Threshold-Based Mechanisms to Discriminate Transient from Intermittent Faults

Andrea Bondavalli, *Member*, *IEEE*, Silvano Chiaradonna,
Felicita Di Giandomenico, and Fabrizio Grandoni

Abstract—This paper presents a class of count-and-threshold mechanisms, collectively named α-count, which are able to discriminate between transient faults and intermittent faults in computing systems. For many years, commercial systems have been using transient fault discrimination via threshold-based techniques. We aim to contribute to the utility of count-and-threshold schemes, by exploring their effects on the system. We adopt a mathematically defined structure, which is simple enough to analyze by standard tools. α-count is equipped with internal parameters that can be tuned to suit environmental variables (such as transient fault rate, intermittent fault occurrence patterns). We carried out an extensive behavior analysis for two versions of the count-and-threshold scheme, assuming, first, exponentially distributed fault occurrencies and, then, more realistic fault patterns.

Index Terms—Fault discrimination, threshold-based identification, transient and intermittent faults, modeling and evaluation, fault diagnosis.

1 Introduction

Most computer system applications are expected to have a "reasonably" continuous service, ranging from a few minutes' wait after a crash, to a few seconds/milliseconds downtime as often required in controllers of critical systems. This means that, when a fault occurs, some action has to be taken to get the system operating again. The extent of such actions depends on the nature of the fault and on the characteristics of the system. In [1], [2], physical faults are classified as permanent or temporary. Permanent faults lead to errors whenever the relevant component is activated; the only way to handle such faults is to remove the component. Temporary physical faults can be distinguished into internal faults (usually known as intermittent) and external faults (transient). The former are caused by some internal part going outside its specified behavior. After their first appearance, they usually exhibit a relatively high occurrence rate and, eventually, tend to become permanent. On the other hand, transient faults cannot be traced to a defect in a particular part of the system and, normally, their adverse effects rapidly disappear. Since most malfunctions derive from transient faults [2], if they do not occur too frequently, removing the affected component would not be the best solution for most systems. Trying to keep operative a component that has been hit by transient faults is particularly worthwhile in the design of unattended dependable systems (e.g., aerospace applications), where the components should be kept in the system until the throughput benefit of keeping the faulty component on-line is offset by the greater probability of multiple (hence, catastrophic) faults. In any case, the effects of the faults need to be dealt with: Simply retrying or restarting may be enough in commercial systems, while, in more demanding applications, on-line error masking and/or a state recovery of the faulty component before restart may be required.

The importance of distinguishing transient faults so that they can be dealt with specifically is testified to by the wide range of solutions proposed, although with reference to specific systems (see Section 6). One commonly used method, for example, in several IBM mainframes, is to count the number of error events: Too many events in a given time frame would signal that the component needs to be removed.

This paper presents a family of mechanisms based on a count-and-threshold scheme, collectively named α-count, designed to discriminate intermittent and permanent faults against low rate, low persistency transient faults. Most similar solutions in the literature apply to general purpose mainframe systems, well-equipped with system crash logs, which give detailed information for use in (complex) postmortem analysis. Our goal is to develop mechanisms simple enough to 1) be studied in depth by standard analytical means and 2) be implementable as small, low-overhead and low-cost modules, suitable even for embedded, real-time, dependable systems. The behavior of a basic version of α-count is presented in [3] with reference to a fault model restricted to the exponential distribution of fault occurrences. We now present a single-threshold scheme and a better performing double threshold scheme. To characterize the latter's enhanced behavior, more significant performance parameters have been used. We perform a two step analysis: First, we make the usual assumptions about exponential fault distribution; then, we

- *A. Bondavelli is with the Department of systems and Information Science, University of Firenze, Via Lombroso 6/17, Firenze, Italy. E-mail: a.bondavalli@dsi.unifi.it.*
- *S. Chiaradonna is with CNUCE-CNR, Via Alfieri No. 1, 56010 Ghezzano-Pisa, Italy. E-mail: S.Chiaradonna@guest.cnuce.cnr.it.*
- *F. Di Giandomenico and F. Grandoni are with IEI-CNR, Via Alfieri No. 1, 56010 Ghezzano-Pisa, Italy. E-mail: {digiandomenico, grandoni}@iei.pi.cnr.it.*

Manuscript received 23 Nov. 1998; accepted 5 Oct. 1999.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 106368.

make more accurate fault hypotheses to better represent the real-life behavior of intermittent and transient faults. The extended fault hypotheses increased the model's analytical aspects, without impairing the easy analyzability of the mechanism.

The rest of the paper is organized as follows: Section 2 introduces the general model of the system, together with hypotheses on the component's behavior and the performance figures for the fault discrimination mechanism. In Section 3, the basic single-threshold scheme is presented and analyzed under the simplistic exponentially distributed faults assumption. Section 4 is devoted to modeling and evaluating the basic scheme under more realistic distributions of fault occurrences. Section 5 introduces and analyzes a double-threshold scheme aimed at facilitating a trade-off between conflicting goals in the design of the basic scheme. In Section 6, the literature is briefly reviewed and some comparisons with heuristics shown. Finally, conclusions are drawn in Section 7.

2 System Model

The description of our mechanisms will be given with reference to a model, where the system is partitioned into components enclosed in well-defined boundaries (which constitute the field replaceable units of the system), thus establishing the finest resolution for error detection and fault diagnosis. Besides the fault discrimination mechanism, the system includes:

- An *error detection subsystem* which evaluates whether a component is behaving correctly or not. Error detection results in signals, which for the purposes of this paper are assumed to have binary values. We use this signal stream to feed the fault discrimination mechanism. Computing systems are normally equipped with several low-level error detection devices (e.g., overflow checking, divide-by-zero, program code signatures, watchdog timers). Error notification may also be generated by testing programs or be a subproduct of other mechanisms the system is equipped with (e.g., of the error processing mechanisms properly geared with supplemental error signaling tools or of distributed agreement algorithms augmented with disagreement indications).
- A *fault passivation subsystem*, which is in charge of processing the signals produced by the fault discrimination mechanism.

Hereafter, it will be assumed that, when an error in a system component is detected, the effect of the error is removed by the time the component is used again. This obviously raises the issue of the cost of such an assumption. Error treatment itself (or its cost) can be neglected in some cases (e.g., when stateless components are used or in redundant computations where the entire state of the components is subject to a vote and the correct value can be recovered from the result of this vote), but may be very onerous in other cases.

2.1 Basic Assumptions

The analytical modeling and quantitative evaluations of the fault discrimination mechanisms introduced in the following sections will be carried out in two phases: 1) The conventional assumption of exponential fault distribution is made, then 2) more realistic scenarios, including increasing rates of intermittent fault manifestations and bursty transient fault intervals, will be considered.

Unless specified otherwise, all reference will be made to a generic system component. Below are some general assumptions relating to phase 1):

1. All fault-related events (error detector triggering, completion of testing routines, etc.) occur at discrete points in time, described by a discrete variable, L; two successive points in time differ by a (constant) *time unit* (or step). For each component, a (binary) signal $J^{(L)}$ is issued at step L: $J^{(L)} = 0$, meaning that no error has been detected, or $J^{(L)} = 1$, if an error was revealed.
2. Faults affecting the hardware directly supporting the discrimination mechanisms are not considered.
3. The signal $J^{(L)}$ on the component's behavior is correct with a probability c, which accounts for the coverage of the error detection mechanism used.
4. System components may be affected by permanent, intermittent, or transient faults. During each time unit, a component may be affected at most by a single transient fault and/or by a single internal fault, permanent or intermittent.
5. Permanent and intermittent faults are exponentially distributed. Therefore, the probabilities of their occurrence in a time unit are constant and will be denoted by q_p and q_i, respectively.
6. A component affected by a permanent fault gives rise to an error at each step; an intermittent fault repeatedly causes errors after the fault occurrence with constant probability q at each step.
7. A transient fault lasts only for one step. This is realistic if the duration of transient faults is small compared to that of the step. Transient faults are exponentially distributed, with a constant probability of occurrence q_t in a generic time unit.

The more realistic fault scenario examined in phase 2) is defined by keeping assumptions 1-5 above, but substituting the following for assumptions 6 and 7:

6.′ A component affected by a permanent fault gives rise to an error at each step; an intermittent fault repeatedly causes errors in the steps following the fault occurrence with a probability expressed by a generic function $q(d)$, where d is a discrete time variable originating at the fault occurrence.

7.′ A transient fault still only lasts for one time unit. Transient faults, while maintaining their exponential time distribution, are characterized by the alternation of periods where the *normal* fault rate is observed, and periods with an *abnormal*, higher rate. The duration of a period is assumed to follow an exponential law (with normal periods quite a bit longer than abnormal ones).

TABLE 1
Symbols and Definitions

Symbol	Definition
α	score associated with each not-yet-removed component
XD	total number of time steps between a permanent or intermittent fault occurrence and its signalling
D	average of XD
F_D	Cumulative Distribution Function of XD
NU	fraction of component life in which an healthy component u is not effectively used in the system.
q_t	probability of transient fault per time-unit
$q_p + q_i$	probability of intermittent or permanent fault per time unit
c	probability that a signal on the components behaviour is correct
K	the ratio by which α is decreased upon receiving $J^{(L)}=0$
α_T	threshold used in the single-threshold scheme to identify a component as affected by a permanent or intermittent fault
α_H	higher threshold used in the double-threshold scheme to identify a component as affected by a permanent or intermittent fault
α_L	lower threshold used in the double-threshold scheme to keep a component in full service

2.2 Figures of Interest and Notation Summary

Informally, our fault discrimination mechanisms aim to balance two conflicting requirements:

1. To signal, as quickly as possible, all components affected by permanent or intermittent faults (hereafter *faulty components* or *FCs*). In fact, gathering information to discriminate between transient and intermittent faults takes time, thus giving rise to a longer fault latency. This increases the chances of catastrophic failure and also increases the requirements on the error processing subsystem in fault-tolerant systems.
2. To avoid signaling components other than *FC*s (hereafter, *healthy components* or *HCs*). Depriving the system of resources that can still do valuable work may be even worse than relying on a faulty component.

The performance of the mechanisms with respect to the above goals will be evaluated through:

1. the average D or the Cumulative Distribution Function (CDF) FD of XD, the total elapsed time steps between a (permanent or intermittent) fault occurrence in a component u and the signaling of u as faulty; in this time interval, u is maintained in use and relied upon.
2. NU, the fraction of component life (with respect to its expected life as determined by the expected occurrence of a permanent/intermittent fault) in which a healthy component u is not actually used in the system. NU is a measure of the penalty inflicted by the discrimination mechanism on the utilization of *HC*s.

Table 1 shows the symbols used for the main parameters, together with their definitions. The default values for q_t, $q_p + q_i$ are derived by considering the case when the module to be diagnosed is a processor or computer executing a cyclic task at a repetition rate of 100 times per hour. The rate for intermittent-permanent faults is assumed to be 10^{-4} faults/hour, thus $q_p + q_i = 10^{-4}/100 = 10^{-6}$. We assume that transient faults are 10 times more frequent than permanent faults, i.e., $q_t = 10^{-5}$. The probability that the error detection subsystem gives wrong signals is assumed to be no worse than that of transient faults as, otherwise, there would be no gain in the dependability of the overall system. The default value of the parameter c is thus set to $1 - 10^{-5}$.

3 A Basic Threshold-Based Filtering Mechanism and Its Analysis

3.1 α-Count, an On-Line Single Threshold-Based Fault Identification Mechanism

As in several well-known threshold schemes (see Section 6 for a brief review), our first basic mechanism, the *single-threshold* α-count, gathers all signals related to the correctness of a given component from any available error detector, weighing down signals as they get older, to decide the point in time when keeping a system component on-line is no longer beneficial.

A simple, but nevertheless satisfactory, formulation of the filtering function α is the following:

$$\alpha^{(L)} = \begin{cases} \alpha^{(L-1)} \cdot K & \text{if} \quad J^{(L)} = 0 \\ \alpha^{(L-1)} + 1 & \text{if} \quad J^{(L)} = 1 \end{cases} \qquad 0 \le K \le 1$$
$$\alpha^{(0)} = 0, \tag{1}$$

where L is the discrete time variable defined in Section 2.1 and α is a score associated with each not-yet-removed component to keep track of the errors experienced by that component.

When the value of $\alpha^{(L)}$ exceeds a given threshold α_T, the component is diagnosed as affected by a permanent or an intermittent fault. This event produces an α-signal, which is sent to the fault passivation subsystem. The values to assign to the parameters K and α_T so that the strategy works best depend on the expected frequency of permanent, intermittent, and transient faults and on the probability c of correct judgments of the error signaling mechanism used. Note that $\lceil \alpha_T \rceil$ represents the minimum number of consecutive errors sufficient to consider a component as bad enough for removal, while K represents the ratio in which α is decreased after a time step without error signals, thus setting the time window where memory of previous errors is retained.

The filtering function α may be given different formulations from (1) above when searching for particular performances. For example, the additive expression below associates a constant decrement to each errorless computation, while, in (1), the decrement is proportional to the current count.

$$\alpha^{(L)} = \begin{cases} \alpha^{(L-1)} + 1 & \text{if} \quad J^{(L)} = 1 \\ \alpha^{(L-1)} - dec & \text{if} \quad \begin{array}{l} J^{(L)} = 0 \;\; AND \;\; \alpha^{(L-1)} - dec > 0, \\ 0 \leq dec \end{array} \\ 0 & \text{if} \quad J^{(L)} = 0 \;\; AND \;\; \alpha^{(L-1)} - dec \leq 0. \end{cases} \tag{2}$$

With this function, the decay time of the current value is proportional to the value itself, while, in form (1), it depends (roughly) only on the parameter K. This would result in different optimal settings for the mechanism's parameters and also in differences in behavior; however, the modeling and the analysis will be worked out for form (1) only since it is straightforward to adapt the methods used here to form (2) and, of course, to other variants of the α function as well.

3.2 Analysis of the Single-Threshold α-Count

The first phase of the evaluation of the single-threshold α-count is now carried out, using hypotheses 1 to 7 (Section 2.1). Because of the assumption of exponential fault distributions, it is trivial to see that an FC will always eventually be identified as such. Unfortunately, the same argument means that we cannot rule out that an HC might be wrongly signaled as faulty. Besides these asymptotic results, the behavior of the mechanism has been modeled by Stochastic Activity Networks (SAN) [4]. Two SANs have been derived: The first, modeling the mechanism acting on a faulty component, is used to compute an approximation of the average delay D to the identification of an FC; the second, modeling the mechanism acting on a healthy component, allows the approximate evaluation of the average unused fraction of an HC life NU. A faulty component is affected, by definition, either by a permanent or by an intermittent fault. The estimation of the number of steps needed to recognize a permanent fault is trivial: The probability of $J^{(L)} = 1$ is very close to 1 (it is $1 \cdot c$), therefore, the expected time to the threshold crossing is approximately bounded by $\lceil \alpha_T \rceil$, considering that the value of α will generally be greater than 0 when the permanent fault occurs. Hence, we have only modeled the case of intermittent faults, which requires a longer identification time. As required by SAN models, a discrete approximation of the real valued variable α is used. In the following sections, the approximate values are chosen to lead to a conservative evaluation of the performance parameters.

The SAN models are presented in the next two subsections and evaluated in Section 3.3. As for the models presented later, the details of the modeling are not necessarily a prerequisite for appreciating the results of subsequent evaluations.

3.2.1 Model of the Single-Threshold α-Count Acting on a Faulty Component

F_SAN-1 (Fig. 1) represents the behavior of the single-threshold α-count acting on the FC u when affected by an intermittent fault. Inside F_SAN-1, α is represented by its approximation variable $\alpha\_inf \leq \alpha$; α_inf will then reach α_T in an average number of steps D higher than, or equal to, the average number of steps which would be required by the real α. Moreover, to better enforce D to be an upper bound, we assume that, at the first occurrence of a permanent/intermittent fault, $\alpha\_inf = 0$, while, in general, α_inf will take positive, albeit small, values.

F_SAN-1 has three places: 1) not_remov (a token here means that u is not yet flagged as faulty); 2) *remov* (a token here marks the actual identification of u as an FC, with the attending emission of the α-signal); 3) $alpha\_state$ (used to maintain the current value of α_inf). The exponentially distributed timed activity (with rate 1) $next\_step$ represents the receipt of a signal; it has two cases: *case 1*, representing the occurrence of $J^{(L)} = 1$, and *case 2*, representing that of $J^{(L)} = 0$.

In order to allow the steady-state evaluation of the SAN, an additional timed activity, $next\_miss$, has been inserted. It represents the (dummy) restart of α-count upon the issuing of the α-signal; $next\_miss$ is also exponentially distributed with rate 1. The two output gates *incr* and *decr* perform the basic steps for computing α_inf. When α_inf reaches the value α_T, the gate *incr* deposits a token in the place *remov* and sets to 0 the value of α_inf in order to restart in $next\_miss$.

The probabilities associated with *case 1* and *case 2* in the $next\_step$ transition are expressed as:

$$P(Case\ 1) = (q + q_t)c + (1 - q - q_t)(1 - c)$$
$$P(Case\ 2) = (q + q_t)(1 - c) + (1 - q - q_t)c.$$

The first term of the first expression represents a failure of u (either an activation of the intermittent or an occurrence of a transient fault, with probability $q + q_t$) which is detected by the error signaling mechanism (with probability c); while the second term represents the incorrect detection of a failure. The second expression has a symmetric explanation.

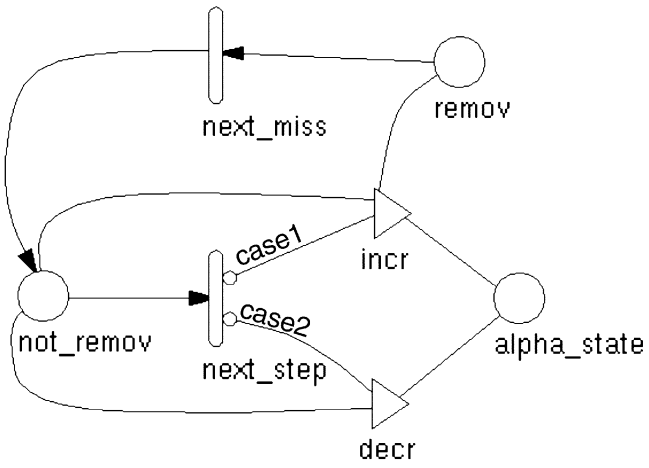


Fig. 1. F_SAN-1: Single-threshold α-count acting on a Faulty Component with exponential fault occurrence.

The value of D is determined by the ratio between the steady-state probabilities that one token is in the places not_remov and *remov*, respectively, which is equal to the ratio between the mean time spent in the two places. Since the activities $next\_step$ and $next\_miss$ have the same rate (equal to 1), the latter ratio corresponds to the average number of visits to the place not_remov before going to the place *remov*, i.e., the average number of steps to the identification of the FC.

3.2.2 Model of the Single-Threshold α-Count Acting on a Healthy Component

The SAN depicted in Fig. 2, H_SAN-1, represents the behavior of the single-threshold α-count acting on the HC u. α is represented in H_SAN-1 by its approximation $\alpha\_sup \geq \alpha$; α_sup will then reach α_T in an average number of steps not higher than that required by α. The use of a lower approximation of the number of steps to signaling u as an FC ensures that the resulting value of NU is an excess approximation of the average unused fraction of the HC's life.

H_SAN-1 has four places. Places not_remov and $alpha\_state$ have the same meaning as in F_SAN-1 ($alpha\_state$ maintains the current value of α_sup), whereas two places are used to distinguish whether the component identified as being an FC is really subject to an intermittent/permanent fault or not. Place f_rem holds a token when the component becomes an FC, place $notf\_rem$ holds a token when α_sup crosses the threshold α_T, while u is still healthy. Activities $next\_step$ and $next\_miss$, which are exponentially distributed timed activities with rate 1, represent, as in the previous SAN, the receipt of a signal and the restart of $\alpha-$count upon issuing the $\alpha-$signal, respectively. The two output gates $incr$ and $decr$ compute α_sup and check its value against the threshold α_T.

In H_SAN-1, the activity $next\_step$ has three cases. *Case 1* represents the occurrence of a permanent or intermittent fault in u, which thus becomes an FC; *case 2* and *case 3* represent, respectively, the receipt of $J^{(L)} = 1$ and of $J^{(L)} = 0$ when u is an HC.

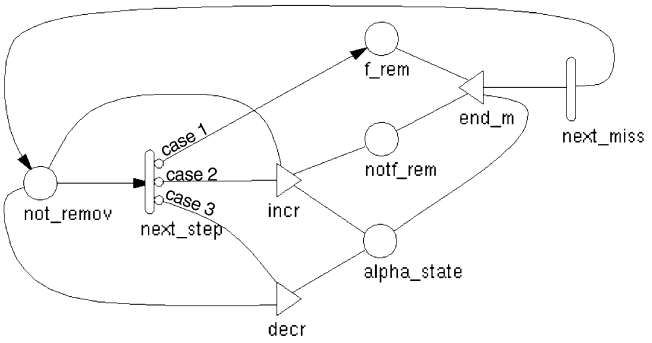


Fig. 2. H_SAN-1: Single-threshold α-count acting on a Healty Component, with exponentially distributed fault occurences.

$$P(Case\ 1) = q_p + q_i$$
$$P(Case\ 2) = q_t c + (1 - q_p - q_i - q_t)(1 - c)$$
$$P(Case\ 3) = q_t(1 - c) + (1 - q_p - q_i - q_t)c.$$

Let E_life be the number of steps to the expected occurrence of a permanent/intermittent fault and E_util be the expected number of steps in which the component is utilized by the system. Then, the unused fraction of a component's useful life is given by $(E\_life - E\_util)/E\_life$. In this model, E_life is given by: $E\_life = 1/(q_i + q_p)$, while E_util is obtained as the ratio between the time spent in not_remov and the time spent in either $notf\_rem$ or f_rem, that is, the ratio between the steady state probability of having one token in place not_remov and the steady state probability of having one token in either $notf\_rem$ or f_rem:

$$E\_util = \frac{P(\#not\_remov = 1)}{P(\#not\_remov = 0)}.$$

Hence,

$$NU = \frac{\frac{1}{q_i+q_p} - \frac{P(\#not\_remov=1)}{P(\#not\_remov=0)}}{\frac{1}{q_i+q_p}}$$
$$= 1 - \frac{P(\#not\_remov = 1)}{P(\#not\_remov = 0)}(q_i + q_p).$$

3.3 Evaluation of the Single-Threshold α-Count

The SAN models in Figs. 1 and 2 were analytically solved by using UltraSAN [5] to evaluate D and NU. The symbols and the default values used for all the parameters in the plots were defined in Section 2.2; the probability q of activation of an intermittent fault per time unit, given that the component is affected by an intermittent fault, is assigned the (default) value of 0.1.

The parameters K and α_T are internal to $\alpha-$count, i.e., under the designer's control. Their values have to be tuned, depending on the other parameters values, in order to get the desired behavior. Here, we show the behavior of α-count at varying values of K and α_T. An extended analysis of the sensitivity to the other (external) parameters can be found in [3].

Fig. 3 shows the plots of D and NU in terms of K for different values of α_T. Since K sets the length of a time window where signaled errors are "remembered" (albeit with diminishing weight), the memory effect induced by K

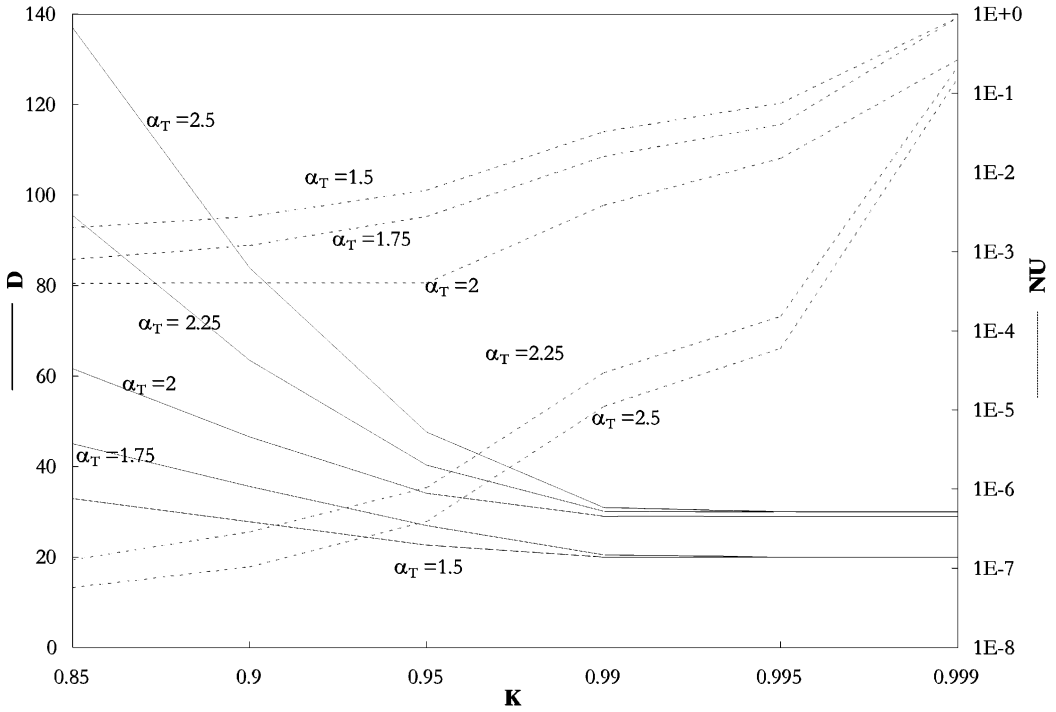


Fig. 3. Effect of the internal parameters K, α_T on the delay figure D and the utilization loss NU.

clearly has contrasting consequences on the delay to fault identification, D, and on the utilization loss, NU. High values of K, causing a slow decay of the α-count, make it quicker to catch an intermittent fault, i.e., give low values of the delay. However, this is at the cost of a higher utilization loss since there is more chance that enough transient faults add up to the α-count over the threshold. The opposite occurs at low values of K. With regard to the threshold height α_T, a symmetrical behavior is observed: Higher values require more error signals (and more time) to trigger the α-count output, thus improving NU while lengthening D.

More specifically, note that, at low values of K, the delay D markedly improves as K increases, up to a region around a value K_D (0.99 in the case of Fig. 3); for $K > K_D$, variations of D become smaller and smaller. In addition, D becomes less sensitive to the threshold height α_T in the region above K_D.

The curves plotted for NU highlight the effect of the discrete increment of α. The curves appear as grouped in two sets, those with α_T less or equal to 2 and those with α_T higher than 2, which depart from each other (for low values of K). The reason is that, in the first case, two errors signaled in the window are sufficient to single out the component, while, when $\alpha_T \geq 2$, three errors are usually necessary.

NU gets its best values for low values of K. The curves show a continuous, but slow, increase up to a region around a value K_{NU} and, for $K > K_{NU}$, NU becomes worse and worse. If $K_D < K_{NU}$, values of K in the interval $[K_D, K_{NU}]$ determine values for both the delay and the utilization loss which cannot be significantly further improved. Thus, it appears that, if no indications of their relative importance are given, values for K should be chosen in such an interval. However, the existence of this interval depends on other parameters as well. For example, as shown in Fig. 4, D is heavily influenced by the intermittent fault activation parameter q: The lower q is (i.e., the longer the expected time between intermittent fault activation), the higher K_D is.

Since a low NU and a low D are conflicting goals, the definition of the "best behavior" for α-count and, thus, the proper tuning of its parameters, entails defining their relative importance. Thus, even in cases where the range for values to be assigned to K can be identified, as previously discussed, it still has to be decided whether D or NU should be optimized: The former case requires low values of α_T, while high values optimize the latter one. More generally, parameters for a specific system can be tuned once the system designer has given constraints on the desired behavior of the mechanism, e.g., "D must be optimized while NU must take values lower than a given threshold."

4 Single-Threshold α-Count under More Accurate Distributions of Failures

The previous analysis of the single-threshold α-count was aimed at understanding the influence of the internal parameters on the performance measures. To make it easier to grasp the essentials of the mechanism's behavior, simple approximations to the fault process (i.e., exponential distribution) have been used. However, the constant rate assumption is not the best as far as intermittent faults are concerned as, in most cases, they tend to become more frequent after the first occurrence and may turn into permanent faults. Transient faults may occasionally occur

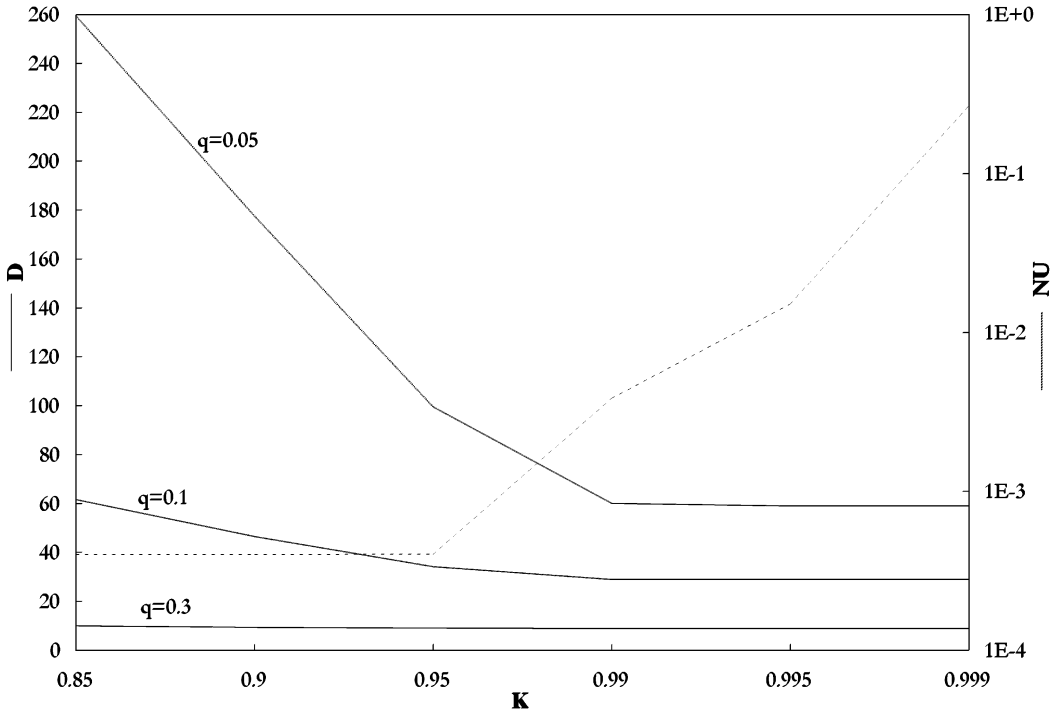


Fig. 4. Values of D and NU as a function of K for a few values of the intermittent fault manifestation probability, q.

with an abnormal frequency (e.g., during a storm). In this section, the original fault model will be refined to account for the above phenomena.

First, the activation rate of intermittent faults is considered as increasing in time after the first occurrence, that is, assumption 6′ of Section 2.1 will replace here assumption 6. Since this change only affects the behavior of faulty components, only the delay in identifying an FC will be reevaluated. It will be given in terms of the Cumulative Distribution Function $F_D(d)$ of the number of steps XD to the FC identification. The fault probability, of course, is given by a time-dependent function $q(d)$ instead of the constant-valued parameter q previously used.

Second, a piecewise exponential distribution of transient faults is introduced, where short, higher-frequency fault bursts emerge from the "noise floor" of the normal transient fault rate, that is, assumption 7′ of Section 2.1 will replace here assumption 7. In the presence of bursts, the evaluation of D by using the simple exponential model leads to a conservative estimation since the occurrence of bursts can only speed up the process of spotting a faulty component. The figures relative to HCs will be reevaluated in the following: The constant transient fault probability q_t is here substituted by the two values q_{ta}, q_{tb} which hold in the "normal" periods and in the "bursty" ones, respectively.

The modifications of the fault model will be introduced one at a time to simplify both the analysis and the comprehension of their effects. In the next two subsections, the detailed SAN models, obtained by modifying those previously presented to take into account the above changes, are described.

4.1 Another Model of the Single-Threshold α-Count Acting on an FC

F_SAN-2, depicted in Fig. 5, which represents the behavior of single-threshold α-count acting on a faulty component, is derived from F_SAN-1 by adding the place $count$, holding the running number of time units d, and the gate $next\_m$, described below. This allows the use of time-dependent probabilities at the cost of increasing the number of states of the model and the time and computational effort needed to obtain the solution. In order to cope with this potential state explosion problem, α-count is analyzed in a finite time interval of length Max_d, i.e., for $0 \leq d \leq Max\_d$ steps from the fault's occurrence, even though the identification of the FC has still to be achieved at that point. F_SAN-2 is described in terms of its differences with respect to F_SAN-1.

The place count is initialized to 1 and is updated by the two output gates $incr$ and $decr$. When $count$ holds $Max\_d + 1$ tokens, the gates $incr$ or $decr$, depending on the case, deposit a token in the place $remov$ and set to 0 the value of α_inf for a restart in $next\_miss$. Upon the firing of $next\_miss$, the gate $next\_m$ resets to 1 the number of tokens in $count$ and not_remov in order to restart the network. The expressions for the probabilities of the two cases of $next\_step$ are derived from those used in F_SAN $-$ 1 by substituting the constant-valued q with a time-dependent function $q(d)$, implemented as a marking-dependent expression $q(\#count)$.

The values of $F_D(d)$ for $d \leq Max\_d$ are obtained as follows: Consider that, whenever α-count succeeds in identifying the FC in at most Max_d steps, the place $count$ contains exactly XD tokens, so, let V be a variable defined as

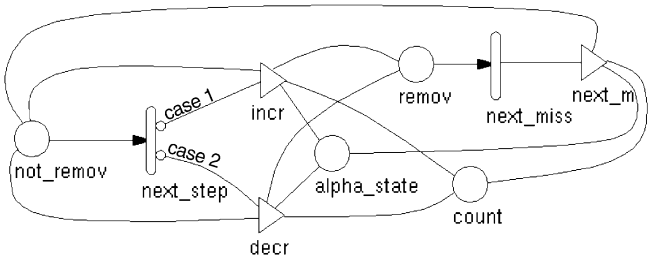


Fig. 5. F_SAN-2: Single-threshold α-count acting on an FC under time-dependent error rates of intermittent faults.

$$V = \begin{cases} \#count & \text{if } \#remov = 1 \\ 0 & \text{if } \#remov = 0 \end{cases}$$

from the steady-state evaluation of $F_V(d)$, the CDF of V, it follows:

$$F_D(d) = \frac{F_V(d) - F_V(0)}{1 - F_V(0)}.$$

Note that V may also take the value $Max\_d + 1$. This accounts for the probability that the identification of the FC u is not performed within Max_d steps from the occurrence of the fault.

4.2 Another Model of the Single-Threshold α-Count Acting on an HC

The behavior of the single-threshold α-count acting on an HC considering more realistic distributions of occurrences of transient faults is modeled by H_SAN-2, depicted in Fig. 6, which is an extension of H_SAN-1. H_SAN-2 represents the alternation of normal periods, whose duration is indicated by the random variable T_a, where the probability of an occurrence of a transient fault per time unit is q_{ta}, and of abnormal periods, having random lengths T_b, characterized by a higher probability q_{tb}. The switch between periods happens with rates λ_{ab} and λ_{ba}. Extending this model to account for many different periods is straightforward.

There are basically three differences between this net and H_SAN-1: 1)two places, $norm\_freq$ and $abnorm\_fr$ and two transitions, T_norm and T_burst, have been added to represent the alternation of normal and abnormal periods, 2) the expressions for the probabilities of case 2 and case 3 of $next\_step$ have been modified using q_{ta} or q_{tb} instead of q_t, depending on the marking of $norm\_freq$ and $abnorm\_fr$, 3) gate end_m has been given the additional task of setting to 1 the number of tokens in $norm\_freq$ and to 0 that of $abnorm\_fr$. From H_SAN-2, an upper bound NU to the utilization loss of the component is obtained, just like in H_SAN-1.

4.3 Evaluation of the Single-Threshold α-Count Using the New Models

The solution of the SAN models yields the values of $F_D(d)$ and NU. The measure 1-$F_D(d)$, indicating the probability of not identifying an FC component within d steps, has been plotted instead of $F_D(d)$ to help better show the effects of different time functions for the probability $q(d)$.

A function commonly recognized to fit the real life behavior of intermittent faults is the (discrete) Weibull distribution [2], [6]. Thus, although the SAN model

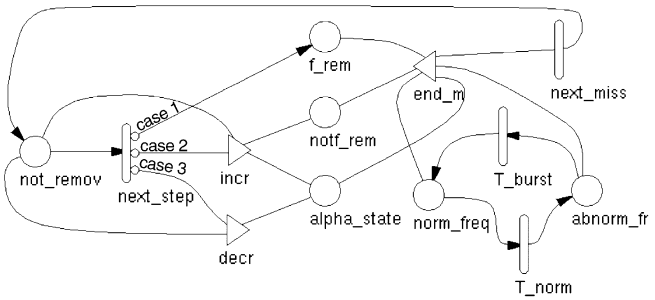


Fig. 6. H_SAN-2: Single-threshold α-count acting on a Healthy Component, with transient fault bursts.

(described in Section 4.1) representing the behavior of α-count on an FC allows the use of arbitrary functions of discrete time, the Weibull discrete-time hazard function will be adopted in the following evaluations, i.e.:

$$q(d) = 1 - (1 - q_w)^{d^\beta - (d-1)^\beta}, \quad d > 0,\ \beta \geq 1,\ \ 0 < q_w < 1.$$

To allow a significant comparison with the exponential distribution used in Section 3, the parameters q_w and β for a family of Weibull distributions are chosen so that the same expected number of intermittent faults is obtained for both distributions in a fixed initial time interval (namely, an average of six fault manifestations in 60 steps).

In the following evaluation, the expected duration of a bursty interval, T_b, will be in the range 10-50 steps (i.e., from six to 30 minutes), while the expected duration of a normal interval, T_a, will be given the two values 2,400 and 220,000 steps (corresponding to 24 hours and about three months, respectively).

The default values for the internal parameters K and α_T are 0.99 and 2.0, respectively. Other parameters, common to the exponential model (e.g., $q_p + q_i$, c, etc.), take the default values defined in Section 2.2 unless otherwise specified. The values of the remaining parameters are specified in the relevant figures.

The effects of an increasing intermittent fault rate on the delay to FC identification, represented by $1 - F_D(d)$, are shown in Fig. 7. Plots of $1 - F_D(d)$ for a few values of β and q_w are reported. Note first that the crossover, around $d = 60$, of the Weibull plots with the exponential plot (represented by $\beta = 1$) is due to the assumption on the averages of both distributions. It is apparent that the exponential model overevaluates the probability of spotting a faulty component at the left of the crossover and, on the other hand, gives lower estimates for larger values of the detection delay. In fact, in Fig. 7, one full order-of-magnitude difference between the exponential curve and the Weibull with $\beta = 1.4$ is observed for $d = 100$. It can thus be concluded that, whenever real intermittent faults get activated with increasing rate, the accuracy of the enhanced model improves significantly.

Fig. 8 shows the influence of burstiness on the utilization loss NU. Fig. 8a, where NU is plotted against T_b and q_{tb}, highlights that the presence of transient fault bursts significantly worsens the utilization loss caused by wrongly tagging an HC as faulty with respect to the exponential distribution. In the figure, the transient fault probabilities q_{ta} and q_{tb} are given value pairs such that the resulting time

averages equal the default q_t (i.e., 10^{-5}), to allow a direct comparison with the exponentially distributed behavior (shown by the curve labeled $q_{tb} = 10^{-5}$). The departure from the exponential behavior grows, as expected, with increasing q_{tb}, e.g., for q_{tb} three orders of magnitude over the default value, NU gets more than one order of magnitude worse. This is remarkable considering that the average fault probability is kept constant and that the burst length is a tiny fraction of the normal interval. With regard to variations in bursty intervals, higher values of T_b, implying a higher number of transient faults during those intervals, lead to higher values of NU; this influence obviously decreases with decreasing values of q_{tb}.

In Fig. 8b, the utilization loss NU is evaluated against α_T and q_{tb}. As already discussed in the pure exponential case, higher values of the threshold α_T have to be set to achieve lower values of NU. The situation gets worse in the presence of bursts: For any value of α_T, the higher q_{tb} is, the higher the utilization loss is. A measure of this effect can be observed by comparing the dashed curves labeled with $q_{tb} = 5 \cdot 10^{-3}$ and $q_t = 7.2 \cdot 10^{-5}$, plotted for distributions with the same mean. On the other hand, given a target value for NU (10^{-3} is chosen as an example in Fig. 8b), a corresponding value for α_T can be determined to attain it. The curves show that increasing values of α_T are required by increasing values of q_{tb}. Of course, the curve labeled with $q_t = 10^{-5}$, corresponding to the exponential distribution, intercepts the lowest value for α_T. Unfortunately, however, higher values of α_T lead to higher values of the identification delay D, with consequent higher risks in running the system while relying on a faulty component. The dilemma "NU vs. D," already faced at the end of Section 3.3, is, in fact, exacerbated by the presence of fault bursts, to the extent that it may be difficult to find a satisfying trade-off in the original α-count mechanism. The next sections describe a solution based on the idea of decoupling the effects of the threshold mechanism by adopting a double-threshold scheme.

5 A Double-Threshold Scheme

5.1 α-Count Using Two Thresholds

Consider now a new α-count scheme, where a component u is kept in full service as long as its score α is lower than a first threshold α_L. When the value of α grows bigger than α_L, u will be restricted to running application programs solely for diagnostic purposes, that is, its behavior will only be monitored by the error signaling subsystem, and the associated α-count mechanism will continue to operate. In other words, u is considered as a "suspect" and, as such, its results are not delivered to the user or relied upon by the system. If α continues to grow, it eventually crosses a second threshold α_H, with $\alpha_H > \alpha_L$: Then, the component u is diagnosed as being affected by a permanent/intermittent fault and the α-signal is issued, as in the original α-count, to the fault passivation subsystem. When, instead, α, after some wandering in the $(\alpha_H - \alpha_L)$ "limbo," becomes lower than or equal to α_L,[1] then the "suspect" component will be brought back into full service.

With this scheme, the lower threshold can be kept at a low level to limit the detection delay XD, while the probability of throwing away good components hardly increases. In fact, occasional fault bursts may anomalously raise an HC's α-count, forcing the component into a nonutilization period, but the probability of an irrevocable elimination can be kept as low as desired by setting a suitably high value for α_H.[2] An adverse effect of this scheme is the need to treat errors that show up in the units confined in the limbo. In fact, to continue observing their behavior, they need to be treated exactly like the on-line components, thus putting their share of burden on the error-processing subsystem.

The models to be used in the analysis of the double-threshold mechanism, under assumptions 1-5 and 6′, 7′ of Section 2.1, are introduced below.

5.2 Double-Threshold α-Count Acting on an HC

H_SAN-3 (Fig. 9) models the behavior of the double-threshold α-count acting on an HC. H_SAN-3 is an extension of H_SAN-2; as in the latter, a bursty fault behavior is modeled and the computation of an upper bound NU of the utilization loss of a healthy component is allowed.

There are basically four differences between H_SAN-3 and H_SAN-2:

1. The place $susp$ (initially set to 0) has been added to represent the alternation of periods in which the component is "suspect" (one token in $susp$) with those where the component is used and supplies results to the user (no token in $susp$);
2. The gate $incr$ has been given the additional task of setting to 1 the number of tokens in $susp$ when the value of α reaches the first threshold α_L;
3. The gate $decr$ has been given the additional task of setting to 0 the number of tokens in $susp$ when the value of α becomes lower than the first threshold α_L;
4. Gate end_m has been given the additional task of setting to 0 the number of tokens in $susp$ (when the component is tagged as faulty).

The unused fraction of a component's useful life is determined as in Section 3.2.2; here, however, E_util is the expected number of steps in which the HC is not suspected, i.e., used and relied upon; it is determined by the following expression:

$$E\_util = \frac{P(\#not\_remov = 1 \ AND \ \#susp = 0)}{P(\#not\_remov = 0)}.$$

Then:

1. In practice, a threshold value α_{L0} with $\alpha_{L0} \leq \alpha_L$ may be more appropriate here to put a degree of hysteresis in the mechanism with the goal of avoiding frequent "bouncing" between the "active" and "suspected" states.

2. An infinite value would, in fact, keep all units running, whether in "limbo" or not.

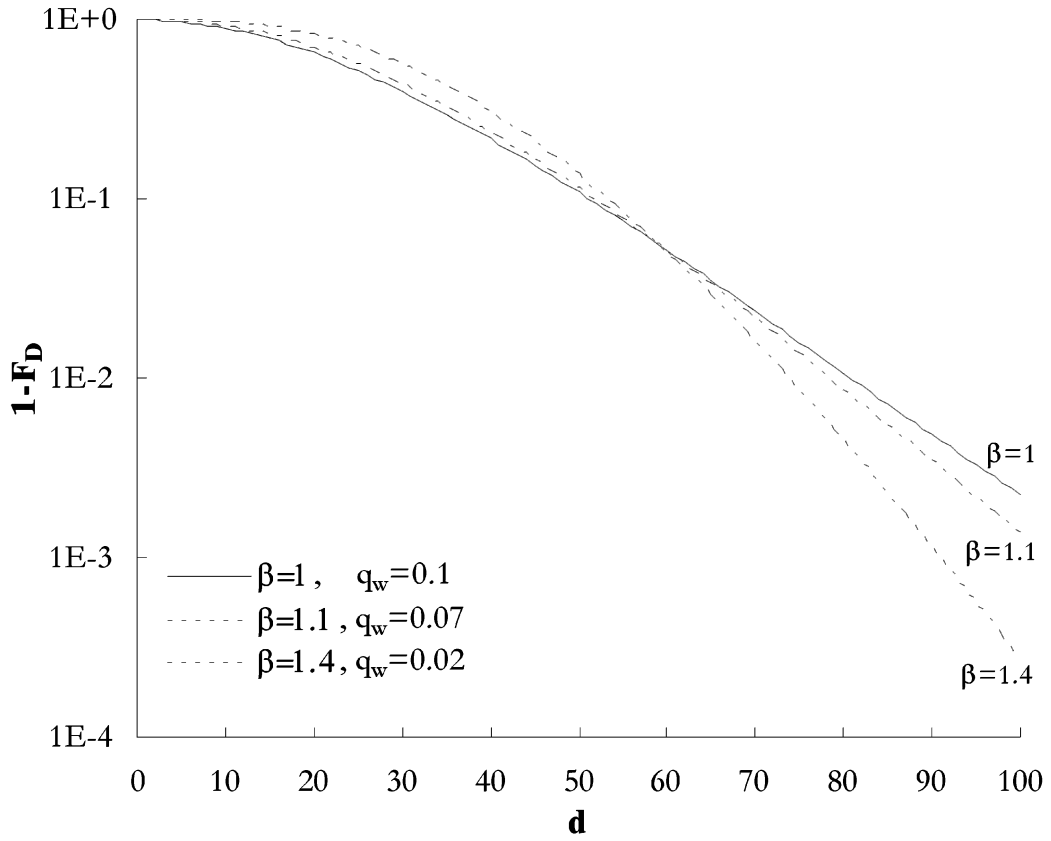


Fig. 7. Influence of increasing intermittent fault rate on the delay to FC identification.

$$NU = \frac{\frac{1}{q_i+q_p} - \frac{P(\#not\_remov=1\ AND\ \#susp=0)}{P(\#not\_remov=0)}}{\frac{1}{q_i+q_p}} = 1 - \frac{P(\#not\_remov = 1\ AND\ \#susp = 0)}{P(\#not\_remov = 0)}(q_i + q_p).$$

5.3 Double-Threshold α-Count Acting on an FC

The behavior of the double-threshold α-count acting on an FC, considering time-dependent error rates of intermittent faults, is modeled using F_SAN-3 (see Fig. 10), a SAN derived from F_SAN-2 (described in Section 4.1), which allows lower bounds to be computed for the number of steps the FC is kept in full service after the permanent/intermittent fault occurrence. As with F_SAN-2, the analysis has been limited to a finite time interval spanning Max_d time units.

There are three main differences between F_SAN-3 and F_SAN-2:

1. The place *count_notsusp* (initially set to 1 since, in the first step, the component is assumed to be in a "nonsuspect" state) counts the steps in which the FC is not suspected. It holds the number of steps elapsed with $\alpha\_inf < \alpha_L$ before the identification of FC as faulty or Max_d if Max_d steps went by, whichever event occurs first;
2. Gates *incr* and *decr* have been given the additional task of incrementing the number of tokens in *count_notsusp* when $\alpha\_inf < \alpha_L$. The gate *incr* deposits a token in the place *remov* when the value of α_inf reaches the value α_H;
3. Gate *next_m* has been given the additional task of resetting to 1 the number of tokens in *count_notsusp*.

F_SAN-3 allows a lower bound of $F_D(d)$ to be computed for $d \leq Max\_d$. Actually, the CDF $F_D(d)$ of XD is defined as

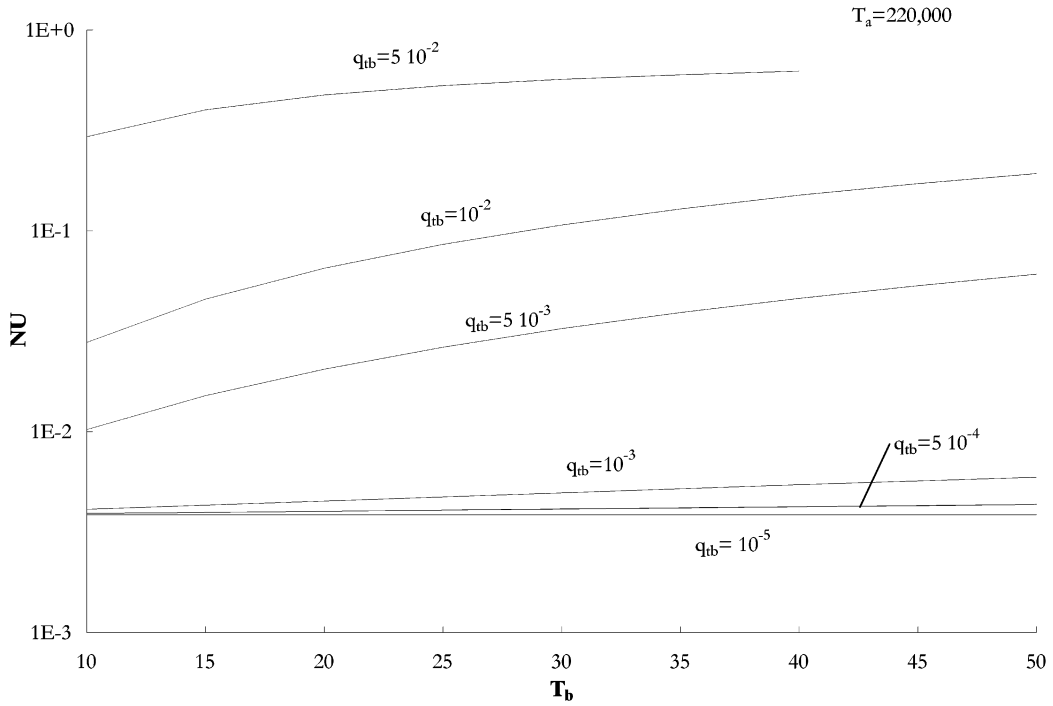
$$F_D(d) = P(XD \leq d) = P(XD \leq d | YD \leq Max\_d) \cdot P(YD \leq Max\_d) + P(XD \leq d | YD > Max\_d) \cdot P(YD > Max\_d),$$

where YD is the number of steps for α_inf to reach α_H. The approximation, obtained by considering

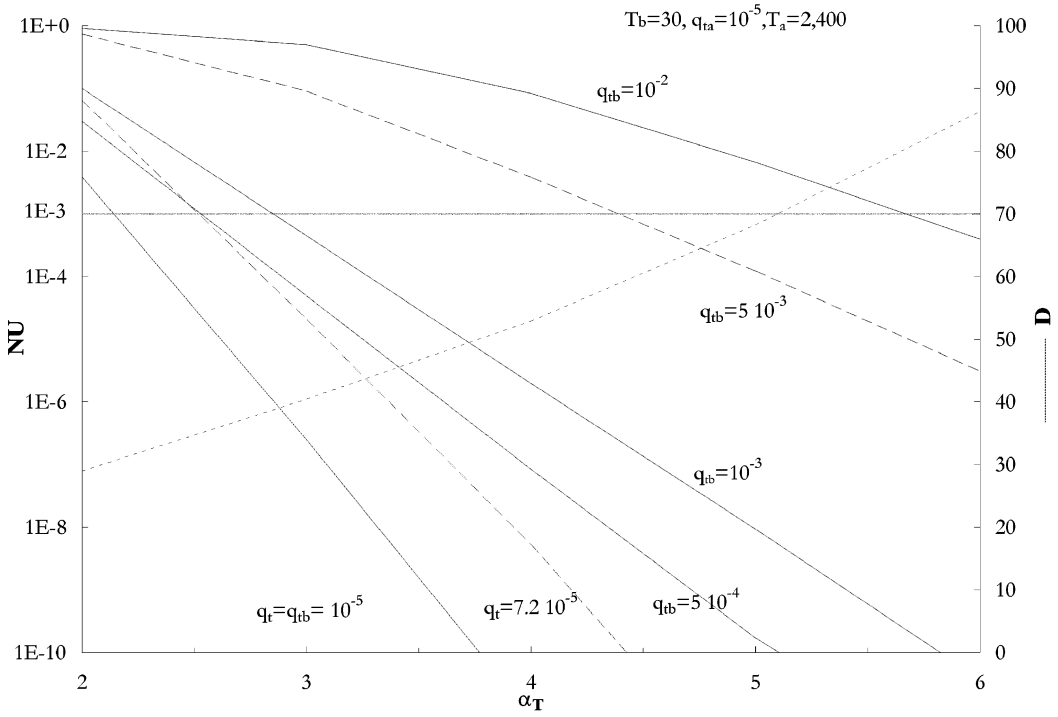
$$F_D(d) = P(XD \leq d | YD \leq Max\_d) \cdot P(YD \leq Max\_d) \quad \text{for } d \leq Max\_d,$$

clearly gives values always lower than the actual CDF.

To obtain a closed expression for $F_D(d)$ in terms of the SAN values, consider that, whenever α-count succeeds in identifying the FC in at most Max_d steps, the place *count* contains exactly YD tokens, while the place *count_notsusp* contains exactly XD tokens. Let V and W be auxiliary variables defined as:



(a)



(b)

Fig. 8. Influence of burstiness on utilization loss.

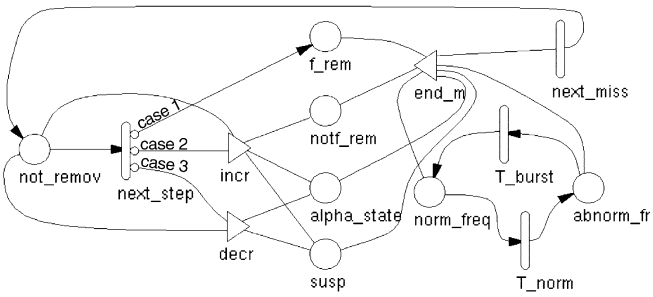


Fig. 9. H_SAN-3: Double-threshold α-count acting on an HC, with transient fault bursts.

$$V = \begin{cases} \#count & \text{if } \#remov = 1 \\ 0 & \text{if } \#remov = 0 \end{cases}$$

$$W = \begin{cases} \#count\_notsusp & \text{if } \#remov = 1 \ \ AND \ \ \#count \leq Max\_d \\ 0 & \text{otherwise} \end{cases}$$

and $F_W(d)$ be the CDF of the variable W, $F_V(d)$ the CDF of the variable V. Clearly:

$$P(XD \leq d | YD \leq Max\_d) = \frac{F_W(d) - F_W(0)}{1 - F_W(0)} \quad \text{for} \quad d \leq Max\_d$$

and

$$P(YD \leq Max\_d) = \frac{F_V(Max\_d) - F_V(0)}{1 - F_V(0)}.$$

To check how strict the lower bound thus obtained is, we can compute $P(YD > Max\_d)$ as well:

$$P(YD > Max\_d) = \frac{F_V(Max\_d + 1) - F_V(Max\_d)}{1 - F_V(0)}.$$

5.4 Evaluation of the Double-Threshold α-Count

The SAN models described in the previous two sections were analytically evaluated by UltraSAN and estimations for the measures NU and $F_D(d)$ were derived. The basic parameter settings are still those presented in Section 2.2 if not otherwise specified. The values of new parameters relating to the double-threshold α-count are indicated directly in the figures.

To understand the added features of the double-threshold scheme over the original single-threshold α-count, consider the typical procedure that a system designer might follow to set up the internal parameters of the mechanism. Given the external parameters, i.e., the fault rates and their characterizations, in the single-threshold case a near-optimal value of K can be determined from Fig. 3, which is likely to be around K_D. A figure like Fig. 8 can then be generated, where NU and D are plotted against the remaining internal parameter α_T. As can be seen, setting a target value for, say, NU, forces the value for D, leaving the designer with no leeway. The settings of NU and D should thus be traded off each other, according to their relative importance in the system design.

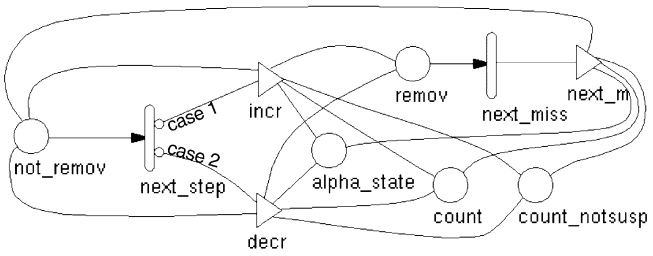


Fig. 10. F_SAN-3: Double-threshold α-count acting on an FC under time-dependent error rates of intermittent faults.

The performance figures resulting from the application of the single-threshold α-count are then compared with those obtained from the double-threshold variant. Hereafter, the range $[\alpha_L, \alpha_H]$ is always assumed to include the value of α_T. A suitable setting for parameters α_H and α_L has to be chosen. Since high values of the single threshold α_T give low values of NU, Fig. 8 could be used to find a suitable value for α_T which fits the target NU, which can then be assigned to the higher threshold α_H. The value to be assigned to α_L has to lead to a good trade-off between a relatively slight negative influence on the utilization factor of an HC and a satisfactorily low value for the probability of keeping an FC on-line. This is what is shown in Figs. 11 and 12.

Fig. 11 shows the effects of the variation of the lower threshold on the utilization loss NU of an HC. To give a quantitative measure of the expected increase in NU, because of nonutilization periods spent by HCs in "limbo," a comparison with the single-threshold case with $\alpha_T = \alpha_H$ is made. Four pairs of plots have been depicted; each pair has a plot relative to the single-threshold mechanism and the other relative to the double-threshold case, for different values of α_H and α_T. The pairs (α_T, q_{tb}) were chosen (on the basis of Fig. 8b) to give values of NU around 10^{-3}, which are considered significant for typical applications. In each plot pair, the difference between the two curves displays how much HC utilization is lost. For $\alpha_L = \alpha_T = \alpha_H$, the mechanism is reduced to the single threshold, i.e., the difference is zero. Clearly, this difference increases with decreasing values of α_L. The interesting fact is that the worsening of NU remains negligible with values of the lower threshold going as low as 1.1. Around this value, the curves exhibit a pronounced knee, which is not surprising since the elemental increment in α-count is, in fact, 1. A first-approximation choice for α_L is then around 1.1. In any case, at even lower values of α_L, there is no dramatic loss, e.g., for $\alpha_T = 3$ (and $q_{tb} = 10^{-3}$) at $\alpha_L = 0.5$, the utilization loss is around $4 \cdot 10^{-3}$: With such a low α_L setting, most permanent/intermittent faults would result in the neutralization of the affected component with virtually no delay.

Having set all the internal parameters, we only need to check whether the probability of keeping an unreliable component on-line is now low enough. This can be seen from Fig. 12, where $F_D(d)$ is plotted against d for different α_L, with $\alpha_H = 3$.

A refinement step can now be made by looking at the benefits to be gained by further shrinking α_L: The benefits of a smaller value may be weighed against the price to be

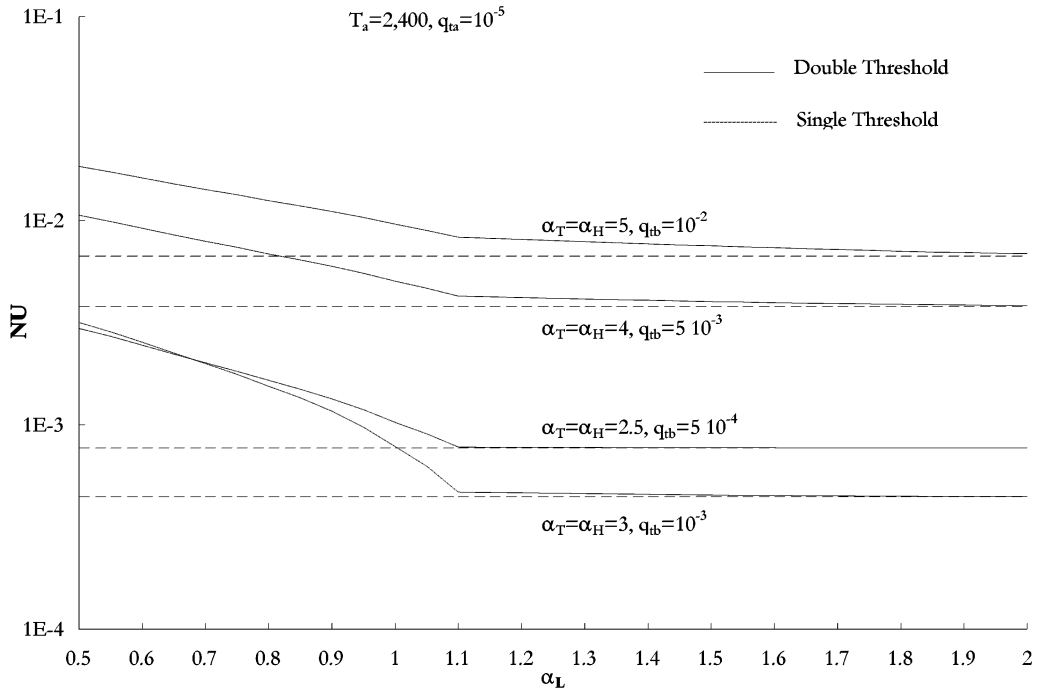


Fig. 11. Comparison of the effects of the double-threshold and the single-threshold on the utilization loss NU.

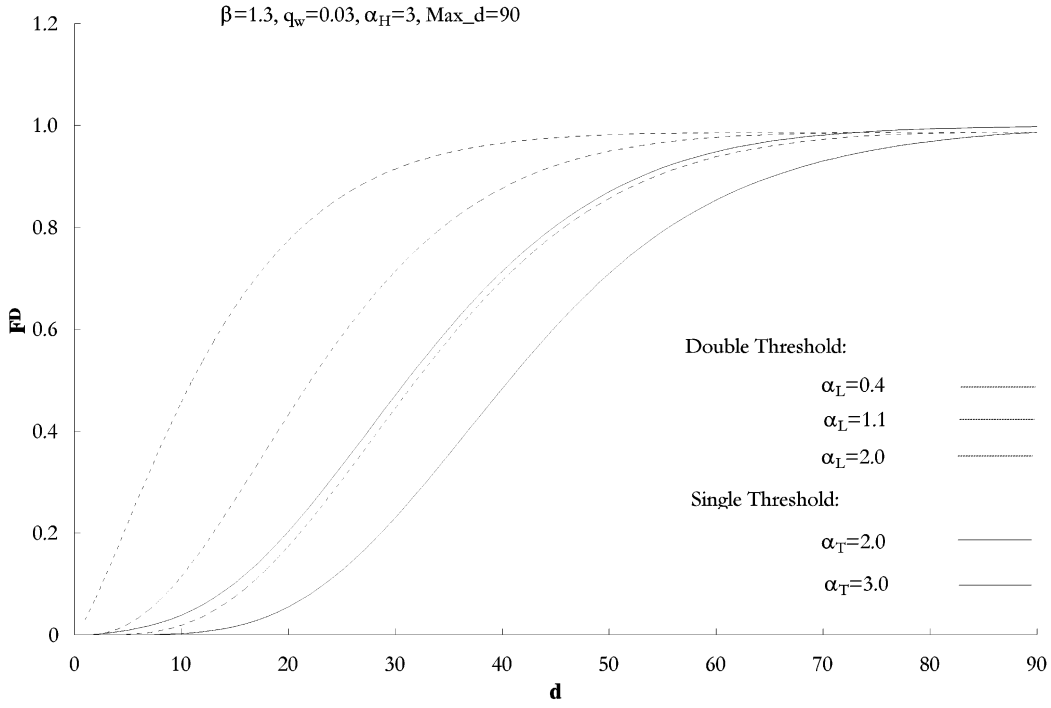


Fig. 12. Effects of the double-threshold and the single-threshold on the delay indicator $F_D(d)$.

paid in terms of NU that can be evaluated from Fig. 11; the step can be repeated until a good trade-off is found.

Since lower values for the threshold give lower delay D, in the comparison between the two α-count schemes, with reference to D, the better performing single-threshold is that where $\alpha_T = \alpha_L$. In Fig. 12, where $\alpha_H = 3$, the plot relative to $\alpha_T = 2$ and the one relative to $\alpha_L = 2$ show that the double-threshold incurs a loss, in terms of $F_D(d)$, with respect to the single-threshold. This loss is the consequence of the nonzero probability that an FC comes back into full service after first surpassing α_L. However, such a "better" behavior regarding $F_D(d)$ of the single-threshold is purely speculative, as it implies worsening NU: From Fig. 11, the two schemes have comparable values of NU if α_T has the same value as α_H. In Fig. 12, however, when $\alpha_T = \alpha_H$, the double-threshold α-count fares much better: the lower the value of α_L, the higher the improvement in the value of $F_D(d)$. For example, if $d = 40$, the probability of having

recognized an FC is about 0.5 for the single-threshold with $\alpha_T = 3$, while, for the double-threshold with $\alpha_H = 3$ and $\alpha_L = 1.1$, it is approximately 0.88. In Fig. 11 for the same setting of α_H, α_L, and α_T, the loss of utilization of an HC is negligible.

6 Related Work

We now consider some transient fault discriminating techniques from the literature. These can be broadly classified into two groups: A) techniques designed to support human intervention and B) algorithm-based, automatic mechanisms. We will relate some of them with α-count for the sake of simple comparisons.

6.1 Group A

In [7], a trend analysis of system error logs is applied which attempts to predict future hard failures by distinguishing intermittent faults (bound to turn into solid failures) from transient faults. The authors conclude, "Fault prediction is possible."

In [8], some heuristics, collectively named Dispersion Frame Technique, for fault diagnosis and failure prediction are developed and applied (off-line) to system error logs taken from a large Unix-based file system. The heuristics are based on the interarrival patterns of the failures (which may be time-varying). For example, there is the 2-in-1 rule, which warns when the time of interarrival of two failures is less than one hour, and the 4-in-1 rule, which fires when four failures occur within a 24-hour period.

In [9], error rate is used to build up error groups. Simple probabilistic techniques are then recursively applied to the quite detailed information given by the error record structure to seek similarities (correlations) among records, which may point to common causes (permanent faults) of a possibly large set of errors. The procedure is applied to event logs gathered from IBM 3081 and CYBER machines.

Because of the detailed error reports available from system logs, the procedures in this group can make sophisticated analyses and may allow more precise diagnoses or permanent failure predictions. However, they are normally applied off-line, they are seldom expressed in algorithmic form, and their complexity does not help the performance analysis. By contrast, the choice for α-count is to forego elaborate inferences on the health of the system, in favor of the higher predictability of a simple mechanism, possibly operating on-line. Despite its simplicity, α-count could be used as a basic tool to implement approximate versions of the above heuristics. For example, a 2-in-1-like rule [8] can be obtained using the single-threshold α-count by setting α_T to $1 + \Delta$ and K to $\Delta^{1/x}$, where x is the time window (or trigger frame) within which two failures are considered too many, and Δ, $0 < \Delta < 1$, is a parameter to represent the accuracy of the approximation to the 2-in-1 rule. By assigning to K the value computed with the above formula, when a second error hits the component u after less than x time units, the threshold α_T will be exceeded since, at the preceding time unit, the value of α associated with u is greater than Δ. With the same assumptions as above, it is clear that the parameters of the single-threshold α-count that satisfy the relation $((K^y + 1)K^y + 1)K^y + 1 = \alpha_T$ make up an approximation of the 4-in-1 rule [8] where, in this case, $4 \cdot y$ is the trigger frame.

6.2 Group B

In TMR MODIAC, the architecture proposed in [10], two failures experienced in two consecutive operating cycles by the same hardware component that is part of a redundant structure make the other redundant components consider it as definitively faulty.

In IBM machines, the idea of thresholding the number of errors accumulated in a time window has long been exploited. In the earlier examples, as in [11], which describes the automatic diagnostics of IBM 3081, the idea is used to trigger the intervention of a maintenance crew; in later models, as in ES/9000 Model 900 [12], a retry-and-threshold scheme is adopted to deal with transient errors.

In [13], error detectors and error-handling strategies are given ample reviews in recognition of their importance to tolerating transient faults. An "error-handling controller" gets its inputs from the error detectors to implement transient fault tolerance; its operation encompasses assessing each fault, labeling it as transient if it occurs less than K times in a given time window.

In [14], if, after a fault, a channel of the core FTP can be restarted, then it is brought back into operation; "however, it is assigned a demerit in its dynamic health variable; this variable is used to differentiate between transient and intermittent failures."

In [15], a list of "suspect" processors is generated during the redundant executions; a few schemes are then suggested for processing this list, e.g., assigning weights to processors that participate in the execution of a job and fail to produce a matching result and taking down for diagnostics those whose weight exceeds a certain threshold.

All of these solutions implement schemes simpler than α-count. In fact, when given suitable parameters, their functionalities may easily be obtained by an α-count instance, e.g., the approach proposed in [10] can be emulated by the single-threshold α-count with $K = 0$ and $\alpha_T = 2$.

7 Conclusions

We have introduced a family of mechanisms, collectively named α-count, to discriminate intermittent and permanent faults against low rate, low persistency transient faults. They are characterized above all by: 1) tunability through internal parameters, to warrant wide adaptability to a variety of system requirements; 2) generality with respect to the system in which they are intended to operate, to ensure wide applicability; 3) simplicity of operation to allow high analyzability through analytical models. The behavior of α-count has been extensively studied, in terms of performance figures, for use in the framework of the whole system's design. Fault hypotheses more adherent to real-life fault behavior than the conventional exponential distribution have been assumed; their effect on the basic mechanism led to the introduction of the double-threshold scheme.

A practical application of the ideas presented in this paper is being pursued in the context of the Esprit GUARDS

project [16]. Mechanisms inspired by the α-count scheme have been included in the design of the GUARDS generic architecture for real-time, dependable systems. Each "channel" of the redundant architecture is equipped with a distributed implementation of α-count for the management of the status of each channel and of the whole system, in order to reach consensus on the necessary reconfiguration actions. Three different prototypes are currently being developed by each end-user project partner.

Other uses of α-count, taken as a system building block, can be devised. For example, consider an N-modular redundant fault-tolerant structure, where instances of α-count are employed in each module for its original goal. A higher level diagnosis layer could be added, which would monitor the α-count values of all modules by looking for correlated errors: A common pattern of rising counts on several dials might, for instance, be an alarming symptom.

Finally, α-count can discriminate between different types of event streams, provided they exhibit sufficiently different characteristic rates. Many other uses of the mechanisms are therefore possible. In the field of dependable systems, α-count could be used to track errors *per se* (i.e., not to hunt faults), to decide whether or not to trigger a costly error processing protocol, which would normally require time-consuming system restart and state restoration procedures.

The practical efficacy of all the mechanisms based on thresholds depends on proper settings of the thresholds. Practitioners have long been used to tuning their systems using expertise and trial-and-error. Our effort aims to give the designer well-defined tools, to exert such actions in a systematic, predictable, and repeatable way. In our evaluations, and in showing how to tune the parameters, we have used fault statistics reported in the current literature. The tuning for a specific system can be carried out by applying the procedures described in this paper, using the actual fault parameters.

ACKNOWLEDGMENTS

This work has been supported, in part, by the CEC in the framework of the ESPRIT 20716 GUARDS project in which the authors are involved through PDCC. The authors would like to acknowledge all the project team for fruitful discussions.

REFERENCES

[1] J.C. Laprie, "Dependability—Its Attributes, Impairments and Means," *Predictably Dependable Computing Systems,* B. Randell, J.C. Laprie, H. Kopetz, and B. Littlewood, eds., pp. 1-28, Springer-Verlag, 1995.

[2] D.P. Siewiorek and R.S. Swarz, *Reliable Computer System—Design and Evaluation.* Digital Press, 1992.

[3] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Discriminating Fault Rate and Persistency to Improve Fault Treatment," *Proc. 27th IEEE FTCS—Int'l Symp. Fault-Tolerant Computing,* pp. 354-362, 1997.

[4] W.H. Sanders and J.F. Meyer, "A Unified Approach for Specifying Measures of Performance, Dependabiliy and Performability," *Dependable Computing for Critical Applications,* A. Avizienis and J. Laprie, eds., vol. 4 of *Dependable Computing and Fault-Tolerant Systems,* pp. 215-237, Springer-Verlag, 1991.

[5] W.H. Sanders, W.D. Obal, M.A. Qureshi, and F.K. Widjanarko, "The UltraSAN Modeling Environment," *Performance Evaluation J.,* vol. 24, pp. 89-115, 1995.

[6] H.E. Ascher, T.-T.Y. Lin, and D.P. Siewiorek, "Modification of: Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Trans. Reliability,* vol. 41, pp. 599-601, 1992.

[7] M.M. Tsao and D.P. Siewiorek, "Trend Analysis on System Error Files," *Proc. 13th IEEE FTCS—Int'l Symp. Fault-Tolerant Computing,* pp. 116-119, 1983.

[8] T.-T.Y. Lin and D.P. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," *IEEE Trans. Reliability,* vol. 39, pp. 419-432, 1990.

[9] R.K. Iyer, L.T. Young, and P.V.K. Iyer, "Automatic Recognition of Intermittent Failures: An Experimental Study of Field Data," *IEEE Trans. Computers,* vol. 39, pp. 525-537, 1990.

[10] G. Mongardi, "Dependable Computing for Railway Control Systems," *Proc. DCCA-3—Dependable Computing for Critical Applications,* pp. 255-277, 1993.

[11] N.N. Tendolkar and R.L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Research and Development,* vol. 26, pp. 78-88, 1982.

[12] L. Spainhower, J. Isenberg, R. Chillarege, and J. Berding, "Design for Fault-Tolerance in System ES/9000 Model 900," *Proc. 22nd IEEE FTCS—Int'l Symp. Fault-Tolerant Computing,* pp. 38-47, 1992.

[13] J. Sosnowski, "Transient Fault Tolerance in Digital Systems," *IEEE Micro,* vol. 14, pp. 24-35, 1994.

[14] J.H. Lala and L.S. Alger, "Hardware and Software Fault Tolerance: A Unified Architectural Approach," *Proc. 18th IEEE FTCS—Int'l Symp. Fault-Tolerant Computing,* pp. 240-245, 1988.

[15] P. Agrawal, "Fault Tolerance in Multiprocessor Systems without Dedicated Redundancy," *IEEE Trans. Computers,* vol. 37, pp. 358-362, 1988.

[16] D. Powell, J. Arlat, L. Beus-Dukic, A. Bondavalli, P. Coppola, A. Fantechi, E. Jenn, C. Rabéjac, and A. Wellings, "GUARDS: A Generic Upgradable Architecture for Real-Time Dependable Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, pp. 580-599, 1999.

Andrea Bondavalli (M-97) received his Laurea' degree in computer science from the University of Pisa in 1986. In the same year, he joined CNUCE-CNR, where he holds the position of "primo ricercatore" working in the Dependable Computing System Group. From April 1991 to February 1992, he was a guest member of the staff at the Computing Laboratory of the University of Newcastle upon Tyne (United Kingdom). Since 1989, he has been working on several European ESPRIT projects, such as BRA PDCS, BRA PDCS2, 20716 GUARDS, 27439 HIDE, and various national projects on dependable computing. He has also served as a program committee member for international conferences and symposia and as a reviewer for conferences and journals. His current research interests include the design of dependable real-time computing systems, software and system fault tolerance, the integration of fault tolerance in real-time systems, and the modeling and evaluation of dependability attributes like reliability, availability, and performability. He is a member of the IEEE and the IEEE Computer Society and the AICA Working Group on "Dependability in Computer Systems."

Silvano Chiaradonna graduated in computer science from the University of Pisa in 1992, working on a thesis in cooperation with CNUCE-CNR. Since 1992, he has been working on dependable computing and participated in European ESPRIT BRA PDCS2 and ESPRIT 20718 GUARDS projects. He was a fellow student at IEI-CNR and has also been serving as a reviewer for international conferences. His current research interests include the design of dependable computing systems, software and system fault tolerance, and the modeling and evaluation of dependability attributes like reliability and performability.

Felicita Di Giandomenico graduated in computer science from the University of Pisa in 1986. Since February 1989, she has been a researcher at the Institute IEI of the Italian National Research Council. During these years, she has been involved in a number of European and national projects in the area of dependable computing systems. She spent nine months (from August 1991 to April 1992) visiting the Computing Laboratory of the University of Newcastle upon Tyne (United Kingdom) as a guest member of the staff. She has served as a program committee member for international conferences, and as a reviewer for conferences and journals. Her current research activities include the design of dependable real-time computing systems, software implemented fault tolerance, issues related to the integration of fault tolerance in real-time systems, and the modeling and evaluation of dependability attributes, mainly reliability and performability.

Fabrizio Grandoni received his Laurea' degree in computer science from the University of Pisa. He then spent several years in the R&D departments of large industrial companies, studying array computer architectures for radar signal processing and working on the theory of self-diagnostic systems, where he coauthored the widely referenced BGM model. In 1980, he joined the research staff of IEI-CNR, where he is presently working in the Dependable Computing Systems Group. In 1980, he was active in the MuTEAM dependable multimicroprocessor project, whose results were awarded three presentations at FTCS-11. He then worked on Byzantine Agreement algorithms, on distributed traffic management in wide-area networks, and in such Esprit projects as Delta-4 and GUARDS. His current research interests include the design of dependable real-time computing systems, and software and system fault tolerance.