the two results to derive r_{opt}.

3.1 Reconfiguration without Retry ($I_{r_e} = I_{r_t} = 0$)

This is the usual recovery method for masked errors and TMR failures without using time redundancy, i.e., $r_e = r_t = 0$. The diagnosed faulty module is replaced with a healthy spare. In case of a TMR failure, all three modules are switched out due to the considerable time overhead of identifying (diagnosing) faulty modules, and the current task is re-executed with the reloaded data as shown in Fig. 1.

$p_{E_s}(X)$ and $p_{T_s}(X)$ in this case are equal to the probabilities of a masked error $p_E(X)$ and a TMR failure $p_T(X)$, respectively. Since masked errors are also recovered through reconfiguration (or retry in the following methods) with high detection coverage c_e, a TMR failure is assumed to occur due mainly to (near) coincident faults occurring within an inter-voting interval, rather than sequentially occurring faults. Let $p_{T_e}(X)$ and $p_{T_i}(X)$ be the probabilities of TMR failures caused by external and internal faults, respectively. Then, $p_{T_e}(X)$ is equal to $1 - e^{-\lambda_e X}$, which is the probability of occurrence of an external fault during X, because external faults are assumed to cause coincident internal faults/errors. $p_{T_i}(X)$ is also obtained by using the probability of coincident internal faults in two or three modules in K inter-voting intervals during X, that is:

$$p_{T_i}(X) = 1 - \left[1 - 3\left(1 - e^{-\lambda_i \Delta x}\right)^2 + 2\left(1 - e^{-\lambda_i \Delta x}\right)^3\right]^K$$
$$\simeq K\left[3\left(1 - e^{-\lambda_i \Delta x}\right)^2 - 2\left(1 - e^{-\lambda_i \Delta x}\right)^3\right].$$

Since TMR failures caused by external and internal faults are not exclusive to each other, $p_T(X)$ is not the direct sum of $p_{T_e}(X)$ and $p_{T_i}(X)$, but equals $p_{T_e}(X) + p_{T_i}(X) - p_{T_e}(X)p_{T_i}(X)$. $p_E(X)$ is the probability of occurrence of an internal fault in only one module during X, which is calculated as:

$$p_E(X) = 3\left(1 - e^{-\lambda_i X}\right) - 3\left(1 - e^{-\lambda_i X}\right)^2 + \left(1 - e^{-\lambda_i X}\right)^3 - p_{T_i}(X)$$
$$\simeq 3K\left(1 - e^{-\lambda_i \Delta x}\right)e^{-2\lambda_i \Delta x}.$$

Let X_a be the actual time required to complete the computation corresponding to X,[5] and f_{X_a} and f_D are the *pdfs* of X_a and the control system deadline D, respectively. Using f_{X_a} and f_D, we can obtain $p_{mh}(X, D)$ as the probability that $X_a > D$:

$$p_{mh}(X, D) = \int_0^\infty \int_D^\infty f_{X_a}(x) F_D(y)\, dx\, dy.$$

The set of samples of X_a is obtained as:

$$X_a \in \left\{X, \left(\overline{X} + t_r\right) + X,\ 2\left(\overline{X} + t_r\right) + X,\ 3\left(\overline{X} + t_r\right) + X,\ \cdots\right\},$$

where t_r is the resetting time and $\overline{X}$ is the mean occurrence time of a TMR failure, which was derived in [14]. Since X_a has discrete values, the probability mass function (*pmf*) of X_a is:

$$f_{X_a}^k = Prob\left[X_a = k\left(\overline{X} + t_r\right) + X\right] = p_T^k(X)\left(1 - p_T(X)\right).$$

Note that f_D is given a priori from experimental data or the analysis of controlled processes [13]. Consequently,

$$p_{mh}(X, D) = \int_0^\infty \sum_{k > \left\lfloor (D - X)/\left(\overline{X} + t_r\right)\right\rfloor} f_{X_a}^k(x) f_D(y)\, dy.$$

3.2 Retry for Masked Errors ($I_{r_e} = 1, I_{r_t} = 0$)

In this case, retry of a period r_e is initiated upon detection of a masked error, and reconfiguration is the only recovery mechanism for TMR failures. Thus, $p_{E_s}(X)$ and $p_{T_s}(X)$ are no longer equal to $p_E(X)$ and $p_T(X)$. p_{E_s} decreases with r_e, because most masked errors are induced by nonpermanent faults and because a simple retry is likely to recover from them (only if they are detected before the completion of an instruction that is to be retried). However, p_{T_s} increases with r_e due to the increased probability of a TMR failure during the retry period, i.e., a TMR failure may be induced by faults occurring sequentially during the retry for a masked error as well as by coincident faults.

Let $R_e^{dec}(r_e)$ and $R_t^{inc}(r_e)$ be the coefficients indicating, respectively, the decrease of masked errors and the increase of TMR failures after retrying for masked errors. Then,

$$R_e^{dec}(r_e) = \left(\frac{\lambda_{ip}}{\lambda_i} + \frac{\lambda_{in}}{\lambda_i} e^{-\lambda_{ia} r_e}\right) e^{-(2\lambda_i + \lambda_e) r_e}$$
$$+ \frac{2\lambda_{in}}{\lambda_i}\left(1 - e^{-\lambda_{ia} r_e}\right)\left(1 - e^{-\lambda_i r_e}\right) e^{-(\lambda_i + \lambda_e) r_e}$$

where the first term represents the effect of an unsuccessful retry and the second term represents a second fault occurrence after successful retry on the first fault occurrence.

$$R_t^{inc}(r_e) = \left(\frac{\lambda_{ip}}{\lambda_i} + \frac{\lambda_{in}}{\lambda_i} e^{-\lambda_{ia} r_e}\right)\left(1 - e^{-(2\lambda_i + \lambda_e) r_e}\right)$$
$$+ \frac{\lambda_{in}}{\lambda_i}\Big(\left(1 - e^{-\lambda_{ia} r_e}\right)\left(1 - e^{-\lambda_e r_e}\right.$$
$$\left. + \left(1 - e^{-\lambda_i r_e}\right)^2 - \left(1 - e^{-\lambda_e r_e}\right)\left(1 - e^{-\lambda_i r_e}\right)^2\Big),$$

where the first term represents an unsuccessful retry (occurrences of fault in any healthy module) and the second term represents a successful retry (occurrences of fault in two or three modules). Then, using $R_e^{dec}(r_e)$, $R_t^{inc}(r_e)$, c_e, $p_E(X)$, and $p_T(X)$, we can obtain $p_{E_s}(r_e, X)$ and $p_{T_s}(r_e, X)$ as:

$$p_{E_s}(r_e, X) = \left(1 - c_e\right)p_E(X) + c_e R_e^{dec}(r_e) P_E(X),$$
$$p_{T_s}(r_e, X) = p_T(X) + c_e R_t^{inc}(r_e) p_E(X). \tag{3.4}$$

$p_{mh}(r_e, X, D)$ is derived in the same way as before except for the change of $f_{X_a}^k$, i.e., substituting $p_{T_s}(r_e, X)$ for $p_T(X)$ since the probability of a TMR failure is changed:

$$f_{X_a}^k = Prob\left[X_a = k\left(\overline{X} + t_r\right) + X\right]$$
$$= p_{T_s}^k(r_e, X)\left(1 - p_{T_s}(r_e, X)\right), \quad 0 \le k \le \infty.$$

3.3 Retry for TMR Failures ($I_{r_e} = 0, I_{r_t} = 1$)

This is the opposite to the previous policy, that is, retry of a period r_t is initiated upon detection of a TMR failure, and a masked error calls for an immediate reconfiguration. Obviously, p_{T_s} decreases with r_t, because most TMR failures are also recovered by a simple retry during which non-permanent faults are likely to disappear. Even if retry is successful, there may still exist a faulty module. Those masked-error states transit-ed from TMR failures during a retry increase $p_{E_s}(X)$.

5. Because of the time overhead of retry and/or reconfiguration for errors during the execution of a mission segment, X_a is usually larger than X.

Let p_{ij} denote the conditional probability of i faulty modules and j permanent-fault modules given a TMR failure (i.e., $i \le 2$), which is computed as:

$$p_{20} = \left(\frac{\lambda_{in}}{\lambda_i}\right)^2 \frac{B}{A+B+C},\ p_{21} = \frac{2\lambda_{in}\lambda_{ip}}{\lambda_i^2}\frac{B}{A+B+C},$$

$$p_{22} = \left(\frac{\lambda_{ip}}{\lambda_i}\right)^2 \frac{B}{A+B+C},$$

$$p_{30} = \frac{\lambda_{en}}{\lambda_e}\frac{A}{A+B+C} + \left(\frac{\lambda_{in}}{\lambda_i}\right)^3 \frac{C}{A+B+C},$$

$$p_{31} = \frac{3\lambda_{in}^2\lambda_{ip}}{\lambda_i^3}\frac{C}{A+B+C},\ p_{32} = \frac{3\lambda_{in}\lambda_{ip}^2}{\lambda_i^3}\frac{C}{A+B+C},$$

$$p_{33} = \frac{\lambda_{ep}}{\lambda_e}\frac{A}{A+B+C} + \left(\frac{\lambda_{ip}}{\lambda_i}\right)^3 \frac{C}{A+B+C},$$

where $A = 1 - e^{-\lambda_e X}$ and

$$B = 3K(1-e^{-\lambda_i \Delta x})^2 e^{-\lambda_i \Delta x}\ (C = K(1-e^{-\lambda_i \Delta x})^3)$$

are the probabilities of occurrences of external faults and near-coincident internal faults inducing two (three) faulty modules during X. Let $R_e^{inc}(r_t)$ and $R_t^{dec}(r_t)$ be the coefficients indicating, respectively, the increase of masked errors and the decrease of TMR failures after retrying for TMR failures. $R_e^{inc}(r_t)$ is obtained by computing the probability that only one faulty module remains in each case of p_{ij}, and $R_t^{dec}(r_t)$ is derived from the cases of more than two permanent or long-lived transient faults, thus:

$$R_e^{inc}(r_t) = p_{21}\left(1-e^{-\lambda_{ia}r_t}\right) + p_{31}\left(1-e^{-\lambda_{ia}r_t}\right)^2$$
$$+\ 2p_{20}\left(1-e^{-\lambda_{ia}r_t}\right)e^{-\lambda_{ia}r_t} + 3p_{30}C_i\left(1-e^{-\lambda_{ia}r_t}\right)^2 e^{-\lambda_{ia}r_t},$$

$$R_t^{dec}(r_t) = p_{33} + p_{32} + p_{22} + p_{21}e^{-\lambda_{ia}r_t} + p_{31}\left(2e^{-\lambda_{ia}r_t} - e^{-2\lambda_{ia}r_t}\right)$$
$$+\ p_{20}e^{-2\lambda_{ia}r_t} + p_{30}\left\{C_e e^{-\lambda_{ea}} + C_i\left(3e^{-2\lambda_{ia}r_t} - 2e^{-3\lambda_{ia}r_t}\right)\right\},$$

where

$$C_e = \frac{\lambda_{en}}{\lambda_e}A\left(\frac{\lambda_{en}}{\lambda_e}A + \left(\frac{\lambda_{in}}{\lambda_i}\right)^3 C\right)^{-1},$$

$$C_i = \left(\frac{\lambda_{in}}{\lambda_i}\right)^3 C\left(\frac{\lambda_{en}}{\lambda_e}A + \left(\frac{\lambda_{in}}{\lambda_i}\right)^3 C\right)^{-1}.$$

Using these coefficients, we obtain $p_{E_s}(r_t, X)$ and $p_{T_s}(r_t, X)$ as:

$$p_{E_s}(r_t, X) = p_E(X) + c_t R_e^{inc}(r_t)p_T(X),$$
$$p_{T_s}(r_t, X) = (1-c_t)p_T(X) + c_t R_t^{dec}(r_t)p_T(X). \quad (3.5)$$

Now, a dynamic failure may occur due mainly to unsuccessful retries if the control system deadline, D, is tight. The controller computer may fail to generate a correct control command within D units of time due to TMR failures after repeating/retrying the execution of an instruction. Thus, the derivation of $p_{mh}(r_t, X, D)$ is different from that of the previous two cases. By approximating the mean end-of-retry period with $\frac{1}{\lambda_{ia}}$ in case of a successful retry, the samples of X_a are:

$$X_a \in \left\{X, X + \frac{1}{\lambda_{ia}}, (\overline{X} + t_r + r_t) + X, (\overline{X} + t_r + r_t) + X + \frac{1}{\lambda_{ia}},\right.$$
$$\left. 2(\overline{X} + t_r + r_t) + X, \cdots\right\}.$$

The *pmf* of X_a is then:

$$f_{X_a}^k = \Pr\left[X_a = k(\overline{X} + t_r + r_t) + X + \delta\frac{1}{\lambda_i a}\right],\ 0 \le k \le \infty,\ \delta \in \{0, 1\}$$
$$= p_T^{k+\delta}(X)(1 - p_s(r_t))^k (1 - p_T(X))^{1-\delta} p_s(r_t)^\delta, \quad (3.6)$$

where $\delta \in \{0, 1\}$ indicates that the mission segment corresponding to X is completed because of a successful retry upon detection of a TMR failure or because of no TMR failure, while repeating the execution of the mission segment k times with k reconfigurations. $p_s(r_t)$ of (3.6) represents the probability of successful retry, which is computed by considering all cases of no more than one faulty module remaining after retrying in each case of p_{ij}:

$$p_s(r_t) = c_t\left[p_{21}\left(1-e^{-\lambda_{ia}r_t}\right) + p_{31}\left(1-e^{-\lambda_{ia}r_t}\right)^2 + p_{20}\left(1-e^{-2\lambda_{ia}r_t}\right)\right.$$
$$\left. +\ p_{30}\left\{C_e\left(1-e^{-\lambda_{ea}r_t}\right) + C_i\left(1-e^{-\lambda_{ia}r_t}\right)^2\left(1+2e^{-\lambda_{ia}r_t}\right)\right\}\right].$$

Consequently,

$$p_{mh}(r_t, X, D) = \int_0^\infty \sum_{k > \left\lfloor \left(D - X - \delta\frac{1}{\lambda_i a}\right)/\left(\overline{X} + t_r + r_t\right)\right\rfloor} f_{X_a}^k(x) f_D(y)\,dy.$$

3.4 Retry for Both Cases ($I_{r_e} = 1$, $I_{r_t} = 1$)

Finally, we present a retry policy for both cases of TMR failure and masked error with retry periods r_t and r_e, respectively. To show a significant decrease in the frequency of reconfigurations, $p_{E_s}(r, X)$ and $p_{T_s}(r, X)$ are also derived by using all the coefficients indicating the increase and/or decrease of masked errors and TMR failures, i.e., in the same way of deriving (3.4) and (3.5):

$$p_{E_s}(r, X) = \left\{p_E(X) + c_t R_e^{inc}(r_t)p_T(X)\right\}\left\{1 - c_e + c_e R_e^{dec}(r_e)\right\},$$

$$p_{T_s}(r, X) = \left\{p_T(X) + c_e(1-c_t)R_t^{inc}(r_e)P_E(X)\right\}\left\{1 - c_t + c_t R_t^{dec}(r_t)\right\}, \quad (3.7)$$

which are obtained by considering both the effects of retry for masked errors ($R_e^{dec}(r_e)$ and $R_t^{inc}(r_e)$) and the effects of retry for TMR failures ($R_e^{inc}(r_t)$ and $R_t^{dec}(r_t)$). The derivation of $p_{mh}(r, X, D)$ is similar to the previous case because both cases use retry for a TMR failure. The only difference from (3.6) is the change of $p_T(X)$ to p_{T0} in $f_{X_a}^k$, where p_{T0} is the increased probability of TMR failures after retrying for masked errors:

$$f_{X_a}^k = p_{T0}^{k+\delta}(X)(1 - p_s(r_t))^k (1 - p_{T0}(X))^{1-\delta} p_s(r_t)^\delta, \quad (3.8)$$

where

$$p_{T0} = p_T(X) + c_e R_e^{inc}(r_e)p_E(X).$$

3.5 Optimal Retry Period and Minimum Number of Spares

Using the derived $p_{E_s}(r_e, X)$, $p_{T_s}(r_t, X)$, and $p_{mh}(r, X, D)$, one can compute P_{es} and P_{mh} from (3.2) and (3.3), which are in turn used to calculate P_{dyn} with (3.1). Now, $r_{opt} = \{r_{eopt}, r_{topt}\}$ is determined by minimizing the derived P_{dyn} with respect to r_e and r_t. (There always exist r_{eopt} and r_{topt} that minimize P_{dyn} over a closed interval, $0 \le r_e, r_t \le D - X$.) The derivation of r_{opt} involves the following three steps:

Step 1: Compute r_{eopt} from the case of $I_{r_e} = 1$ and $I_{r_t} = 0$ and let it be r_e^*.

Step 2: Compute r_{topt} from the case of $I_{r_e} = 1$ and $I_{r_t} = 1$ by using r_e^* and let it be r_t^*.

Step 3: Calculate $P_{dyn}(r = 0)$ and compare it with $P_{dyn}(r_e^*, r_t^*)$. If $P_{dyn}(r = 0) \le P_{dyn}(r_e^*, r_t^*)$, then $r_{opt} = 0$ else $r_{opt} = \{r_e^*, r_t^*\}$.

Steps 1 and 2 to minimize P_{dyn} with respect to r_e and r_t separately are reasonable because the frequency (thus, effects on P_{dyn}) of masked errors is significantly larger than that of TMR failures. Step 3 is taken to choose a better recovery policy between reconfiguration and retry (with the derived retry period).

When the maximum acceptable probability of dynamic failure is given as p_{dyn}^{max}, the minimum number of spares, N_{min}, can be computed by using the derived optimal retry period. Obviously, P_{dyn} decreases with N when other variables are fixed. We can derive the relation between P_{dyn} and N iteratively by increasing N from a certain initial value N_0. From this relation N_{min} is determined using p_{dyn}^{max}. The derivation of N_{min} is described in pseudo-code, where N_0 is simply determined by using the case of $I_{r_e} = 0$ and $I_{r_t} = 0$:

```
\*Recursive method to derive $N_{min}$*\
\* $\Delta$ = acceptable error in the upper bound of $p_{dyn}^{max}$*\
\* PROB_DYN_1(): program to compute $P_{dyn}$ for $I_{r_e} = 0$,
      $I_{r_t} = 0$.*\
\* PROB_DYN_2(): program to compute $P_{dyn}$ using $r_{opt}$ for
      $I_{r_e} = 1$, $I_{r_t} = 1$.*\
$N_0$ : =K        \* K: arbitrary number *\
$P_{dyn}$ := PROB_DYN_1($N_0$)
while ($p_{dyn}^{max} - \Delta \le P_{dyn} \le p_{dyn}^{max}$)
  if ($P_{dyn} \ge p_{dyn}^{max}$) then
    $N_0 := N_0 + 1$
  else $N_0 := N_0 - 1$
end_while
$P_{dyn}$ := PROB_DYN_2($N_0$)
while ($p_{dyn}^{max} \le P_{dyn} \le p_{dyn}^{max} + \Delta$)
  if ($P_{dyn} \le p_{dyn}^{max}$) then
    $N := N - 1$
  else $N := N + 1$
end_while
return $N_{min} := N$
```

4 Numerical Examples

In this section, we present numerical examples of r_{opt} and N_{min} under a certain condition of fault occurrences. A brief performance analysis of several retry policies is also presented using numerical values of P_{dyn}. All variables have the same time unit and thus are listed without any specific unit. Specifically, the basic (time) unit is defined as a task period ($X = 1$) for convenience, i.e., the mission lifetime is simply represented by m. The control system deadline is specified by a distribution function $F_D(d) = \frac{d-2}{2}$ (uniformly distributed) for $2 \le d \le 4$, the result of which can be extended to any other model of the control system deadline. In the numerical results below, we used the resetting time $t_r = 0.1$ and the number of spares $N = 10$. (These numbers are chosen somewhat arbitrarily, but their choice would not change the conclusion we draw.) Fault occurrences are also governed by the following set of fault parameters:

$$\lambda_{ep} = 5 \times 10^{-8},\ \lambda_{en} = 10^{-6},\ \lambda_{ip} = 2 \times 10^{-7};$$

$$\lambda_{in} = 10^{-6},\ \frac{1}{\lambda_{ia}} = 0.01,\ \frac{1}{\lambda_{ea}} = 0.005. \tag{4.1}$$

The examples of P_{dyn} of the four retry policies in Section 3 are shown in Figs. 4, 5, and 6, while varying the mission lifetime m. A simple-minded retry period ($r_i = r_t = 0.02$) is used to demonstrate the effects of retrying for TMR failures and masked errors. Retrying for TMR failures significantly reduces P_{dyn} over the whole range of mission lifetime when the detection coverages are perfect ($c_e, c_t = 1$). When the mission is short, retrying for masked errors may increase P_{dyn}; this is demonstrated by comparing P_{dyn}s of $(I_{r_i} : I_{r_t}) = (0 : 0)$ and $(1 : 0)$, or $(0 : 1)$ and $(1 : 1)$, when $m = 1{,}000$ or $10{,}000$. In case of short missions, a dynamic failure is likely to occur due to missing a control system deadline, rather than exhausting spares. Retrying for masked errors clearly increases the probability of TMR failures (thus the probability of missing a control system deadline), and decreasing the probability of exhausting spares is not effective in reducing P_{dyn} for short missions. However, retrying for masked errors also decreases P_{dyn} as the mission lifetime increases.

When the detection coverages are not perfect ($c_e, c_t < 1$) the relative advantage of retry gets diminished; one can see this by comparing Fig. 4 with Fig. 5 or Fig. 6. Since the retry periods are generally by far smaller than a control system deadline, the retry policy retains its superiority to reconfiguration even if the detection coverages are not perfect, i.e., the time spent on an unsuccessful retry does not significantly affect the probability of missing control system deadlines.

TABLE 1
OPTIMAL RETRY PERIODS $r_{opt} = \{r_i, r_t\}$ WHEN $X = 1$:
I) $K = 5{,}000$ INSTRUCTIONS/TASK, II) $m = 10{,}000$ TASKS/MISSION

(i) m ($\times 10^3$)	1	10	25	50	100	250
r_i	0.1002	0.1248	0.1324	0.1402	0.1448	0.1524
r_t	0.0562	0.0564	0.0560	0.0546	0.0542	0.0762
(ii) K ($\times 10^3$)	2.5	5	7.5	10	15	25
r_i	0.2392	0.1248	0.0844	0.0767	0.0462	0.0279
r_t	0.0456	0.0564	0.0443	0.0440	0.0164	0.0177

TABLE 2
NUMERICAL EXAMPLES OF N_{min} VS. m

m ($\times 10^3$)	1	10	50	100	150	250	300	400
N_{min}	6	7	9	10	12	13	14	15

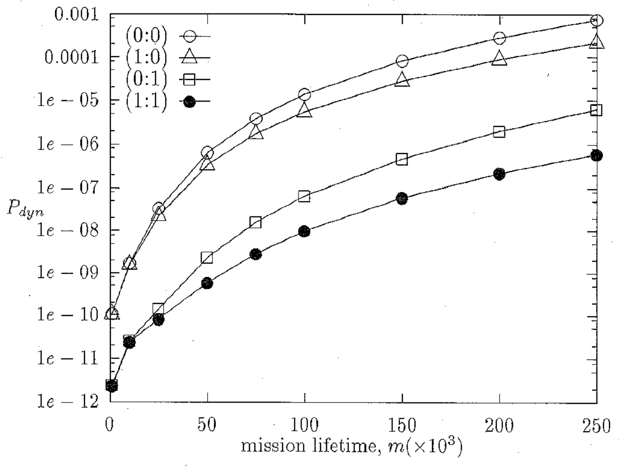


Fig. 4. Probabilities of dynamic failure (P_{dyn}s) of several retry policies (I_{r_e} : I_{r_t}): $c_e = c_t = 1$.

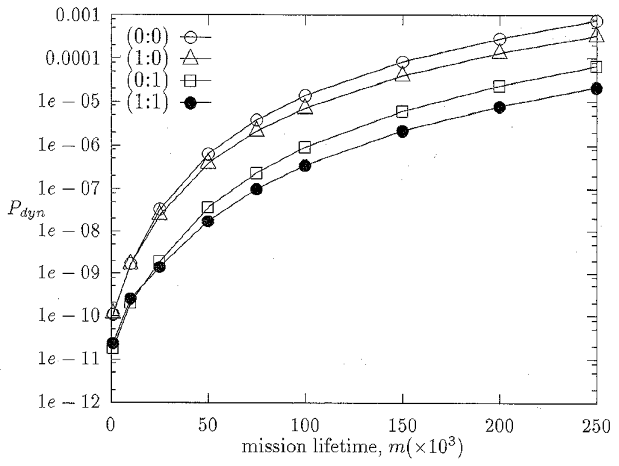


Fig. 6. P_{dyn}s of several retry policies (I_{r_e} : I_{r_t}): $c_e = c_t = 0.7$.

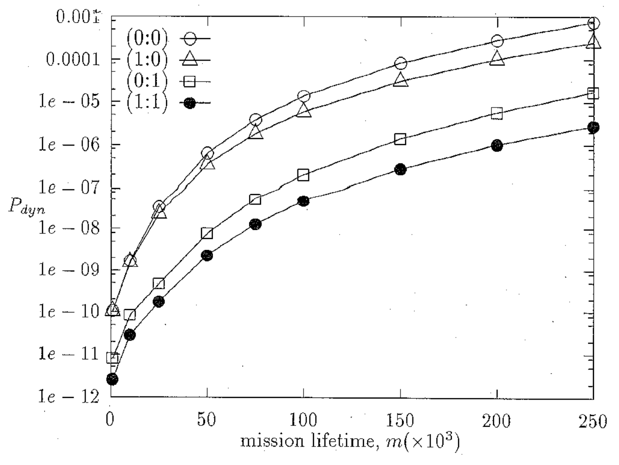


Fig. 5. P_{dyn}s of several retry policies (I_{r_e} : I_{r_t}): $c_e = c_t = 0.9$.

Several examples of r_{opt} are shown in Table 1 while varying the mission lifetime m and the task/invocation period K. The invocation period is measured by the number of voting times for intermediate computation results (K) and the inter-voting interval (Δx), as defined in Section 2. Those values are derived by the method proposed in Section 3, i.e., r_i is derived first and r_t computed by using the derived r_i. Case (i) uses the same task period ($K = 5{,}000$) which is equal to one time unit (i.e., X = 1), while Case (ii) takes the same mission lifetime ($m = 10{,}000$). Both cases use the same values of fault parameters, the number of spares, the resetting time, and the control system deadlines as given earlier. Although the absolute values of r_{opt} ($r_{opt} \times K$) change a little with m and K, the relative values of r_{opt} with respect to a task period (X) changes significantly with K. One can observe that the mission lifetime does not affect the optimal retry period as much as the task period under the same condition.

N_{min} is given in Table 2 for various mission lifetimes with a required level of $P_{dyn} = 10^{-9}$/mission. Although other parameters affect the values of N_{min}, N_{min} heavily depends upon m. P_{dyn} decreases with an increase of N, but there is a lower bound of P_{dyn} because the minimum value of the probability of missing a control system deadline cannot be decreased beyond a certain value only by increasing N.

5 Conclusion

In this paper, we have analyzed the effects of retry policies for TMR failures and masked errors on the probability of dynamic failure. We have also proposed a method to derive the optimal retry periods for both TMR failures and masked errors by minimizing the probability of dynamic failure. For this purpose, we adopted a detection scheme with high coverages of TMR failures as well as masked errors. The proposed instruction-retry policy outperforms reconfiguration even with low detection coverage, as shown in Fig. 6.

Although the TMR structure can mask only one manifested faulty module, the occurrence of fault(s) in another module or coincident faults in multiple modules can lead to a TMR failure. Our retry policy reduces effectively the frequency of TMR failures by recovering from masked erroneous module(s), and it also recovers from a TMR failure as a result of coincident faults by considering the control system deadlines and the number of spares. Our analysis also includes the relation between the number of spares and the probability of dynamic failure, which is a key factor in determining the minimum number of spares so as to satisfy the required level of the probability of dynamic failure at minimum cost for a given mission.

Acknowledgments

The work reported was supported in part by the U.S. Office of Naval Research under Grant N00014-91-J-1115 and by NASA under Grant NAG-1-1120. Any opinions, finding, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the funding agencies.

References

[1] C.M. Belcastro, "Laboratory Test Methodology for Evaluating the Effects of Electromagnetic Disturbances on Fault-Tolerant Control Systems," NASA TM-101665, Nov. 1989.

[2] M. Berg and I. Koren, "On Switching Policies for Modular Redundancy Fault-Tolerant Computing Systems," *IEEE Trans. Computers*, vol. 36, no. 9, pp. 1,052-1,062, Sept. 1987.

[3] P.K. Chande, A.K. Ramani, and P.C. Sharma, "Modular TMR Multiprocessor System," *IEEE Trans. Industrial Electronics*, vol. 36, no. 1, pp. 34-41, Feb. 1989.

[4] N. Gaitanis, "The Design of Totally Self-Checking TMR Fault-Tolerant Systems," *IEEE Trans. Computers*, vol. 37, no. 11, pp. 1,450-1,454, Nov. 1988.

[5] A.L. Hopkins Jr., T.B. Smith III, and J.H. Lala, "FFTMP—A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," *Proc. IEEE*, vol. 66, no. 10, pp. 1,221-1,239, Oct. 1978.

[6] M. Kameyama and T. Higuchi, "Design of Dependent-Failure-Tolerant Microcomputer System Using Triple-Modular Redundancy," *IEEE Trans. Computers*, vol. 29, no. 2, pp. 202-205, Feb. 1980.

[7] H. Kim and K.G. Shin, "On Reconfiguration Latency in Fault-Tolerant Systems," *Proc. IEEE 1995 Aerospace Applications Conf.*, pp. 287-301, Snowmass at Aspen, Colo., Feb. 1995.

[8] I. Koren and Z. Koren, "Analysis of a Class of Recovery Procedures," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 703-712, Aug. 1986.

[9] Y.H. Lee and K.G. Shin, "Optimal Design and Use of Retry in Fault-Tolerant Computing Systems," *J. ACM*, vol. 35, pp. 45-69, Jan. 1988.

[10] T.-H. Lin and K.G. Shin, "An Optimal Retry Policy Based on Fault Classification," *IEEE Trans. Computers*, vol. 43, no. 9, pp. 1,014-1,025, Sept. 1994.

[11] S.R. McConnel, D.P. Siewiorek, and M.M. Tsao, "The Measurement and Analysis of Transient Errors in Digital Computer Systems," *Digest of Papers, FTCS-9*, pp. 67-70, June 1979.

[12] C.V. Ramamoorthy and Y.-W. Han, "Reliability Analysis of Systems with Concurrent Error Detection," *IEEE Trans. Computers*, vol. 24, no. 9, pp. 868-878, Sept. 1975.

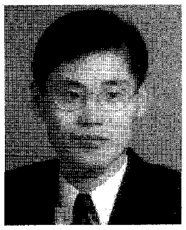
[13] K.G. Shin and H. Kim, "Derivation and Application of Hard Deadlines for Real-Time Control Systems," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1,403-1,413, Nov./Dec. 1992.

[14] K.G. Shin and H. Kim, "A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods," *IEEE Trans. Computers*, vol. 43, no. 10, pp. 1,151-1,162, Oct. 1994.

[15] 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," *Proc. IEEE*, vol. 66, no. 10, pp. 1,178-1,199, Oct. 1978.

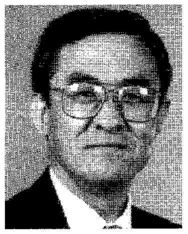
[16] J.F. Wakerly, "Microcomputer Reliability Improvement Using Triple-Modular Redundancy," *IEEE Trans. Computers*, vol. 64, no. 6, pp. 889-895, June 1976.

[17] X.-Y. Zhuo and S.-L. Li, "A New Design Method of Voter in Fault-Tolerant Redundancy Multiple-Module Multi-Microcomputer System," *Digest of Papers FTCS-13*, pp. 472-475, June 1983.



Hagbae Kim (S'90–M'94) received the BS degree from in electronics engineering from Seoul National University, Seoul, Korea, in 1988, and the MS and PhD degrees in electrical engineering and computer science from the University of Michigan, Ann Arbor, in 1990 and 1994, respectively.

Currently, he is an assistant professor of electrical engineering at Yonsei University, Seoul, Korea. He was a National Research Council (NRC) resident research associate at NASA Langley Research Center, Hampton, Virginia, from 1994 to 1996, where he participated in the Highly Intensity Radiation Field (HIRF) project assessing the effects of ElectroMagnetic Interference (EMI) on digital controller computers. His current research interests include real-time control systems, automation of manufacturing systems, fault-tolerant computing, reliability modeling, and probability and stochastic processes.



Kang G. Shin received the BS degree in electronics engineering from Seoul National University, Seoul, Korea in 1970, and both the MS and PhD degrees in electrical engineering from Cornell University, Ithaca, New York, in 1976 and 1978, respectively. Dr. Shin is a professor and director of the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor.

He has authored or coauthored more than 360 technical papers (about 150 of these in archival journals) and numerous book chapters in the areas of distributed real-time computing and control, fault-tolerant computing, computer architecture, robotics and automation, and intelligent manufacturing. He has written (jointly with C.M. Krishna) a textbook, *Real-Time Systems*, which is scheduled to be published by McGraw-Hill in 1996. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are investigating various issues related to real-time and fault-tolerant computing.

Dr. Shin has also been applying the basic research results of real-time computing to multimedia systems, intelligent transportation systems, and manufacturing applications ranging from the control of robots and machine tools to the development of open architectures for manufacturing equipment and processes. (The latter is being pursued as a key thrust area of the newly-established NSF Engineering Research Center on Reconfigurable Machining Systems.)

He is an IEEE fellow and chaired the IEEE Technical Committee on Real-Time Systems during 1991-1993, was a distinguished visitor of the Computer Society of the IEEE, an editor of *IEEE Transactions on Parallel and Distributed Computing*, and an area editor of the *International Journal of Time-Critical Computing Systems*.