Determination of an Optimal Retry Time in Multiple-Module Computing Systems

Chao-Ju Hou Kang G. Shin

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

Abstract

The 'optimal' (in some sense) amount of time used for (or the optimal number of times) retrying an instruction upon detection of an error in a computing system is usually determined under the assumption that the system is composed of a single module, within which all fault activities are confined until some module-replacement action is taken. However, a computing system is usually composed of at least three modules, namely, CPU, memory, and I/O, and the execution of an instruction often requires the cooperation of two or more modules. It is thus more realistic to consider the fault activities in multiple-module systems.

In this paper, we first relax the single-module assumption and model the fault activities in a multiple-module system as a Markov process. We apply the randomization approach to decompose the Markov process into a discrete-time Markov chain subordinated to a Poisson process. Using this decomposition, we can derive several interesting measures, such as (1) the conditional probability of successful retry given a retry period and the fact that a non-permanent fault has occurred, and the mean time until which all modules in the system enter a fault-free state. All the measures derived are used to determine, along with the parameters characterizing fault activities and costs of recovery techniques, (i) whether or not retry should be used as a first-step recovery means upon detection of an error and (ii) the best retry period subject to a specific probability of successful retry.

1 Introduction

The first step of fault-tolerance is to recognize the existence of an error. A fault is defined as the malfunction of a physical component or a bug in software, and it does not affect the system's function until its manifestation, or the occurrence of an error. Existence of an error is recognized through detection mechanisms. The second step, system reconfiguration, is to isolate the faulty component from the rest of the system and reconfigure hardware/software, so that the whole system remains operational. Finally, the computational processes which have been contaminated by the error have to be recovered. The latter two steps can usually be achieved by employing *recovery* techniques.

Various recovery techniques have been proposed to handle different types of fault: *permanent*, *intermittent*, and *transient*. Permanent faults are solid/hard failures and persist forever, which result mainly from component aging. Transient faults are caused mainly by temporary changes in environmental, electrical, or mechanical conditions. They may be active for an unpredictable period of time and die out. Intermittent faults are usually the results of manufacturing defects such as loose connections or bonds. They cycle between active and inactive states, also in an unpredictable manner. Since no single recovery technique is known to be effective against <u>all</u> possible faults, we must usually use a combination of several recovery techniques.

A typical procedure for handling faults consists of instruction retry, program rollback, program reload and restart, and module replacement [1, 2, 3]. Instruction retry [4, 5] is usually applied as the first step of error recovery. Whenever an error is detected, the latest unsuccessful instruction is repeated. If this retry is not successful, one can employ program rollback and/or program reload and restart. In case of program rollback, the state of the program is periodically saved on other safe devices. The state includes the values of program variables and the contents of the internal registers. The saved states of a program are called *checkpoints* or *recovery points*. When an error is detected, the most recently saved checkpoint for the program will be loaded, and the program resumes execution from that checkpoint. If all these recovery techniques fail, one has to resort to system diagnosis and reconfiguration, i.e., identify and remove the faulty module and resume the execution on a new fault-free processor.

Since instruction retry requires little additional hardware and software as compared to the other recovery techniques and thus smaller program completion and recovery overheads, it is usually used as a first-step recovery means. However, retry will not always succeed in recovering the system from faults/errors. Specifically, an instruction retry will be successful only if the following two conditions are satisfied:

C1. The system failed during the execution of the latest uncompleted instruction. This condition can be sat-

The work reported in this paper was supported in part by the Office of Naval Research under Grants N00014-92-J-1080 and N00014-91-J-1226. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the ONR.
The work reported in this paper was supported in part by the National Science Foundation under Grant MIP-9203895, and the Office of Naval Research under Grants N00014-92-J-1080.

isfied if errors are detected upon their occurrence by some signal-level detection mechanism [1], i.e., zero error latency.

C2. The existing fault disappears during the time of retry or *retry period*, i.e., the retry period should be long enough (by perhaps retrying the same instruction more than once)[1] so that the fault dies out within this period.

We assume in this paper that **C1** can be achieved by employing on-line detection mechanisms with high coverage. That is, an error is confined to a module where the fault causing that error had occurred and the affected process can be restored to integrity. One consequence of **C1** is that the damage caused by the fault is recoverable by restoring the process to some prior fault-free state and all data needed to retry the instruction are available. **C2** is impossible in case of permanent faults. Fortunately, only less than 10%, and perhaps as few as 2%, of errors are known to be caused by permanent faults [6, 2]. Retry for a non-permanent fault is likely to succeed if a retry period is selected properly. Moreover, its low overhead and cost makes retry the most cost-effective means to recover from non-permanent faults. Hence, retry is usually applied first upon detection of an error, and the retry period should be controlled so as to maximize the difference between the expected gain in performance that results from retrying for non-permanent faults and the expected loss that results from retrying for permanent faults. The main intent of this paper is to derive an optimal retry period by maximizing this difference.

The design/analysis of various recovery procedures has been addressed by numerous researchers [2, 3, 4, 5]. In contrast to their approaches, we determine the optimal retry period from a *system-oriented* view. We first construct a continuous-time Markov chain which characterizes fault activities in a multiple-module system. Using the randomization approach [7, 8], we then derive the probability of successful retry given a retry period, the mean end-of-retry time, and the distribution of the time for the system to enter a fault-free state. Using these quantities, we can determine (1) whether retry should be applied before applying a different recovery technique based on the parameters characterizing fault activities — e.g., fault occurrence rate, the probability of a fault being permanent, transient, or intermittent, and the distribution of active/benign duration for non-permanent faults — and recovery costs, and (2) the minimum retry period that achieves a given probability of successful retry.

Another point that differentiates our work from others is that we relax the commonly used assumption that all fault activities are confined in a single module until some module replacement action is taken. Note that a computing system is composed of at least three modules (i.e., CPU, memory, and I/O), and execution of an instruction usually requires the cooperation of multiple modules. Hence we must consider *fault activities in multiple modules*. To our best knowledge, this is the first to relax the single-module assumption in determining the retry period and model fault activities in a multiple-module system.

[1]Since it is easy to convert a retry period to the number of retries, the term "retry period" will be used throughout the paper.

The rest of the paper is organized as follows. In Section 2, we outline the important results in the randomization technique. Section 3 describes the fault model used, the assumptions made, the continuous-time Markov chain that characterizes the fault model, and the quantities to be derived. In Section 4 we analytically derive the optimal retry period using the quantities derived in Section 2. Section 5 gives representative numerical examples to demonstrate the system recovery behavior under different fault occurrence conditions. We conclude this paper with Section 6.

2 Preliminary: Randomization

We use the randomization approach first introduced in [8, 9] to compute the probability of successful retry given a retry period. Randomization is commonly used as a computational method to compute transient probabilities of Markov processes with finite state spaces. The main idea of this approach is to transform the Markov process into a Markov chain subordinated to a Poisson process as explained below. Consider a Markov process $\{X(t), t \geq 0\}$ on a finite state space $S = \{0, 1, \ldots, N\}$. Let q_{ij} denote the corresponding transition rate from state i to state j, and let $q_i \triangleq \sum_{j \neq i} q_{ij}$ denote the rate of leaving state i. Then, the generator matrix, Q, of the Markov process can be expressed as:

$$Q = \begin{pmatrix} -q_0 & q_{01} & q_{02} & \cdots & q_{0N} \\ q_{10} & -q_1 & q_{12} & \cdots & q_{1N} \\ \vdots & & & & \\ q_{N1} & q_{N2} & q_{N3} & \cdots & -q_N \end{pmatrix}. \tag{2.1}$$

Let Λ be any value such that $\Lambda \geq q_i, \forall i$. At some time instant, if the process is in state i, then it leaves state i at rate q_i, but this is equivalent to assuming that transitions occur at rate Λ, but only the fraction q_i/Λ of them are real (in the sense that the process really leaves state i and enters some other state and hence real transitions occur at rate q_i), and the remaining fraction $1 - \frac{q_i}{\Lambda}$ are fictitious transitions which keep the process in state i. That is, any Markov process can be thought of as being a process which spends an exponential amount of time with rate Λ in state i and then makes a transition to state j with a probability

$$P_{ij} = \begin{cases} 1 - \frac{q_i}{\Lambda}, & j = i, \\ \frac{q_{ij}}{\Lambda}, & j \neq i, \end{cases} \tag{2.2}$$

or in matrix form,

$$P = Q/\Lambda + I.$$

Consequently, there exist two component processes in a Markov process: a discrete-time Markov chain $\{Y_n, n = 0, 1, \ldots\}$ on S with transition matrix $P = Q/\Lambda + I$, and a Poisson process $\{N(t), t \geq 0\}$ with rate Λ. Moreover, $\{Y_n, n = 0, 1, \ldots\}$ and $\{N(t), t \geq 0\}$ are independent of each other, and the process $\{Y_{N(t)}, t \geq 0\}$ has the same finite dimensional distribution as (and is thus probabilistically identical to) the original Markov process.

This decomposition not only gives a physical interpretation of Markov processes, but also facilitate the computation of transient probabilities of a Markov process. Specifically, let the transient probabilities of the Markov process be $\pi_i(t) =$

$P(X(t) = i)$, and $\Pi(t) = (\pi_0(t), \pi_1(t), \pi_2(t), \dots, \pi_N(t))$. Then, conditioning on the number of occurrences of the Poisson process in $[0, t]$, and using the ***law of total probability***, we have

$$\begin{aligned}\pi_i(t) &= P(X(t) = i) = P(Y_{N(t)} = i) \\ &= \sum_{n=0}^{\infty} P(Y_{N(t)} = i \mid N(t) = n) \cdot P(N(t) = n) \\ &= \sum_{n=0}^{\infty} P(Y_n = i) \cdot \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}, \qquad (2.3)\end{aligned}$$

where $P(Y_n = i) \equiv \phi_i(n)$ is the probability of the Markov chain being in state i at the nth step. Let $\Phi(n) = (\phi_1(n), \phi_2(n), \dots)$, then Eq. (2.3) can be rewritten as

$$\Pi(t) = \sum_{n=0}^{\infty} \Phi(n) \cdot \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}. \qquad (2.4)$$

Since $\Pi(0) = \Phi(0)$, and $\Phi(n) = \Phi(0)P^n$, where P is the transition matrix of the discrete-time Markov chain (Eq. (2.2)), Eq. (2.4) can be rewritten as

$$\Pi(t) = \sum_{n=0}^{\infty} \Pi(0)P^n \cdot \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}. \qquad (2.5)$$

3 Fault Model and Parameters of Interest

In this section, we first describe the fault model of a multiple-module system. Second, we characterize the fault model with an embedded continuous-time Markov chain under the assumption that at most one fault exists in each module at any moment, and apply the randomization technique to the Markov chain developed. Then, we discuss how to extend the model to the (more general but rare) case that multiple faults are possible on a single module. Although all the concepts and expressions are derived for an arbitrary number of modules, we confine our illustrative examples to the case of three modules for the clarity of presentation.

3.1 Fault Model

We assume that faults arrive at the ith module according to a time-invariant Poisson process with rate λ_i. We also assume that transient, intermittent, and permanent faults occur with probability p_{tf}, p_{if}, p_{pf}, respectively, and their occurrences are independent of one another. Consequently, transient, intermittent, and permanent faults occur at exponential rate $\lambda_i p_{tf}$, $\lambda_i p_{if}$, and $\lambda_i p_{pf}$, respectively. If a permanent fault occurs, it remains persistent in the system until the component containing the fault is replaced. If a transient fault occurs, it disappears after an active duration, where the active duration is exponentially distributed with rate τ_i. If an intermittent fault occurs, it may become benign after an active duration, and then reappear after a benign duration, where the active and benign time are exponentially distributed with rate μ_i and ν_i, respectively. That is, an intermittent fault cycles between active and benign states.

Because instruction retry is effective only if an error is detected upon its occurrence, we assume that errors are detected

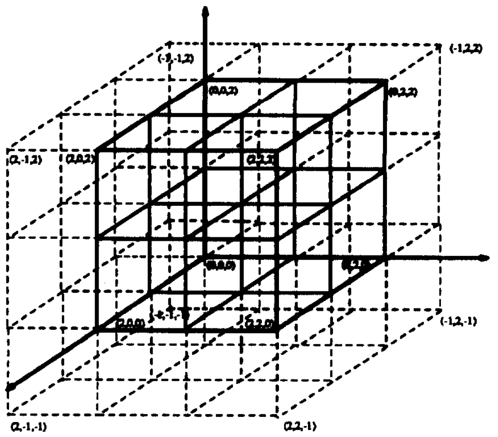

Figure 1: The fault model for a three-module system. -1=TF=transient fault, 0=NF=no fault, 1=IF=intermittent fault, 2=BF=benign fault. State transitions and their transition rates are not shown for clarity.

immediately upon their occurrence by, for example, signal-level detection mechanisms [1]. Also, faults occurred in one module are assumed not to affect other modules, i.e., fault occurrences in different modules are statistically independent. This assumption results from the fact that faults are usually the malfunction of hardware components, and are independent of one another [6, 1]. Last, we assume that there is at most one fault in each module at any time, since the inter-arrival time of faults is usually much larger than any other fault-related durations. We will discuss in Section 3.3 how to relax the last assumption.

3.2 Construction of a Continuous-Time Markov Process

Under the assumptions of fault behavior in Section 3.1, we model a multi-module system with a Markov process with an example state space Fig. 1 where the system consists of three modules. (State transitions and their rates are not shown for clarity.) Because permanent faults cannot be recovered by retry, they are excluded from the Markov chain which will be used to derive effective retry periods. The state space S consists of state vectors of the form $(s_1, s_2, ..., s_n)$, where n is the number of modules in the system, and $s_i \in \{-1, 0, 1, 2\}$ represents the state of the ith module with the following interpretation:

- **-1** represents the transient-fault (TF) state, i.e., there exists a transient fault in the module.
- **0** represents the no-fault (NF) state, i.e., no fault exists in the module.
- **1** represents the intermittent-fault (IF) state, i.e., there exists an active intermittent fault in the module.
- **2** represents the benign-fault (BF) state, i.e., an intermittent fault has become inactive in the module.

For example, in Fig. 1, the state vector $(1, 2, 0)$ indicates that there exists an active intermittent fault in the first mod-

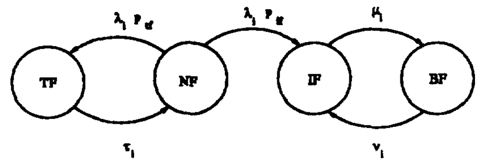

Figure 2: One-dimensional state transition diagram.

ule (CPU), a benign intermittent fault in the second module (memory), and no fault in the third module (I/O).

The Markov model is flexible in the sense that it allows for a variety of fault patterns. Also, the model allows for a situation where different sets of faults may occur to different modules. For example, if only transient, transient, intermittent faults could occur in the first, second, third module, respectively, in a three-module system, then $s_1 \in \{0,-1\}, s_2 \in \{0,-1\}, s_3 \in \{0,1,2\}$.

Recall that retry will succeed only if all faults have disappeared during the retry period t, that is, the system has moved to a state vector none of whose components are 1 or -1 during the period t. On the other hand, if the system only moves among states in which at least one component is 1 or -1 during the retry period t, the retry would fail. Based on this observation, we divide S into Failed Set (FS) and Successful Set (SS), where

$$FS = \{(s_1, s_2, ..., s_n) : \exists i \text{ such that } |s_i| = 1\},$$

and

$$SS = \{(s_1, s_2, ..., s_n) : |s_i| \neq 1\ \forall i\}.$$

For example, in the case of a three-module system, if all three types of faults are possible, we have

$$\begin{aligned} SS \;=\; \{\; & (0,0,0),(0,0,2),(0,2,0),(2,0,0),(0,2,2), \\ & (2,0,2),(2,2,0),(2,2,2)\;\}, \end{aligned}$$

and all the other 56 states belong to FS.

Note that state transitions along the ith coordinate are associated with the state evolution in the ith module. Due to the assumption that faults occur independently among modules, state transitions along the same coordinate exhibit the same behavior. For example, state transitions in Fig. 1 between (0,0,0) and (1,0,0) are the same as those between $(0,s_2,s_3)$ and $(1,s_2,s_3)$, $\forall s_2,s_3 \in \{-1,0,1,2\}$, because they all describe the fault evolution from NF to IF in the first module. Moreover, state transitions and their physical meanings along one coordinate are virtually the same as those along another coordinate except that fault activities/transitions correspond to a different module and are perhaps with different rates along different coordinates. Consequently, it suffices to characterize the state evolution of the system by a one-dimensional state-transition diagram shown in Fig. 2, where the transition rates from 0(NF) to -1(TF), -1(TF) to 0(NF), 0(NF) to 1(IF), 1(IF) to 2(BF), 2(BF) to 1(IF) are denoted as $\lambda_i p_{tf}$, τ_i, $\lambda_i p_{if}$, μ_i, and ν_i, respectively.

The system state evolution can be described as a Markov process $\{X(t), t \geq 0\}$ on the state space $S = \{(s_1, s_2, ..., s_n) : s_i \in \{-1,0,1,2\}, n \in \mathcal{N}$ is the number of modules$\}$. Using the randomization technique summarized in Section 2, we can decompose $\{X(t), t \geq 0\}$ into a discrete-time Markov chain, $\{Y_n, n = 0,1,\ldots\}$, embedded in a Poisson process, $\{N(t), t \geq 0\}$ with rate

$$\begin{aligned} \Lambda \;=\; & \max_{(s_1,s_2,...,s_n)\in S}\{(\sum_{i \text{ s.t. } s_i = 0} \lambda_i(p_{tf}+p_{if}) + \sum_{i \text{ s.t. } s_i = -1} \\ & \tau_i + \sum_{i \text{ s.t. } s_i = 1} \mu_i + \sum_{i \text{ s.t. } s_i = 2} \nu_i \;)\;\}, \end{aligned} \tag{3.1}$$

where each term in Eq. (3.1) is the transition rate of some state $(s_1, s_2, ..., s_n)$.

3.3 Extension to Multiple-Fault Case

The Markov model described in Section 3.2 can be extended to the more general case that allows multiple faults in a single module as follows. The state space S now consists of state vectors $(\underline{s_1}, \underline{s_2}, ..., \underline{s_n})$, where n is the number of modules in the system, and $\underline{s_i}$ describes the state of the ith module and is a three-tuple

$$\underline{s_i} \triangleq s_1^i\ s_2^i\ s_3^i,$$

where s_1^i, s_2^i, and s_3^i are the number of transient faults, active intermittent faults, and benign intermittent faults, in the ith module, respectively. We assume that $0 \leq s_j^i \leq K$, $j = 1,2,3$, where K is a sufficiently large number so that the quantities of interest derived from the model that uses K and those derived from the model that uses $K+1$ are within a specific error of tolerance.[2]

Similar to the model described in Section 3.2, state transitions along the ith coordinate are associated with the state evolution in the ith module, and can be uncoupled with state transitions along other coordinates under the assumption that faults occur independently among modules. Consequently, the state evolution in the system can be characterized by the state transition diagram that describes fault activities in one module, as shown in Fig. 3 (where only transitions around $\underline{s_i} = s_1^i\ s_2^i\ s_3^i$ are shown). Note that the number of allowable states for one module (i.e., along one coordinate) is now $(K+1)^3$ (instead of 4). The transition rates are derived in a straightforward manner as in Fig. 2. For example, the transition from $s_1^i\ s_2^i\ s_3^i$ to $(s_1^i+1)\ s_2^i\ s_3^i$ takes place when a transient fault occurs in the ith module and is thus with rate $\lambda_i p_{tf}$. The transition from $s_1^i\ s_2^i\ s_3^i$ to $s_1^i\ (s_2^i-1)\ (s_3^i+1)$ occurs when an active intermittent fault in the ith module becomes benign and is thus with rate $s_2^i \mu_i$.

The system under consideration can then be described as a Markov process $\{X(t), t \geq 0\}$ on the state space

$$S = \{(\underline{s_1}, \underline{s_2}, ..., \underline{s_n}) : \underline{s_i} = s_1^i\ s_2^i\ s_3^i, 0 \leq s_j^i \leq K, j = 1,2,3\}$$

to which randomization can be applied to get a discrete-time Markov chain, $\{Y_n, n = 0,1,2,...\}$, and a Poisson process, $\{N(t), t \geq 0\}$, with rate

$$\Lambda = \max_{(\underline{s_1},\underline{s_2},...,\underline{s_n})\in S}\{\;\sum_{i=1}^{n}(\lambda_i(p_{tf}+p_{if}) + s_1^i\tau_i + s_2^i\mu_i + s_3^i\nu_i)\;\},$$

[2]According to our simulation results, the value of K needs not be large, because the fault occurrence rate, λ_i, is usually several orders smaller than the other rates, and thus, the probability that multiple faults exist is usually negligible.

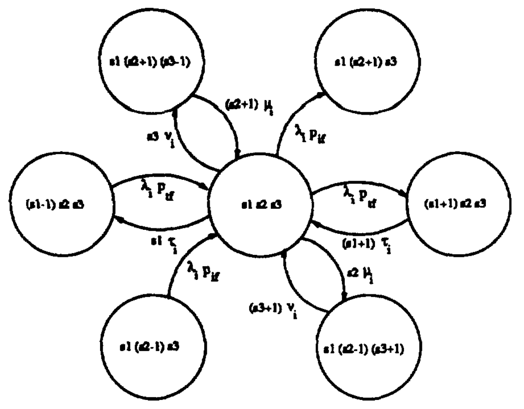

Figure 3: "One dimensional" state transition diagram for the case which allows multiple faults on a single module. $0 \leq s_j \leq K$, $j = 1,2,3$. Only transitions around $\underline{s_i} = s_1^i\ s_2^i\ s_3^i$ are shown. Transitions are applicable only when the corresponding states exist.

where $\sum_i(\lambda_i(p_{tf} + p_{if}) + s_1^i\tau_i + s_2^i\mu_i + s_3^i\nu_i)$, as shown in Fig. 3, is the transition rate of state $(\underline{s_1}, \underline{s_2}, ..., \underline{s_n})$.

The set of failed states, FS, and the set of successful states, SS, can be identified as

$$FS = \{(\underline{s_1}, \underline{s_2}, ..., \underline{s_n}) : \exists i \text{ such that } s_1^i + s_2^i \neq 0\},$$

and

$$SS = \{(\underline{s_1}, \underline{s_2}, ..., \underline{s_n}) : s_1^i + s_2^i = 0, \forall i\}.$$

3.4 Parameters to be Derived

Many transient quantities can be derived by applying the randomization technique on the decomposed discrete-time Markov chain and Poisson process. Specifically, we want to derive:

$P_{rs}(t)$: the probability that retry succeeds given that the retry period is t and a non-permanent fault has occurred. Using this information, we can determine the retry period for a specified probability of successful retry.

$E(L(t))$: the mean end-of-retry period. Specifically, let $r \in (0, \infty)$ be the time for the system to first enter a state that belongs to SS, and let $L(t) = \min(t, r)$ be the end-of-retry time given a maximum retry period of t, i.e., the time at which the retry stops because either all the faults have disappeared (so a successful retry, $r < t$) or the retry period is exhausted ($r \geq t$). Obviously, $E(L(t)) = t$ if at least one permanent fault occurs in the system (i.e., retry will never succeed). On the other hand, $E(L(t))$ is finite for non-permanent faults, and provides a mean estimate of retry costs.

$P(T_{SS} \leq t)$: the distribution of the first SS-passage time. Specifically, let T_{SS} be the first time the system visits a state that belongs to SS, i.e.

$$T_{SS} = \min\{t : X(t) \in SS\},$$

With this probability distribution, we can compute the mean SS-passage time, $E[T_{SS}]$. By the definitions and interpretations of $P_{rs}(t)$, T_{SS}, and $L(t)$, the following relations hold: (1) $P_{rs}(t) = P(T_{SS} \leq t)$, and (2) $E(T_{SS}) = \lim_{t\to\infty} E(L(t))$, both of which will be used to verify the correctness of our derivation.

4 Derivation

In Section 3, we modeled the fault evolution as a Markov process $\{X(t), t \geq 0\}$ on a finite space $S = \{(s_1, s_2, ..., s_n) : s_i \in \{-1, 0, 1, 2\}\}$ (or $S = \{(\underline{s_1}, \underline{s_2}, ..., \underline{s_n}) : \underline{s_i} = s_1^i\ s_2^i\ s_3^i,\ 0 \leq s_j^i \leq K, j = 1,2,3\}$ in the case of multiple faults in a single module), where S can be decomposed into two mutually exclusive subsets, FS and SS. Also, the constructed Markov process $\{X(t), t \geq 0\}$ can be decomposed by the randomization technique into a discrete-time Markov chain Y_n subordinated to a Poisson process $N(t)$, which is used in this section to derive parameters of interest.

4.1 Probability of Successful Retry

Let $p(n,k)$, $0 \leq k \leq n+1$, denote the probability that the underlying discrete-time Markov chain (obtained after randomization) visits k fault states (i.e., states in FS) given n state changes. For example, $p(n, n+1)$ is the probability that the underlying Markov chain always stays in fault states during these n state changes. Consequently, the probability that a retry of period t fails is the probability that the underlying Markov chain always stays in fault states regardless of the number of state changes in $[0, t]$, i.e.,

$$\begin{aligned} 1 - P_{rs}(t) &= \sum_{n=0}^{\infty} p(n, n+1) \cdot P(n \text{ state changes in time } t) \\ &= \sum_{n=0}^{\infty} p(n, n+1) \cdot \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}, \end{aligned}$$

or,

$$P(t) = 1 - \sum_{n=0}^{\infty} p(n, n+1) \cdot \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}, \qquad (4.1)$$

where Λ is the rate of the underlying Poisson process obtained after uniformization, and is given in Eq. (3.1). The error resulting from the truncation of the infinite sum in Eq. (4.1) can be easily bounded. Specifically, let R_m denote the error resulting from truncating Eq. (4.1) to m steps, then

$$\begin{aligned} R_m &= \sum_{n=m+1}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} \cdot p(n, n+1) \\ &\leq \sum_{n=m+1}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!} = 1 - \sum_{n=0}^{m} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}, \end{aligned}$$

where $\leq$ in the above expression results from the inequality $p(n, n+1) \leq 1$. m can be evaluated *a priori* for a given error tolerance.

Now, the remaining task is to calculate $p(n,k)$. Let $p(n,k,a_i)$ be the probability of the underlying Markov chain visiting fault states k times out of n steps and let a_i be the state visited after the last transition. We have

$$p(n,k,a_i) = \begin{cases} \sum_{j=1}^{|S|} p(n-1,k-1,a_j)\cdot P_{ji} & \text{if } a_i \in FS, \\ \sum_{j=1}^{|S|} p(n-1,k,a_j)\cdot P_{ji} & \text{if } a_i \in SS. \end{cases}$$

The initial conditions are given by

$$p(0,1,a_i) = \begin{cases} \pi_i(0), & \text{if } a_i \in FS, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$p(0,0,a_i) = 0.$$

Note that whenever an instruction is retried, the system must be in a fault state, i.e., k in $p(n,k,a_i)$ must be ≥ 1, thus $p(0,0,a_i)=0$, $\forall a_i$. Finally, by the law of total probabilities,

$$p(n,k) = \sum_{i=1}^{|S|} p(n,k,a_i).$$

4.2 Mean End-of-Retry Time

Recall that r is defined in Section 3 as the time the system first enters a state $\in SS$ and $L(t) = \min(t,r)$ is defined as the end-of-retry time given that the retry period is t. Analytically,

$$\begin{aligned} E[L(t)] &= \int_0^t (1-P(s))\,ds \qquad (4.2) \\ &= \int_0^t \sum_{n=0}^{\infty} p(n,n+1)\cdot \frac{e^{-\Lambda s}(\Lambda s)^n}{n!}\,ds \\ &= \frac{1}{\Lambda}\cdot \sum_{n=0}^{\infty} \Big(1-\sum_{j=0}^{n} \frac{e^{-\Lambda t}(\Lambda t)^j}{j!}\Big)\cdot p(n,n+1) \end{aligned}$$

As $t \to \infty$, $E[L(t)]$ becomes the mean time for the system to enter a state $\in SS$, and can be used to indicate whether or not retry should be used for a particular system configuration, i.e.,

$$\lim_{t\to\infty} E[L(t)] = \frac{1}{\Lambda}\cdot \sum_{n=0}^{\infty} p(n,n+1). \qquad (4.3)$$

4.3 Distribution of First SS-Passage Time

Recall that T_{SS} is defined as the time the system first enters a state $\in SS$, i.e., $T_{SS} = \min\{t : X(t) \in SS\}$. Randomization can be used to compute the distribution of T_{SS} as follows: we define an associated process $\{X_{SS}(t), t \geq 0\}$ on the state space $FS \cup \{S_a\}$, where S_a is the absorbing state formed by collapsing all states in SS into it. The corresponding generator matrix Q_{SS} can be expressed as

$$\begin{aligned} q_{SS;i,j} &= q_{i,j} && \text{for } i,j \notin SS, \\ q_{SS;i,S_a} &= \textstyle\sum_{j\in SS} q_{i,j} && \text{for } i \notin SS, \\ q_{SS;S_a,j} &= 0 && \text{for all } j. \end{aligned}$$

Note that the transition intensities of $\{X_{SS}(t), t \geq 0\}$ are identical to $\{X(t), t \geq 0\}$ except that there does not exist any transfer out of S_a (or equivalently, all states in SS). Consequently, we have

$$P(T_{SS} \leq t) = P(X_{SS}(t) = S_a). \qquad (4.4)$$

That is, the distribution of the time until the system first enters a state in SS (left-hand side of Eq. (4.4)) is equivalent to computing a transient state probability for the Markov process, $X_{SS}(t)$, where all states in SS are lumped into S_a, and has the generator matrix Q_{SS} (right-hand side of Eq. (4.4)). Using the uniformization technique, we have (via Eq. (2.5))

$$\begin{aligned} P(X_{SS}(t) = S_a) &= \sum_{n=0}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}\cdot P(X_{SS,n} = S_a) \\ &= \sum_{n=0}^{\infty} \frac{e^{-\Lambda t}(\Lambda t)^n}{n!}\cdot \pi_{SS,S_a}(n), \qquad (4.5) \end{aligned}$$

where $\pi_{SS,S_a}(n)$ can be computed from

$$\Pi_{SS}(n) = \Pi_{SS}(n-1)\, P_{SS},$$

and P_{SS} is the transition probability matrix and can be computed from Q_{SS} by the same approach as in Eq. (2.2).

4.4 Determination of Retry Period

The significance of the quantities derived above lies in that they can be used to determine: (1) whether or not to apply instruction retry as a first-step recovery means, and (2) the smallest retry period that achieves a specified probability of successful retry. Specifically, let $C_1(t)$ and C_2 denote the cost function of instruction retry given retry period t and the cost function of applying other time-redundancy recovery techniques (e.g., program rollback, program reload and restart),[3] respectively, and let P_{req} denote the required probability of successful retry (which is given as a design parameter). Obviously, $C_1(t)$ is a monotonically non-decreasing function of t.

The first question can be answered as follows. If there does not exist any $t > 0$ such that

$$C_1(t) + (1-(1-p_{pf})\cdot P_{rs}(t))\cdot C_2 \leq C_2, \qquad (4.6)$$

or, in a simpler form,

$$C_1(t) \leq (1-p_{pf})\cdot P_{rs}(t)\cdot C_2, \qquad (4.7)$$

then retry should not be applied, where p_{pf} is the probability of a fault being permanent, and $(1-(1-p_p)\cdot P_{rs}(t))$ is the probability that retry fails given a retry period t. That is, if, for every t, the cost of retrying for the period t as the first-step recovery means (the left hand side of Eq. (4.6)) is greater than the cost of not applying instruction retry (the right hand side of Eq. (4.6)), then retry should not be applied at all. The second question can be answered by finding the smallest t that satisfies both Eq. (4.7) and

$$(1-p_{pf})\cdot P_{rs}(t) \geq P_{req}. \qquad (4.8)$$

[3] C_2 would be dependent on the retry period t if the assumption C1 does not hold. That is, if errors cannot be detected upon their occurrence, the latest checkpoint used in program rollback might have been contaminated by error propagation, and the program may be forced to roll back to an earlier checkpoint, resulting in a higher cost.

$P_{rs}(t) = P(T_{ss} \leq t)$	Parameter set			
t	I	II	III	IV
4	0.30015	0.35948	0.36999	0.37458
10	0.64378	0.46213	0.56871	0.51337
16	0.83057	0.54341	0.68109	0.62513
22	0.92307	0.61422	0.76376	0.71386
28	0.96624	0.67520	0.82500	0.78225
34	0.98553	0.72729	0.87037	0.83444
$E(L(t))$	Parameter set			
t	I	II	III	IV
4	3.4041	3.0049	3.2307	2.9803
10	6.4806	6.5145	6.3159	6.2928
16	7.9954	9.4925	8.5484	8.8660
22	8.7007	12.015	10.201	10.838
28	9.0160	14.142	11.426	12.340
34	9.1524	15.930	12.333	13.483
$\lim E[L(t)] = E(T_{ss})$	9.2532	23.5042	14.2085	16.8511

Table 1: Numerical results of $P(t)$ and $E(L(t))$. The fault that triggers instruction retry is assumed to be on the ith module with probability 1/3, $i = 1, 2, 3$.

5 Numerical Examples

In this section, we present the numerical results derived from the multiple-module model under various fault occurrence conditions. Our numerical experiments indicate that the discrepancy between the model that allows multiple faults in a single module (Section 3.3) and the model that assumes at most one fault in a module at any time (Section 3.2) is negligible for all $\lambda_i \leq 0.1$. Hence, we will focus on the latter model. Also, for clarity of presentation, we confine our illustrative examples to a three-module system. However, as the numerical results indicate, the conclusions drawn from the three-module system are consistent with those from the case of an arbitrary number of modules.

As expected (commented in Section 3), the following relations hold: (1) $P(T_{ss} \leq t) = P_{rs}(t)$, and (2) $E(T_{ss}) = \lim_{t\to\infty} E(L(t))$, both of which verify the correctness of our derivation. Representative numerical results are given in Table 1 and Fig. 4, where the following four sets of parameters are used:

I. $\lambda_i = 10^{-4}$, $\tau_i = 0.2$, $\mu_i = 0.2$, and $\nu_i = 10^{-2}$, $i = 1, 2, 3$;

II. $\lambda_1 = 10^{-6}$, $\tau_1 = 0.8$, $\mu_1 = 0.8$, $\nu_1 = 10^{-3}$; $\lambda_i = 10^{-2}$, $\tau_i = 5 \times 10^{-2}$, $\mu_i = 5 \times 10^{-2}$, and $\nu_i = 10^{-1}$, $i = 2, 3$;

III. $\lambda_1 = 10^{-2}$, $\tau_1 = 5 \times 10^{-2}$, $\mu_1 = 5 \times 10^{-2}$, $\nu_1 = 10^{-1}$; $\lambda_i = 10^{-6}$, $\tau_i = 0.8$, $\mu_i = 0.8$, and $\nu_i = 10^{-3}$, $i = 2, 3$;

IV. $\lambda_1 = 10^{-6}$, $\tau_1 = 0.8$, $\mu_1 = 0.8$, $\nu_1 = 10^{-3}$; $\lambda_2 = 10^{-2}$, $\tau_2 = 5\times10^{-2}$, $\mu_2 = 5\times10^{-2}$, and $\nu_2 = 10^{-1}$; $\lambda_3 = 10^{-4}$, $\tau_3 = 0.2$, $\mu_3 = 0.2$, and $\nu_3 = 10^{-2}$.

$p_{tf} = 0.7$ and $p_{if} = 0.3$ are used in all four parameter sets, and the fault that triggers instruction retry is assumed to be on the ith module with probability 1/3, $i = 1, 2, 3$. Note that parameter set I and IV represent a homogeneous system and a heterogeneous system, respectively, while parameter set II and III represent a fault-prone system with two "bad" modules (i.e., modules that are more prone to faults and that take a longer time for their faults to become inactive.) and a less fault-prone system with two "good" modules, respectively. Several observations are in order:

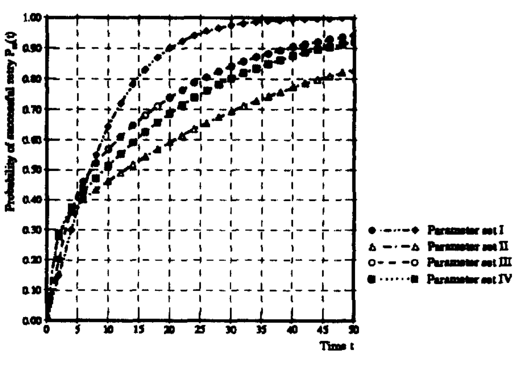

Figure 4: (a) $P_{rs}(t)$ for different parameter sets. The fault that triggers instruction retry is assumed to be on the ith module with probability 1/3, $i = 1, 2, 3$.

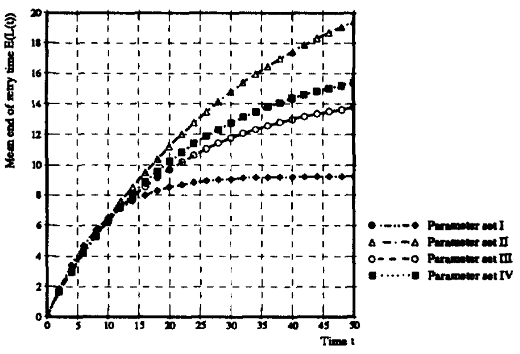

Figure 4: (continued) (b) $E(L(t))$ for different parameter sets.

- If $P_{rs}(t)$ is relatively small for a certain type of systems (so retry is unlikely to succeed for a given time t), the corresponding mean end-of-retry time $E(L(t))$ would be large.
- $P_{rs}(t)$ derived for a parameter set significantly differs from that for another. Consequently, modeling the computing environment as a multiple-module system is necessary to compute an accurate retry period that will achieve a given probability of successful retry.

To investigate how fault parameters affect $P_{rs}(t)$, we vary λ_i, τ_i, μ_i, and ν_i from 10^{-8} to 10^{-1}, 10^{-3} to 5.0, 0.05 to 2.0, 0.05 to 2.0, respectively, for parameter set I, one at a time while keeping the other parameters unchanged. Figs. 5–7 plot the effects of varying λ_i, μ_i, and τ_i on $P_{rs}(t)$. The effect of varying ν_i on $P_{rs}(t)$ is similar to that of varying λ_i, and thus omitted. The effects of varying μ_i and τ_i are more pronounced than varying the other parameters. This is because retry succeeds only after both transient and intermittent faults become inactive, and thus, μ_i and τ_i — that characterize the active duration of transient and intermittent faults — play a dominant role in determining the retry period. On the other hand, varying μ_i has a more notable effect than varying τ_i. This is because an intermittent fault does not die out, but instead cycles between active and benign states. Thus, μ_i has a greater and long-lasting influence on the system.

6 Conclusion

We proposed in this paper a continuous-time Markov model to characterize the fault activities in a multiple-module computing system. The randomization technique is then applied to this model to derive several quantities of interest, i.e., the probability of successful retry given a retry period, the mean end-of-retry time, and the distribution of time for the system to first enter a fault-free state. These quantities can be used to determine whether or not instruction retry should be applied as a first-step recovery technique with respect to fault characteristics and recovery costs, and the smallest retry period that achieves a specified success probability.

Our analysis is valid as long as errors are detected upon their occurrence. Extension to the case where there is a nonzero latency between error occurrence and detection (i.e., relaxation of condition C1 in Section 1) is worthy of further investigation, and will be reported in a forthcoming paper.

References

[1] K. G. Shin and Y.-H. Lee, "Error detection process — Model, design, and its impact on computer performance," *IEEE Transactions on Computers*, vol. C–33, no. 6, pp. 529–540, June 1984.

[2] I. Doren, Z. Koren, and Y. H. Su, "Analysis of a class of recovery procedures," *IEEE Transactions on Computers*, vol. c–35, no. 8, pp. 703–710, August 1986.

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

[4] A. M. Saleh and J. H. Patel, "Transient-fault analysis for retry techniques," *IEEE Transactions on Reliability*, vol. 37, no. 3, pp. 323–330, August 1988.

[5] Y.-H. Lee and K. G. Shin, "Optimal design and use of retry in fault-tolerant computer systems," *Journal of the Association for Computing Machinery*, vol. 35, no. 1, pp. 45–69, January 1988.

[6] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, 1982.

[7] W. K. Grassman, "Transient solutions in markovian queueing systems," *Comput. and Ops. Res.*, vol. 4, pp. 47–53, 1977.

[8] D. Gross and D. R. Miller, "The randomization technique as a modeling tool and solution procedure for transient markov processes," *Operations Research*, vol. 32, no. 2, , March–April 1984.

[9] B. Melamed and M. Yadin, "Randomization procedures in the computation of cumulative–time distributions over discrete state markov processes," *Operat. Res.*, vol. 32, no. 4, pp. 926–944, July–Aug. 1984.

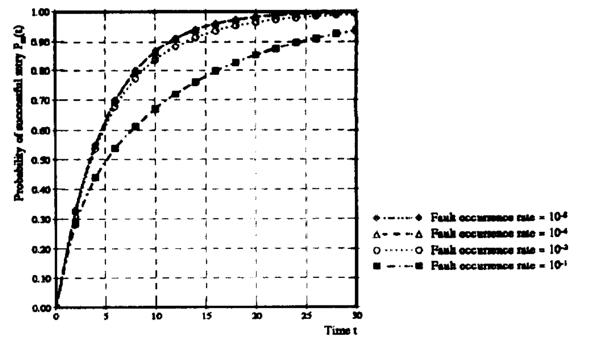

Figure 5: Effect of variation of λ_i on $P_{rs}(t)$. Parameters are specified in parameter set I.

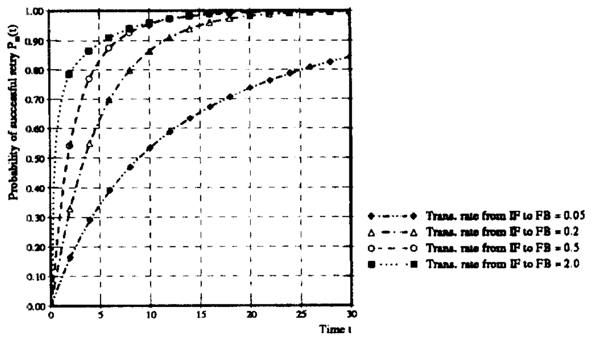

Figure 6: Effect of variation of μ_i on $P_{rs}(t)$.

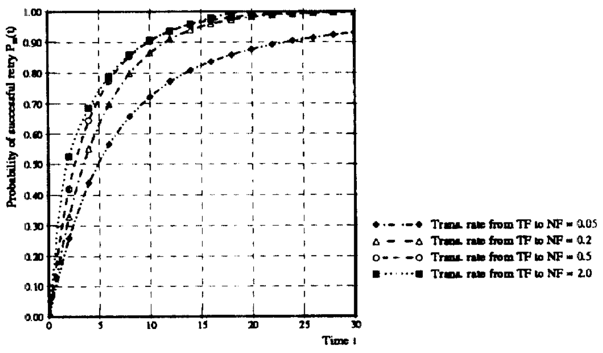

Figure 7: Effect of variation of τ_i on $P_{rs}(t)$.