On Switching Policies for Modular Redundancy Fault-Tolerant Computing Systems

MENACHEM BERG AND ISRAEL KOREN, SENIOR MEMBER, IEEE

Abstract—**The objective of fault-tolerant computing systems is to provide an error-free operation in the presence of faults. The system has to recover from the effects of a fault by employing certain recovery procedures like program rollback, reload, and restart, etc. However, these recovery procedures, result in interruptions in the system's operation, thus reducing the availability of the system for user applications. Fault-tolerant systems for critical applications include, therefore, standby spares that are ready to replace active modules which fail to recover from the effects of a fault. A standby spare may also be used to replace a module suffering from frequent fault occurrences resulting in too many repetitions of the recovery process, in order to increase the availability of the system for user applications.**

In this case a module switching policy is needed indicating upon a fault occurrence, whether to retry a failing module or switch it out and replace it by a spare, considering the remaining mission time and the probability of a system crash. A module switching policy for dynamic redundancy systems is presented in this paper and the improvement in application-oriented availability due to the use of this policy is illustrated.

Index Terms—**Application-oriented availability, deterioration models, failure rate, fault tolerance, modular redundancy, module switching policy, recovery, standby spare.**

I. INTRODUCTION

THE purpose of incorporating fault-tolerance into a computing system is to minimize or even eliminate service interruptions which are due to faults. A fault-tolerant computing system should not be just a system producing error-free outputs but useful error-free outputs. Any time the system is performing tasks which are not directly related to its operational objectives, this time period is considered wasted. Examples of such tasks are error detection procedures, recoveries from failures, restoring of data or programs, reconfigurations of the system, initialization and synchronization processes.

Consequently, we distinguish here between *application-oriented availability* and *system availability*. The latter is defined as the steady-state probability that the system is operational. We define the application-oriented availability as the steady-state probability that the system is operational and is running user applications. It is always less than or equal to the system availability and can be computed from it by subtracting the portion of time that the system is executing the above mentioned tasks.

Note that the application-oriented availability is different from the computational availability as defined in [3]. Computational availability is an appropriate measure in cases where the computational capacity of the system is varying. For systems in which a failure results in an interruption in the user service but does not change the computational capacity of the system, the measure of application-oriented availability is appropriate.

The difference between the application-oriented availability and the classical system availability measure may be quite substantial, e.g., a frequently occurring failure in an active module of a computing system may result in a large number of time-consuming error-recoveries and resynchronizing processes. A situation like this might have an intolerable impact on the performance of a critical real-time control system (e.g., in aerospace applications like vehicle guidance systems). In these cases it might be beneficial to replace such a module with a spare one if available, in order to achieve a higher application-oriented availability.

To study these phenomena we have to consider the possible failure modes and the appropriate recovery procedures. Failures occurring in digital systems come from a variety of sources, e.g., defective components, design deficiencies, and environmental causes. Each of these sources generates faults in a different way and of different types. Still, faults may be roughly classified into three types, permanent faults, intermittent faults, and transient faults [1], [2], [9]–[14]. Permanent faults are solid hardware failures. Intermittent faults are due to physical defects in the hardware that manifest themselves intermittently in an unpredictable manner (in this class we may include data sensitive design errors [4]). Transient faults are due to temporary environmental conditions (such as temperature, humidity, vibration, power fluctuation, electromagnetic fields, etc.) which may persist for an unpredictable period of time [10].

Although intermittent and transient faults disappear after a short duration, their damage to the information structure of the system is in many cases permanent. Since the exact nature of the fault and its effects are usually unknown upon occurrence, recovering from the fault is one of the system's most complex and difficult functions (e.g., [6], [15]). If the system suffers a permanent fault, the failing module must then be identified and replaced. However, it is known that the nonpermanent faults

Manuscript received February 27, 1985; revised November 6, 1985 and October 9, 1986. This work was supported in part by the Electrical, Computer, and System Engineering Division of the National Science Foundation under Grant ECS-81-05276.

M. Berg is with the Department of Industrial Engineering, University of Toronto, Toronto, Ont., Canada M5S1A4, on leave from the Department of Statistics, University of Haifa, Haifa 31999, Israel.

I. Koren is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 on leave from Technion-Israel Institute of Technology, Haifa 3200, Israel.

IEEE Log Number 8715301.

(i.e., intermittent and transient ones) constitute the majority of the faults occurring in computing systems [12], [13]. Hence, recovery capabilities for nonpermanent faults which do not require giving up hardware resources prematurely, are usually built into these systems. Various recovery techniques are used nowadays in fault-tolerant computing systems, like instruction retry, program rollback, reload, and restart, etc. [1], [11]–[13]. None of these techniques can be effective against all possible faults. Hence, several recovery steps are employed in support of one another to increase effectiveness [1], [8], [11]–[13].

Due to the complexity of the recovery problem, each recovery procedure has its deficiencies and may fail and cause a system crash [1], [12], [15]. In some systems, recovery deficiencies account for as much as 35 percent of the down-time [15]. Even if the recovery procedure is successful, an overhead time is involved. First, there is a detection time between fault occurrence and fault detection and then, there is the recovery time including initialization and synchronization of the failing module [5], [6], [11].

Usually, the information processing performed by the system during the detection time (and even before that if a reload and restart operation is needed) is contaminated and must be repeated. Thus, a cost to the system is associated with every recovery procedure activated. The cost of a single application of a recovery procedure might be low; however, a frequently occurring nonpermanent fault will considerably reduce the application-oriented availability of the system. Moreover, the rate of some intermittent faults like those caused by deteriorating or aging components, gradually increases until they become permanent. Their transition into solid faults may take from a few minutes to several months during which the frequency of their occurrence increases intolerably. Such a situation should be avoided on time.

In addition, too many applications of a recovery procedure will undoubtedly increase the probability of an unsuccessful recovery leading to a system failure. Consequently, in some cases it may be worthwhile to switch out the module which is subject to nonpermanent fault occurrences and reconfigure the system.

What is needed, therefore, is a module switching policy that will indicate the kind of operation to be taken by the system whenever a module fails, i.e., should we retry the module that has failed or maybe switch it out and replace it by a spare. This module switching policy should optimize some system cost function. The application-oriented availability measure as defined above is concerned only with the portion of time that the system is operational and running user applications. This measure does not make a distinction between several brief interruptions in service (which may in some cases be harmless), and a single interruption period equaling in length to the sum of the shorter ones. The latter might be disastrous to critical real-time applications.

Hence, instead of maximizing the application-oriented availability we attempt to optimize a more complex version of this availability measure. Namely, we assign one cost per time unit to brief periods of service interruption and a different cost per time unit to longer interruptions, paying special attention to system crashes for which the system becomes inoperational for the remaining mission time. Based on this we define a system cost function which is the expected cost of all system down-time periods (more accurately, periods at which the user applications are not being executed) and derive an optimal switching policy that minimizes this cost function.

Clearly, the optimal module switching policies for various structures of fault-tolerant computing systems are not necessarily the same. In this paper we concentrate on dynamic (standby) redundancy fault-tolerant systems. Similar policies for other configurations of fault-tolerant systems are being investigated.

A module replacement policy for dynamic redundancy systems has been presented in [7]. However, it has been assumed there that the switched-out module is considered faulty and discarded. The drawback of such a decision is that the premature retiring of a module which might still have a useful life service may unnecessarily reduce the system's mission time.

Consequently, we consider here the switched-out module as a possible spare which will be used, if and only if no "perfect" spare is available. Under this assumption the reasons against switching out the module (which is subject to frequent nonpermanent fault occurrences) are the possibility of a system failure during reconfiguration due to imperfect coverage (e.g., a fatal switching failure) and the overhead time associated with reconfiguration.

The optimal switching policy for a dynamic redundancy system is derived in Section III after the underlying mathematical model is outlined in Section II. Numerical examples are then presented in Section IV. Section V is devoted to possible extensions of our mathematical model and final conclusions are presented in Section VI.

II. The Mathematical Model

Consider a computing system, one of whose modules has m standby spares ready to be switched in upon failure of the active module. The active module is subject to fault occurrences and we adopt here the viewpoint that fault occurrences obey a Poisson process [4]–[9], [11]–[14]. Although the failure rate is treated as a constant in most related works, it is recognized that it may be a function of time and even some module parameters. It is apparent in particular, that as time goes on certain types of faults are more likely to occur and the status of the module may be deteriorating.

One way to model the deterioration process of modules is to assume that the failure rate increases with the occurrence of failures. The simplest model in this direction is one that distinguishes only between modules that had no failures, the failure rate of which is λ_0, and the modules that had at least one failure—whose failure rate is λ_1 ($> \lambda_0$). Restated, the failure rate of a module is initially λ_0 and after the first failure it increases to λ_1, and remains so thereafter. A module with failure rate λ_0 (λ_1) is said to be in state 0(1). This dichotomized deterioration model is analyzed in this work in detail. More general deterioration models are discussed in Section V.

When a failure occurs we have two possible actions at our disposal: we can retry the failed module in order to bring it

back to operation, or else replace it by a spare—if one is still available. The cost associated with a replacement, denoted by C_s, represents the loss due to system's down-time while the replacement operation is being executed. A replacement action could bring about a crash of the system, the probability of which event is denoted by s, in which case a penalty of C_f is paid for any unit of time in the uncompleted portion of the mission. If a crash does not occur, then the system resumes operation with the new active module while the module that has been switched out from operation is to be tested. This testing procedure, which does not result in any cost since no system down-time is involved, either completes successfully, if the fault is not permanent, or fails. In the former event, the probability of which we denote by $1 - p$, the retired module becomes a standby spare, while otherwise it is discarded and the group of spares is decreased by one.

The alternative operation, i.e., retrying the module that has failed, incurs cost C_r and has the following outcomes. Either a crash occurs, the probability of which is denoted by r, and a penalty of C_f is paid for any unit of uncompleted mission time. Or, a crash does not occur, then with probability $1 - p$ the retry operation proceeds and the system resumes operation, or else (with probability p) the fault is permanent, the retry operation fails and the failed module must be replaced. The costs and consequences of this forced replacement are as described above only that now the switched-out module is to be discarded right away (no further testing is needed), i.e., the size of the group of spares is necessarily decreased by one.

Obviously, when a replacement is made, either immediately or following an unsuccessful retry operation, the module chosen for replacement is one in state 0, i.e., an unused one, as long as this type of module has not been yet exhausted.

The decision whether to retry or replace a failed module depends on all the factors that have been mentioned above: the probabilities of crash s and r, the probability that a failure results in a permanent fault p, and the costs C_r, C_s, and C_f. Since the latter cost factor C_f, is per time unit, the decision also depends heavily on the remaining mission time. A crash closer to the end of the mission will incur less failure penalty.

III. Construction of the Optimal Policy

The dependence of the optimal action on the remaining mission time leads to a time-continuous dynamic programming formulation of the problem. First we note that the state space of the decision process at any time consists of N—the total number of spares ($N = 0, 1, 2, \cdots$), k—the number of spares in state 0 ($0 \le k \le N$), and the state i ($i = 0, 1$) of the active module. However, at a failure epoch before any action has been taken, the last component of the state space is redundant since by virtue of our dichotomized deterioration model, the state of the active module at failure must be 1.

Let $D^i_{k,N}(t)$ ($N = 0, 1, \cdots; k = 0, \cdots, N; i = 0, 1; t \ge 0$) be the minimal expected cost (of all the time periods at which the user applications are not serviced) at time t where N, k, and i are as defined above. The time t is measured backwards from the end of the mission. This convention has been adopted (throughout the paper) since only the remaining portion of the mission time affects our decision process. Consequently, this convention simplifies the expressions to follow and their analysis.

The optimality equation for $D^i_{k,N}(t)$ is given by

$$D^i_{k,N}(t) = \int_0^t \lambda_1 e^{-\lambda_1(t-u)} \min\{h^{(r)}_{k,N}(u);\ h^{(s)}_{k,N}(u)\}\, du$$

$$N = 1, 2, \cdots; k = 1, 2, \cdots, N; i = 0, 1 \qquad (1)$$

where

$$h^{(r)}_{k,N}(u) = C_r + rC_f u + (1-r)[(1-p)D^1_{k,N}(u) + p\{C_s + sC_f u + (1-s)D^0_{k-1,N-1}(u)\}]$$

is the expected cost in $(0, u)$ if there is a failure at u upon which a *retry* action is taken and afterwards an optimal policy is followed, and

$$h^{(s)}_{k,N}(u) = C_s + sC_f u + (1-s)[pD^0_{k-1,N-1}(u) + (1-p)D^0_{k-1,N}(u)]$$

is the expected cost in $(0, u)$ if there is a failure at u upon which a *replacement* action is taken and afterwards an optimal policy is followed.

The latter two expressions hold for $k \ge 1$, while for $k = 0$ (i.e., no spares at state 0 are available) we have

$$D^i_{0,N}(t) = \int_0^t \lambda_1 e^{-\lambda_1(t-u)} \min\{h^{(r)}_{0,N}(u);\ h^{(s)}_{0,N}(u)\}\, du \qquad (2)$$

where

$$h^{(r)}_{0,N} = C_r + rC_f u + (1-r)[(1-p)D^1_{0,N}(u) + p\{C_s + sC_f u + (1-s)D^1_{0,N-1}(u)\}]$$

and

$$h^{(s)}_{0,N} = C_s + sC_f u + (1-s)[pD^1_{0,N-1}(u) + (1-p)D^1_{0,N}(u)].$$

When $N = 0$, the only action available to us is a retry of the module. Hence,

$$D^1_{0,0}(t) = \int_0^t \lambda_1 e^{-\lambda_1(t-u)} \cdot \{C_r + rC_f u + (1-r)[(1-p)D^1_{0,0}(u) + pC_f u]\}\, du.$$

The solution of this integral equation is

$$D^1_{0,0}(t) = \frac{1}{\Phi_r}\left(C_r - \frac{C_f}{\lambda_1}\right)(1 - e^{-\lambda_1 \Phi_r t}) + C_f t \qquad (3)$$

where $\Phi_r = 1 - (1 - r)(1 - p)$.

For $D^0_{0,0}(t)$ we have,

$$D^0_{0,0}(t) = \int_0^t \lambda_0 e^{-\lambda_0(t-u)} \cdot \{C_r + [r + (1-r)p]C_f u + (1-r)(1-p)D^1_{0,0}(u)\}\, du.$$

Substituting (3) and integrating yields

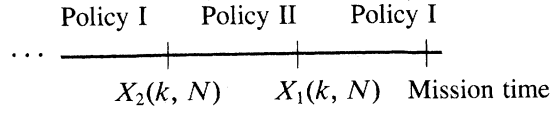
$$D^0_{0,0}(t)=\frac{1}{\Phi_r}\left[\left(C_r-\frac{C_f}{\lambda_1}\right)+\Phi_r\left(\frac{C_f}{\lambda_1}-\frac{C_f}{\lambda_0}\right)\right](1-e^{-\lambda_0 t})$$
$$+C_f t-\frac{(1-\Phi_r)\lambda_0}{(\lambda_0-\Phi_r\lambda_1)\Phi_r}\left(C_r-\frac{C_f}{\lambda_1}\right)(e^{-\Phi_r\lambda_1 t}-e^{-\lambda_0 t}). \quad (4)$$

The expressions in (3) and (4) provide us with the necessary starting points for the solution of the recursive optimality equation (2). Once the functions $D^i_{0,1}(t), \cdots, D^i_{0,N}$ have also been computed, they provide us, in turn, with the boundary functions for the solution of the recursive optimality equation (1).

We begin the solution process by stating two basic properties of the functions $D^i_{k,N}(t)$ that can be verified from the optimality equations, and the results in (3) and (4). For all functions $D^i_{k,N}(t)$, $N = 0, 1, \cdots; k = 0, \cdots, N; i = 0, 1$, we have (recall that time is always measured backwards),

i) $D^i_{k,N}(0)=0$
ii) $D^i_{k,N}(t)$ is continuously increasing in t. (5)

Since we have only two actions at failure, the optimal policy for any given (k, N) must alternate between these two actions. Consequently, there exists, for any given (k, N), a sequence of numbers $X_1(k, N), X_2(k, N), \cdots$ so that if a failure occurs in the time intervals $[X_{2n}(k, N), X_{2n+1}(k, N)]$, one of the actions is optimal. While if the failure occurs in the intervals $[X_{2n+1}(k, N), X_{2n+2}(k, N)]$ the other action is optimal ($n = 0, 1, \cdots$). This can be presented graphically as follows.



Once the values of these $X_j(k, N)$ are known, we may calculate the minimal expected cost $D^i_{k,N}(t)$. Thus, deriving an optimal policy for a given (k, N) is reduced to calculating a sequence of policy switching points $X_j(k, N)$, ($j = 1, 2, \cdots$). The number of these policy switching points is expected to be small as has been verified in many numerical examples. Therefore, the computational complexity of our scheme is substantially lower than that of the conventional dynamic programming approach where the cost function is usually calculated for every time-step (the size of which determines the accuracy of the final results).

The sequence of policy switching points $X_j(k, N)$, ($j = 1, 2, \cdots$) depends on the optimal order of the actions, i.e., which action corresponds to which set of intervals, but not on the state of the active module which is necessarily 1 upon failure. For some index j, $X_j(k, N)$ could be infinite implying that from that point on, the optimal action does not change any more.

To determine the optimal order of the actions we make the key observation that

$$h^{(r)}_{k,N}(0)<h^{(s)}_{k,N}(0), \text{ if and only if } C_r/C_s<1-p(1-r) \quad (6)$$

for all $N = 0, 1, \cdots; k = 0, 1, \cdots, N$.

Thus, due to (5), this simple check which does not require knowledge of the unknown functions, tells us which action comes first, i.e., it reveals the optimal order of the actions. Moreover, the optimal order is invariant with respect to k and N, a result that will prove to be of much use later.

The derivation of the optimal policy is detailed in the Appendix where closed-form expressions for the cost functions are derived. Having closed-form expressions simplifies the reevaluation of the cost function which might be needed whenever a module fails (and the system recovers from the effect of the failure) since some system parameters may change upon a failure, e.g., number of spares and state of spares or active modules.

IV. Numerical Examples

A software package (based on the IMSL double-precision library) for the calculation of the cost functions $D^i_{k,N}(t)$ and the policy switching points has been prepared. Using this package several examples have been analyzed, some of which are presented in the following.

The first example chosen for presentation is a system with a single standby spare which has, when the mission starts, a failure rate λ_0. The operational module which may be replaced by the spare one, has initially the same failure rate. Consequently, the cost function which is to be calculated is $D^0_{1,1}(t)$. However, its evaluation requires the calculation of $D^1_{1,1}(t)$, $D^0_{0,1}(t)$, $D^1_{0,1}(t)$, $D^0_{0,0}(t)$, and $D^1_{0,0}(t)$. These in turn, require the computation of two sets of policy switching points namely, $X_j(0, 1)$ and $X_j(1, 1)$.

In Fig. 1, four of these cost functions are depicted for a system with the following set of parameters.

Set I: $\lambda_1/\lambda_0 = 10$, $p = 0.05$, $r = 0.05$, $s = 0.03$, $C_s/C_r = 2$, $C_f = 10^3$.

In this figure time has been normalized and one time unit equals the average lifetime of a module with a failure rate λ_1. For the system given by Set I, a single policy switching point was found for $(k, N) = (0, 1)$ and similarly for $(k, N) = (1, 1)$. The computed values are $X_1(0, 1) * \lambda_1 = 0.432$ and $X_1(1, 1) * \lambda_1 = 0.256$. This means that for $(k, N) = (1, 1)$ the optimal policy calls for replacement of the operational module by the standby spare upon failure, up to $0.256/\lambda_1$ time units before the mission is over. From this point on the operational module will be retried upon failure.

The resulting optimal policy has an intuitive explanation. The cost ratio $C_s/C_r = 2$ indicates that the immediate cost incurred when replacing the operational module by the spare one is twice as high as that incurred when the module is retried. On the other hand, the probability of a system crash when a replacement is performed (s) is lower than that for a retry operation (r) giving preference to a replacement operation. When the mission is about to be over (t is small), we are mainly concerned about the immediate cost and consequently, we prefer to retry the module rather than replace it. However, when the remaining mission time is large, the long-term effect of a possible system crash is of more concern than the immediate cost. Therefore, we prefer replacement of the module over retrying it.

Expressions for the functions $D^0_{1,1}$ and $D^1_{1,1}$ were derived

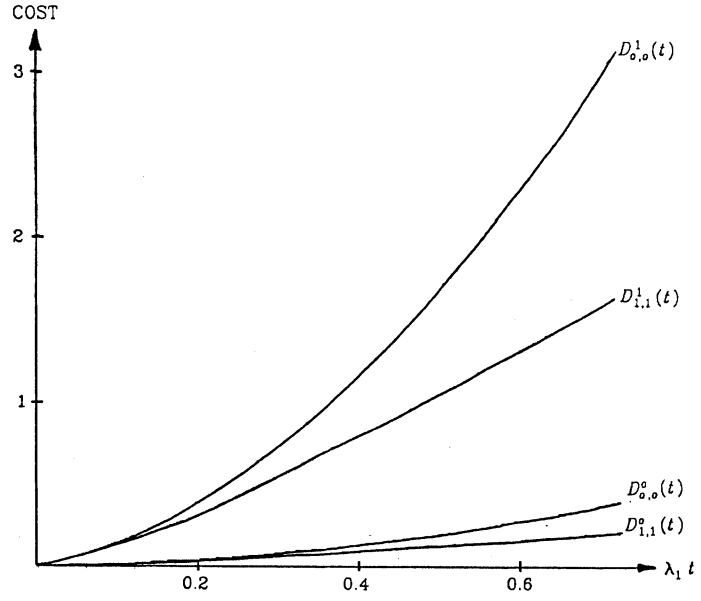


Fig. 1. Four system cost functions for Set I of system parameters.

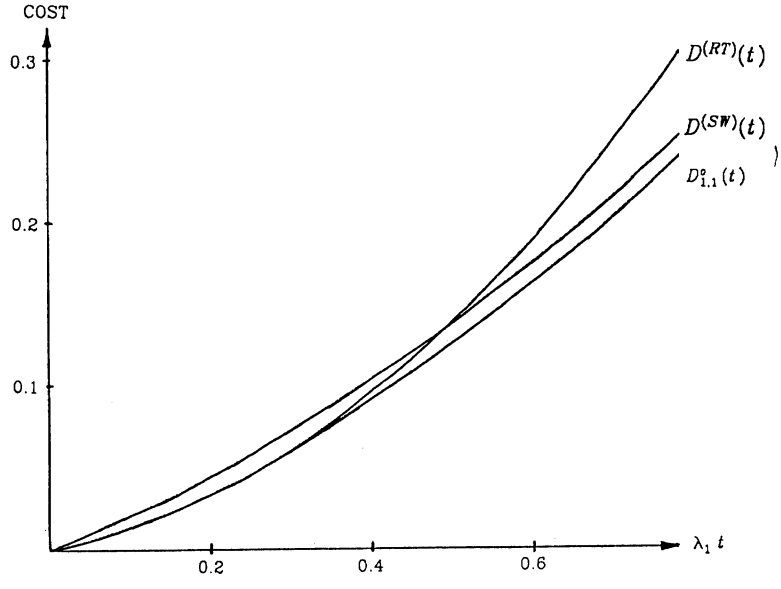


Fig. 2. The system cost (Set I of parameters) for optimal and nonoptimal policies.

and evaluated for the different time intervals and the resulting values are plotted in Fig. 1. It is interesting to note that in Fig. 1 $D^1_{1,1}(t) > D^0_{0,0}(t)$, i.e., for the given system parameters, having a single operational module (with no spare) with a failure rate λ_0 is preferable to having two modules (one in operation, the other a standby spare), both with a higher failure rate λ_1. The opposite has been observed in other examples.

In Fig. 2 the cost (in system down-time) incurred when the optimal policy is followed, is compared to the system cost for two other policies according to which the same decision (switch out or retry) is made throughout the mission time. In this figure $D^{(SW)}(t)$ denotes the cost if the operational module is always switched out and replaced upon failure. $D^{(RT)}(t)$ is defined similarly for a retry operation. The system cost associated with the optimal policy is clearly lower than those associated with the nonoptimal policies.

The improvement in system cost due to the adoption of the optimal policy is illustrated in Fig. 3. The improvement is defined by $1 - D^0_{1,1}(t)/D^{(SW)}(t)$ and similarly for $D^{(RT)}(t)$. For example, the cost is reduced by as much as 40 percent if instead of switching out the operational module (and replacing it by the spare) upon failure, one follows the optimal policy that calls for a retry operation when the mission is about to be over.

The deterioration model that we have employed requires the estimation of the failure rate ratio λ_1/λ_0. This ratio is clearly difficult to accurately estimate and, therefore, the sensitivity of the optimal policy to this ratio has to be analyzed. As an example of such a sensitivity analysis we have calculated the policy switching points for the above system parameters but with two different estimated failure rate ratios namely, $\lambda_1/\lambda_0 = 2$ and $\lambda_1/\lambda_0 = 15$. For each of these two values we have calculated the cost function $D^0_{1,1}(t)$ for a system with a failure

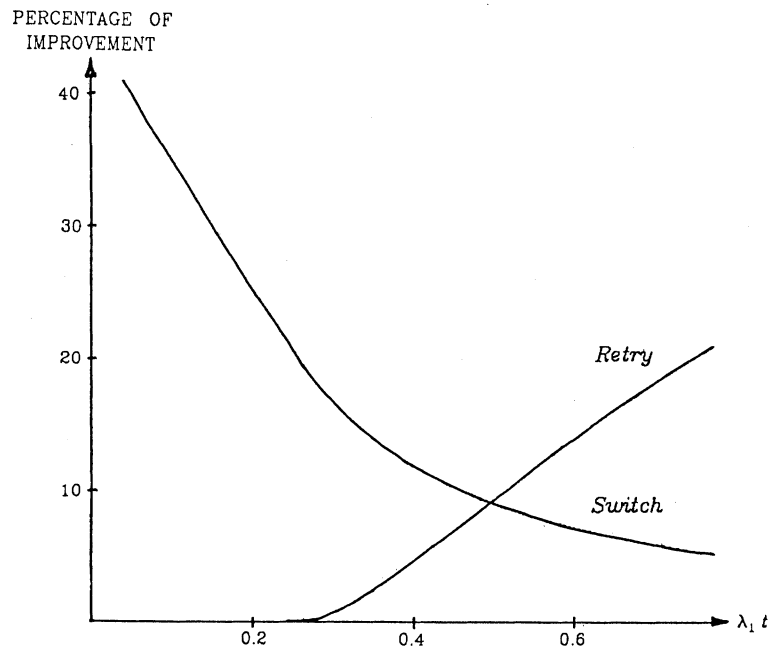


Fig. 3. Percentage of improvement in cost due to the optimal policy (Set I of parameters).

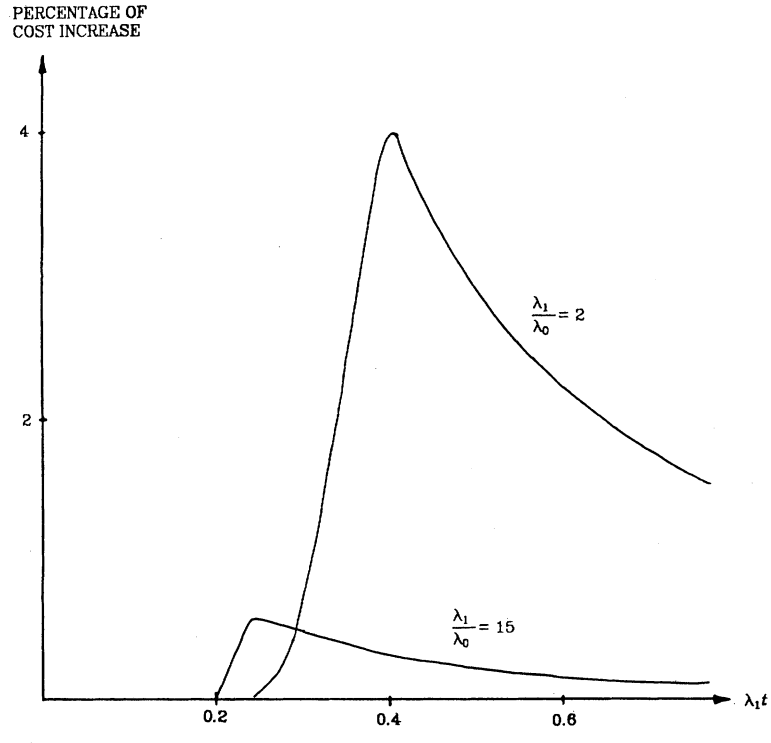


Fig. 4. Percentage of cost increase due to inaccurate estimation of the failure rate ratio.

rate ratio of $\lambda_1/\lambda_0 = 10$ (as before) but for which the policy switching points were precalculated based on an inaccurate estimation of the above ratio. The results are illustrated in Fig. 4 showing that the system cost increase due to inaccurate estimation of the failure rate ratio is almost negligible compared to the reduction in cost due to the adoption of the optimal policy (Fig. 3).

The above calculations were repeated for a second set of parameters, i.e.,

Set II: $\lambda_1/\lambda_0 = 10$, $p = 0.05$, $r = 0.01$, $s = 0.08$, $C_s/C_r = 0.5$, $C_f = 10^3$.

Here, two policy switching points were computed, $X_1(0, 1) * \lambda_1 = 0.162$ and $X_1(1, 1) * \lambda_1 = 0.202$, where the optimal policy calls for switching out of the operational module upon failure (and replacing it by the spare) in the time interval near the end of the mission. The reason for this decision is the lower immediate cost incurred when replacing the operational module (C_s) compared to the retry cost (C_r). When the

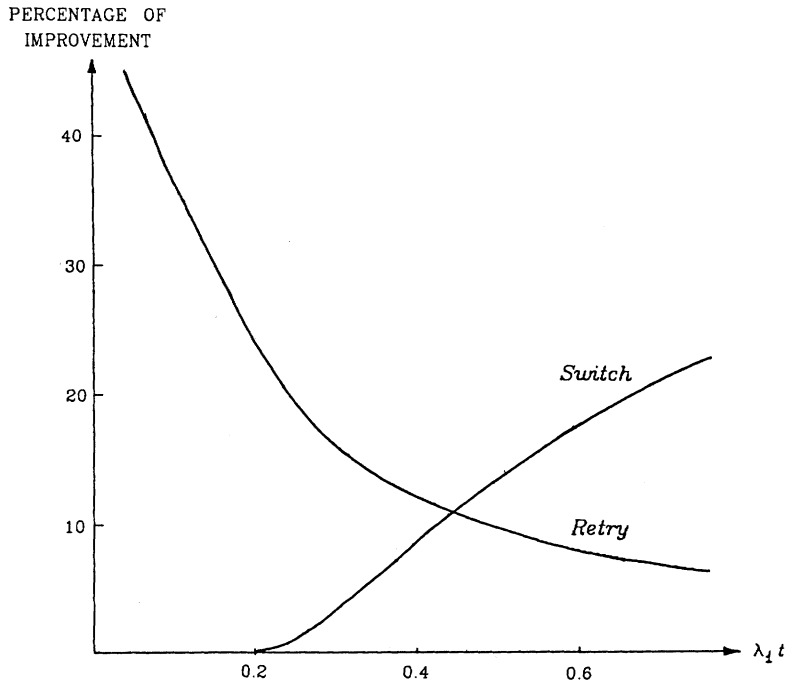


Fig. 5. Percentage of improvement in cost due to the optimal policy (Set II of parameters).

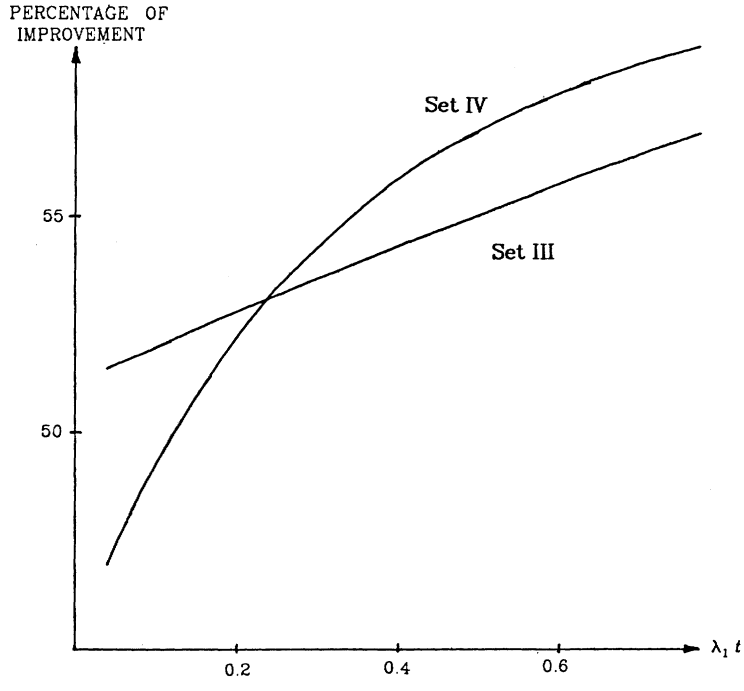


Fig. 6. Percentage of improvement in cost due to the optimal policy (Sets III and IV of parameters).

remaining mission time is large, the long-term cost of a system crash is of concern and, therefore, the optimal policy calls for a retry operation which has a lower probability of system crash (r). Fig. 5 depicts the improvement in system cost due to the adoption of the optimal policy. Here, too, the improvement is quite substantial illustrating the effectiveness of the optimal policy.

The importance of following the optimal policy is not restricted to cases where finite values for the policy switching points exist. Even in cases where the optimal policy calls for the same type of operation throughout the mission time [the operation is determined by (6)], one might gain by following this policy. In Fig. 6 two such cases are depicted. One has the same system parameters as in Set I except that $C_r/C_s = 0.5$, we call it Set III. The second, Set IV, is similar to Set II except that $C_s/C_r = 2$. For example, the lower curve in Fig. 6 shows the improvement in system cost if the optimal policy that calls for a replacement upon failure throughout the mission time is followed, instead of the nonoptimal policy of always retrying the failing module. Fig. 6 shows that cost reductions higher

than 50 percent might be expected when selecting the right policy.

In summary, the above presented numerical results illustrate the reductions in system cost one may expect when optimal module switching policies are applied.

V. More General Deterioration Models

The dichotomized deterioration model can be generalized in a natural way by assuming that the failure rate increases at every failure epoch and does so until the Mth failure after which it does not change any more. A module can, therefore, be in states: 0, 1, $\cdots$, M where its failure rate is λ_0, λ_1, $\cdots$, λ_M, respectively. Thus, λ_0 corresponds to the failure rate of modules that have not yet failed, λ_1 corresponds to modules that have failed exactly once, and so forth.

The mathematical approach that has been developed for the special case $M = 1$, i.e., the dichotomized deterioration model, can be readily extended to this more general situation. It, however, requires the enlargement of the state space to an $M + 1$ dimensional one since a counter for the number of spares in each of the possible $M + 1$ states is needed.

Specifically, define first a vector $k = (k_0, k_1, \cdots, k_{M-1}, N)$ where N is the total number of spares, k_i is the number of spares in state i ($i = 0, 1, \cdots, M-1$) and $N - \sum_{i=0}^{M-1} k_i \geq 0$ is the number of spares in state M. Then, define $D_k^i(t)$ as the optimal expected cost at time t (measured backwards) where the active module is in state i ($i = 0, 1, \cdots, M$) and the spares are in state k. We can again form a recursive integral equation for the function $D_k^i(t)$, recalling that if the operational module is switched out upon failure, the replacement module is taken from the subgroup of available modules with the smallest failure rate. As an illustration consider the case $M = 2$. The recursive functional equations are in this case

$$D^i_{k_0,k_1,N}(t) = \int_0^t \lambda_i e^{-\lambda_i(t-u)} * \min \left\{ \begin{matrix} h^{(r),i}_{k_0,k_1,N}(u) \\ h^{(s),i}_{k_0,k_1,N}(u) \end{matrix} \right\} du \tag{7}$$

where for $k_0 \geq 1$,

$$\begin{aligned} h^{(r),i}_{k_0,k_1,N}(u) &= C_r + (1-r)pC_s + [r + (1-r)ps]C_f u \\ &\quad + (1-r)(1-p)D^i_{k_0,k_1,N}(u) \\ &\quad + p(1-r)(1-s)D^0_{k_0,-1,k_1,N-1}(u). \end{aligned} \tag{8}$$

If $k_0 = 0$ and $k_1 \geq 0$, then the last term in (8) should be replaced by $D^1_{0,k_1-1,N-1}(u)$, and if $k_0 = 0$ and $k_1 = 0$ then it has to be replaced by $D^2_{0,0,N-1}(u)$.

For $h^{(s),i}_{k_0,k_1,N}(u)$ we have to distinguish between the cases $i = 0$ and $i = 1, 2$. For $i = 0$,

$$\begin{aligned} h^{(s),0}_{k_0,k_1,N}(u) &= C_s + sC_f u + (1-s)(1-p) \\ &\quad \cdot D^0_{k_0-1,k_1+1,N}(u) + p(1-s)D^0_{k_0-1,k_1,N-1}(u) \end{aligned} \tag{9}$$

and for $i = 1, 2$ we have,

$$\begin{aligned} h^{(s),i}_{k_0,k_1,N}(u) &= C_s + sC_f u + (1-s)(1-p)D^0_{k_0-1,k_1,N}(u) \\ &\quad + p(1-s)D^0_{k_0-1,k_1,N-1}(u). \end{aligned} \tag{10}$$

The last term in (9) and (10) should be replaced as was done for (8) if $k_0 = 0$ or $k_0 = k_1 = 0$. Similar changes should be done with the next to last term in these two equations.

The models considered hitherto may look in certain situations too rigid due to the assumption that the failure rate increases at every single failure. The following simple model extension allows much more flexibility in this aspect. We can add control parameters to the model and make the change of the failure rate, at the failure epoch, depend on the relation between these parameters and the time that has elapsed since the last failure of the module. The motivation behind this approach is that frequent failures indicate worsening of the module status, whereas sparsed failures may be interpreted as "normal" or even an indication of improvement (a Bayesian revision mechanism of the failure rate has similar complications although in a different conceptual framework).

To illustrate these ideas consider again the dichotomized model and assume that a module which had a failure rate λ_0 (λ_1) prior to the failure will have a change to failure rate λ_1 (λ_0) if the time that has elapsed since the last failure is less (more) than "a" ("b"); otherwise no change of failure rate occurs. For this simple model extension with two control parameters a and b we obtain the following functional expression.

$$\begin{aligned} D^i_{k,N}(t) &= \int_0^b \lambda_i e^{-\lambda_i(t-u)} * \min \left\{ \begin{matrix} h^{(r)}_{k,N}(u) \\ h^{(s)}_{k,N}(u) \end{matrix} \right\} du \\ &\quad + \int_b^t \lambda_i e^{-\lambda_i(t-u)} * \min \left\{ \begin{matrix} g^{(r)}_{k,N}(u) \\ g^{(s)}_{k,N}(u) \end{matrix} \right\} du. \end{aligned} \tag{11}$$

The expressions for $h^{(r)}_{k,N}(u)$ and $h^{(s)}_{k,N}(u)$ are the same as before, see (2), while $g^{(r)}_{k,N}(u)$ and $g^{(s)}_{k,N}(u)$ are given as follows,

$$\begin{aligned} g^{(r)}_{k,N}(u) &= C_r + (1-r)pC_s + [r + (1-r)ps]C_f u \\ &\quad + (1-r)(1-p)D^0_{k,N}(u) + p(1-r)(1-s)D^0_{k-1,N-1}(u) \end{aligned} \tag{12}$$

$$\begin{aligned} g^{(s)}_{k,N}(u) &= C_s + sC_f u + (1-s)(1-p)D^0_{k,N}(u) \\ &\quad + p(1-s)D^0_{k-1,N-1}(u). \end{aligned} \tag{13}$$

A close look at the sets of equations (7) and (11) reveals that they can be treated by the very same tools that were used for the solution of the dichotomized model. The solution of (11) can then be used to find the optimal replace or retry policy, as a function of a and b, as was illustrated for the basic dichotomized model.

VI. Conclusions

Optimal module switching policies have been introduced in the paper and one such policy has been developed for dynamic redundancy fault-tolerant computing systems. A dichotomized model for the deterioration process of active modules was adopted here. However, it has been shown that the algorithm for constructing the optimal policy can be extended so as to handle other, more general models for the deterioration process.

A software package for the calculation of the system cost functions and the determination of the policy switching points has been developed. The package was used for the numerical analysis of several examples, some of which were presented in this paper. These examples show that the application of the optimal switching policy might improve substantially the computational availability of a fault-tolerant system.

Similar optimal switching policies for other fault-tolerant architectures of computing systems like hybrid redundancy systems and gracefully degrading systems (e.g., [13]) should be developed.

Appendix

In this Appendix we derive the optimal policy assuming that (6) holds as it is (so that a retry action comes first). From the structure of (2) we may conclude that

$$D^1_{0,N}(t) = A_{0,N}(t) \qquad \text{for } 0 \le t \le X_1(0, N) \qquad \text{(A-1)}$$

where $A_{0,N}(t)$ is the solution of the integral equation,

$$D^1_{0,N}(t) = \int_0^t \lambda_1 e^{-\lambda_1 (t-u)} h^{(r)}_{0,N}(u)\, du. \qquad \text{(A-2)}$$

Solving (A-2) we obtain

$$A_{0,N}(t) = \frac{1}{\Phi_r}\left[C_r + (1-r)pC_s - \frac{\Psi}{\Phi_r}\frac{C_f}{\lambda_1}\right](1 - e^{-\lambda_1 \Phi_r t}) + \frac{\Psi}{\Phi_r} C_f t + \frac{p(1-r)(1-s)}{\Phi_r} \cdot \int_0^t \lambda_1 \Phi_r e^{-\lambda_1 \Phi_r (t-\tau)} D^1_{0,N-1}(\tau)\, d\tau \qquad \text{(A-3)}$$

where $\Psi = r + (1 - r)ps$.

Since $A_{0,N}(t)$ depends on $D^1_{0,N-1}(t)$ we have to carry out the solution procedure for successively increasing values of N. Starting with $N = 1$, we obtain from (A-3) after the substitution of $D^1_{0,0}(t)$ from (3),

$$A_{0,1}(t) = \frac{1}{\Phi_r}\left[\left(C_r - \frac{C_f}{\lambda_1}\right)\left(1 + \frac{p(1-r)(1-s)}{\Phi_r}\right) + (1-r)pC_s\right](1 - e^{-\lambda_1 \Phi_r t}) + C_f t - \lambda_1 \frac{p(1-r)(1-s)}{\Phi_r}\left(C_r - \frac{C_f}{\lambda_1}\right) te^{-\lambda_1 \Phi_r t}. \qquad \text{(A-4)}$$

Now define $\tilde{B}_{0,N}(v, x)$ as the expected cost in $[0, v]$ of a policy which prescribes a retry upon failure if it occurs in $[0, x]$ but a replacement if it occurs in $[x, v]$, $(0 < x < v)$. The active module at time x is assumed to be in state 1. This function can be expressed by means of the integral equation

$$\tilde{B}_{0,N}(v, x) = A_{0,N}(x)e^{-\lambda_1 (v-x)} + \int_x^v \lambda_1 e^{-\lambda_1 (t-u)} h^{(s)}_{0,N}(u)\, du$$

where in $h^{(s)}_{0,N}(u)$ we set $\tilde{B}_{0,N}(u, x)$ for $D^1_{0,N}(u)$.

The solution of this equation is given by

$$\tilde{B}_{0,N}(v, x) = A_{0,N}(x)e^{-\lambda_1 \Phi_s (v-x)} + \frac{1}{\Phi_s}\left[C_s - \frac{sC_f}{\lambda_1 \Phi_s}\right](1 - e^{-\lambda_1 \Phi_s (v-x)}) + \frac{sC_f}{\Phi_s}(v - xe^{-\lambda_1 \Phi_s (v-x)}) + \frac{p(1-s)}{\Phi_s}\int_x^v \lambda_1 \Phi_s e^{-\lambda_1 \Phi_s (v-u)} D^1_{0,N-1}(u)\, du \qquad \text{(A-5)}$$

where $\Phi_s = 1 - (1 - s)(1 - p)$.

To find the first policy switching point $X_1(0, N)$ we compute $\tilde{B}_{0,N}(v, x)$ for an increasing sequence of values of x, say $x = d, 2d, 3d, \cdots$ where v is set at $x + \epsilon$ (ϵ being very small). Let $x = md$ be the first time we have

$$\tilde{B}_{0,N}(x + \epsilon, x) < A_{0,N}(x + \epsilon).$$

Then $X_1(0, N)$ belongs by construction to the interval $[md, (m + 1)d]$. Subsequently, we may further subdivide this interval and reapply the same procedure to approach $X_1(0, N)$ as close as we wish.

If $X_1(0, N)$ is finite, we denote $B_{0,N}(t) = \tilde{B}_{0,N}(t, X_1(0, N))$ and obtain

$$D^1_{0,N}(t) = B_{0,N}(t) \qquad \text{for } X_1(0, N) < t < X_2(0, N). \qquad \text{(A-6)}$$

We now proceed to the computation of $X_2(0, N)$. To do that we define $\tilde{C}_{0,N}(v, x)$ as the expected cost if a retry is done upon failure if it occurs in $[0, X_1(0, N)]$, a replacement is done if the failure happens in $[X_1(0, N), x]$, and a retry is done if the failure occurs in $[x, v]$, where $X_1(0, N) < x < v$. This function is obtained by solving the integral equation

$$\tilde{C}_{0,N}(v, x) = B_{0,N}(x)e^{-\lambda_1 (v-x)} + \int_x^v \lambda_1 e^{-\lambda_1 (t-u)} h^{(r)}_{0,N}(u)\, du$$

where in $h^{(r)}_{0,N}(u)$ we set $\tilde{C}_{0,N}(u, x)$ for $D^1_{0,N}(u)$.

The solution of this equation is given by

$$\tilde{C}_{0,N}(v, x) = B_{0,N}(x)e^{-\lambda_1 \Phi_r (v-x)} + \frac{1}{\Phi_r}\left[C_r + (1-r)pC_s - \frac{\Psi}{\Phi_r}\frac{C_f}{\lambda_1}\right](1 - e^{-\lambda_1 \Phi_r (v-x)}) + \frac{\Psi}{\Phi_r} C_f (v - xe^{-\lambda_1 \Phi_r (v-x)}) + \frac{p(1-r)(1-s)}{\Phi_r}\int_x^v \lambda_1 \Phi_r e^{-\lambda_1 \Phi_r (v-u)} D^1_{0,N-1}(u)\, du. \qquad \text{(A-7)}$$

Again, by comparing $\tilde{C}_{0,N}(x + \epsilon, x)$ and $B_{0,N}(x + \epsilon)$ we determine $X_2(0, N)$. If the resulting policy switching point is finite, we denote $C_{0,N}(t) = \tilde{C}_{0,N}(t, X_2(0, N))$ and obtain,

$$D^1_{0,N}(t) = C_{0,N}(t) \qquad \text{for } X_2(0, N) < t < X_3(0, N). \qquad \text{(A-8)}$$

The above procedure with the necessary algebraic modifica-

tions, is repeated and as a result the sequence $X_j(0, N); j = 1, 2, \cdots$ is generated and in the process the function $D^1_{0,N}(t)$ is obtained.

Once $D^1_{0,N}(t)$ is known we can compute $D^0_{0,N}(t)$ by using the equation

$$D^0_{0,N}(t) = \int_0^t \lambda_0 e^{-\lambda_0(t-u)} \min \{h^{(r)}_{0,N}(u); h^{(s)}_{0,N}(u)\}\, du \quad \text{(A-9)}$$

whose validity rests on the fact that the sequence $X_j(0, N)$ is not affected by the change of the state, from 1 to 0, of the active module at time t. Under our current assumption that (6) holds, $h^{(r)}_{0,N}(u)$ is selected in the interval $[0, X_1(0, N)]$, $h^{(s)}_{0,N}(u)$ is selected in the interval $[X_1(0, N), X_2(0, N)]$ and so on.

We start with the calculation of $D^1_{0,1}(t)$ and $D^0_{0,1}(t)$ and continue to $N = 2$ by employing a similar procedure to generate the sequence $X_1(0, 2), X_2(0, 2), \cdots$ and compute $D^1_{0,2}(t)$ and $D^0_{0,2}(t)$. The procedure is then repeated for increasing values of N as far as desired.

Next, we consider again (1). First we need to solve the general integral equation

$$D^1_{k,N}(t) = \int_0^t \lambda_1 e^{-\lambda_1(t-u)} h^{(r)}_{k,N}(u)\, du \qquad N = 1, 2, \cdots; k = 1, 2, \cdots, N. \quad \text{(A-10)}$$

The solution of this equation is given by

$$A_{k,N}(t) = \frac{1}{\Phi_r}\left[C_r + (1-r)pC_s - \frac{\Psi}{\Phi_r}\frac{C_f}{\lambda_1}\right](1 - e^{-\lambda_1\Phi_r t}) + \frac{\Psi}{\Phi_r} C_f t + \frac{p(1-r)(1-s)}{\Phi_r} \cdot \int_0^t \lambda_1\Phi_r e^{-\lambda_1\Phi_r(t-\tau)} D^0_{k-1,N-1}(u)\, du. \quad \text{(A-11)}$$

The dependence of (A-11) on $D^0_{k-1,N-1}(t)$ leads to a recursive solution procedure and starting with $N = k = 1$ we obtain, after substituting $D^0_{0,0}(u)$ from (4)

$$A_{1,1}(t) = \frac{1}{\Phi_r}\left\{(1-r)pC_s + \left(C_r - \frac{C_f}{\lambda_1}\right)\left(1 + \frac{p(1-r)(1-s)}{\Phi_r}\right) + p(1-r)(1-s)\left(\frac{C_f}{\lambda_1} - \frac{C_f}{\lambda_0}\right)\right\} * (1 - e^{-\lambda_1\Phi_r t}) + C_f t - \frac{(1-\Phi_r)\lambda_0\lambda_1}{\lambda_0 - \Phi_r\lambda_1}\left(C_r - \frac{C_f}{\lambda_1}\right)\frac{p(1-r)(1-s)}{\Phi_r} te^{-\lambda_1\Phi_r t} + \frac{p(1-r)(1-s)}{\Phi_r}\frac{\lambda_1\Phi_r}{\lambda_0 - \lambda_1\Phi_r}\left\{\frac{1}{\phi_r}\left(C_r - \frac{C_f}{\lambda_1}\right)\left[1 - \frac{(1-\Phi_r)\lambda_0}{\lambda_0 - \Phi_r\lambda_1}\right] + \left(\frac{C_f}{\lambda_1} - \frac{C_f}{\lambda_0}\right)\right\}(e^{-\lambda_0 t} - e^{-\lambda_1\Phi_r t}). \quad \text{(A-12)}$$

Then

$$D^1_{1,1}(t) = A_{1,1}(t) \qquad \text{for } 0 \le t \le X_1(1, 1). \quad \text{(A-13)}$$

To obtain $X_1(k, N)$ we, as for $k = 0$, define $\tilde{B}_{k,N}(v, x)$ as

$$\tilde{B}_{k,N}(v, x) = A_{k,N}(x)e^{-\lambda_1(v-x)} + \int_x^v \lambda_1 e^{-\lambda_1(t-u)} h^{(s)}_{k,N}(u)\, du = A_{k,N}(x)e^{-\lambda_1(v-x)} + \left[C_s - \frac{sC_f}{\lambda_1}\right](1 - e^{-\lambda_1(v-x)}) + sC_f(v - xe^{\lambda_1(v-x)}) + \int_x^v \lambda_1 e^{-\lambda_1(t-u)}[(1-\Phi_s) \cdot D^0_{k-1,N}(u) + p(1-s)D^0_{k-1,N-1}(u)]\, du. \quad \text{(A-14)}$$

Here, as opposed to the case of $k = 0$, no equation has to be solved. Instead, we have to substitute the known by then expressions for $D^0_{k-1,N}(u)$ and $D^0_{k-1,N-1}(u)$ and integrate. By comparing $\tilde{B}_{k,N}(x + \epsilon, x)$ to $A_{k,N}(x + \epsilon)$, as was done for $k = 0$, one finds the policy switching point $X_1(k, N)$. If $X_1(k, N)$ is finite, we denote $B_{k,N}(t) = \tilde{B}_{k,N}(t, X_1(k, N))$ and proceed to compute $X_2(k, N)$. We define, as for $k = 0$,

$$\tilde{C}_{k,N}(v, x) = B_{k,N}(x)e^{-\lambda_1\Phi_r(v-x)} + \frac{1}{\Phi_r}\left[C_r + (1-r)pC_s - \frac{\Psi}{\Phi_r}\frac{C_f}{\lambda_1}\right](1 - e^{-\lambda_1\Phi_r(v-x)}) + \frac{\Psi}{\Phi_r} C_f(v - xe^{-\lambda_1\Phi_r(v-x)}) + \frac{p(1-r)(1-s)}{\Phi_r} \cdot \int_x^v \lambda_1\Phi_r e^{-\lambda_1\Phi_r(v-u)} D^0_{k-1,N-1}(u)\, du \quad \text{(A-15)}$$

and compare the resulting expression to $B_{k,N}$ to determine $X_2(k, N)$.

Repeating the very same procedure, with the necessary algebraic modifications, we generate the optimal policy switching points and compute the functions $D^i_{k,N}(t)$ ($N = 1, 2, \cdots; k = 1, \cdots, N$) as far as needed. The order of the computation is $D^i_{1,2}(t)$, $D^i_{2,2}(t)$, $D^i_{1,3}(t)$, $D^i_{2,3}(t)$, $D^i_{3,3}(t)$ $\cdots$ where each time we first treat the case $i = 1$ and then $i = 0$.

The above derivation was under the assumption that (6) holds. It is easy to observe that essentially the same procedure can also be applied when the reverse of (6) is true, i.e., replacement precedes retry in the optimal order of actions.

References

[1] T. Anderson and P. A. Lee, *Fault-Tolerance, Principles ana Practice.* Englewood Cliffs, NJ: Prentice-Hall, 1981.

[2] A. Avizienis, "Architecture of fault-tolerant computing systems," in *Dig. 5th Int. Symp. Fault-Tolerant Comp.*, June 1975, pp. 3-16.

[3] M. D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Comput.*, vol. C-27, pp. 540-547, June 1978.

[4] A. Costes, C. Landrault, and J. C. Laprie, "Reliability and availability models for maintained systems featuring hardware failures and design faults," *IEEE Trans. Comput.*, vol. C-27, pp. 548-559, June 1978.

[5] A. L. Hopkins, Jr., T. B. Smith, III, and J. H. Lala, "FTMP—A

highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE,* vol. 66, pp. 1221–1239, Oct. 1978.

[6] H. Ihara *et al.,* "Fault-tolerant computer system with three symmetric computers," *Proc. IEEE,* vol. 66, pp. 1160–1177, Oct. 1978.

[7] I. Koren and M. Berg, "A module replacement policy for dynamic redundancy fault-tolerant computing systems," in *Dig. 11th Int. Symp. Fault-Tolerant Comput.,* June 1981, pp. 90–95.

[8] I. Koren, Z. Koren, and S. Y. H. Su, "Analysis of a class of recovery procedures," *IEEE Trans. Comput.,* vol. C-35, pp. 703–712, Aug. 1986.

[9] I. Koren and S. Y. H. Su, "Reliability analysis of *N*-modular redundancy systems with intermittent and permanent faults," *IEEE Trans. Comput.,* vol. C-28, pp. 514–520, July 1979.

[10] S. R. McConnel, D. P. Siewiorek, and M. M. Tsao, "The measurement and analysis of transient errors in digital computer systems," in *Dig. 9th Int. Symp. Fault-Tolerant Comput.,* June 1979, pp. 67–69.

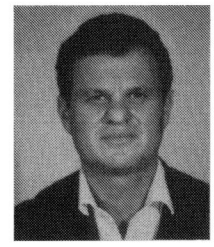
[11] Y. W. Ng and A. Avizienis, "A model for transient and permanent fault recovery in closed fault-tolerant systems," in *Dig. 6th Int. Symp. Fault-Tolerant Comput.,* June 1976, pp. 182–188.

[12] D. P. Siewiorek *et al.,* "A case study of C.mmp, Cm *, and C.vmp: Part I—Experiences with fault tolerance in multiprocessor systems," *Proc. IEEE,* vol. 66, pp. 1178–1199, Oct. 1978.

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

[14] S. Y. H. Su, I. Koren, and Y. K. Malaiya, "A continuous parameter Markov model and detection procedures for intermittent faults," *IEEE Trans. Comput.,* vol. C-27, pp. 567–570, June 1978.

[15] W. N. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE,* vol. 66, pp. 1126–1145, Oct. 1978.



Menachem Berg received the M.Sc. and Ph.D. degrees in operations research from the Technion–Israel Institute of Technology, Haifa, and the B.Sc. degree in statistics and mathematics from the Hebrew University, Jerusalem, Israel.

He is a Senior Lecturer in the University of Haifa and currently on a visiting appointment in the University of Toronto. His previous visiting appointments were in the University of California at Berkeley, Georgia Institute of Technology, University of Illinois at Chicago, University of British Columbia, Vancouver, and University of Sussex, Brighton, U.K. His major areas of interest are reliability theory and maintenance policies. He has published papers on general reliability modeling, computing fault-tolerance, power system reliability, spares provisioning, and replacement and maintenance policies. Currently he is working on problems within the above topics as well as on software reliability and reliability growth analysis. He also has interest in applications of Bayesian decision and inference procedures in reliability theory.

Israel Koren (S'72–M'76–SM'87), for a photograph and biography, see this issue, p. 1029.