Threshold-Based Mechanisms to Discriminate Transient from Intermittent Faults[1]

Andrea Bondavalli, *Member, IEEE Computer society,* Silvano Chiaradonna,

Felicita Di Giandomenico and Fabrizio Grandoni

Abstract

A class of count-and threshold mechanisms, collectively dubbed α-count, able to discriminate between transient faults and intermittent faults in computing systems is presented. Transient faults discrimination has long been pursued in commercial systems: threshold-based techniques have been practiced for several years for this purpose. The present work aims to contribute to the usefulness of count-and-threshold schemes, through analysis of the behaviour and exploration of the effects on the system. A mathematically defined structure simple enough to be analysed by means of standard tools is adopted. α-count is equipped with internal parameters, designed to be tuned to suit environmental variables (such as transient fault rate, intermittent fault occurrence patterns). Extensive behaviour analysis for two embodiments of the scheme, both under the usual assumption of exponentially distributed fault rates and with more realistic fault patterns is carried out.

Key-words: Fault Discrimination, Threshold-based Identification, Transient and Intermittent faults, Modelling and Evaluation, Fault Diagnosis

---

[1]Technical Report, B4-17-06-98, IEI-CNR, June 17, 1998.

1 Introduction

In practically any computer system application, a degree of "reasonably" continuous service is expected, ranging from a few minutes wait after a crash (normally losing perhaps hours worth of work), to a few seconds/milliseconds downtime as often required in controllers of critical systems. This means that, upon the occurrence of a fault, some action has to be taken (whether by humans or by automatic mechanisms) to bring the system back into operation. In the most general case, the fault has to be located (fault diagnosis), the component affected has to be repaired/replaced, the effect of the fault has to be corrected; only then the normal computation can be restarted. This procedure is burdensome in terms of required time to complete and of system resources; moreover it is outright not viable for application fields where no repair or replacement is possible at all, so that the system must get the most from the 'original' components. On the other side of the coin, in case of short persistency faults, free of permanent effects in the system, putting off-line the affected component would be too radical a measure, provided the state of the computation can be someway saved, to be resumed later on. In fact, it has long been recognised [12] that most computer system malfunctions derive from short persistency faults. More formally, in [6, 12] physical faults are classified as permanent or temporary. Permanent faults do result into errors at each activation of the affected component; the only way of handling such faults is to remove the component. Temporary physical faults can be distinguished into internal faults (usually termed intermittent) and external faults (known as transient). The former, caused by some internal part going out its specified behaviour, usually exhibit a relatively high occurrence rate after their first appearance, and tend to become permanent, sooner or later. Conversely, the phenomenological cause of transient faults cannot be traced to a defect in a well identifiable part of the system (to be hopefully repaired or replaced). However, since their adverse effects rapidly disappear, the actions required for their treatment are less costly than in the case of permanent faults, provided their frequency is not too high: a simple retry or restart of the computation may suffice in commercial-grade systems, while in more demanding application fields on-line error masking and/or a proper state recovery of the faulty component before restart may be required.

Typical examples of applications which require special care in the treatment of transient faults are unattended dependable systems (e.g., aerospace applications), where temporary faults have to be treated such that the components must be kept in the system until the throughput benefit gained by keeping the faulty component on-line is offset by the greater probability of multiple (hence catastrophic) fault occurrences.

A wide range of techniques, spanning from simple retry to rather sophisticated off-line error log audit and trend analysis (see Section 6) have been used, or studied in the literature, to deal with transient faults with reference to specific systems. An idea which has long been exploited, e.g. in several IBM mainframes, has been to count the number of fault events: the number getting "too large" in a given time frame would signal the system component at hand as ready to be removed.

In this paper 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 is presented. Most of the similar solutions that can be found in the literature apply to general purpose mainframe systems, well equipped with system crash logs, giving huge, detailed information to be used in possibly complex post-mortem analyses. It has to be stressed that the goal of the present study is to shape mechanisms simple enough to: i) be thoroughly studied by standard analytical means and ii) to be implementable as small, low-overhead and low-cost modules, befitting even embedded, real-time, dependable systems. The behaviour of a basic embodiment of α-count has been described and analysed in [3], although with reference to a restricted fault model. In this paper a single-threshold and a double threshold schemes are presented, their analysis being carried out under fault hypotheses more accurate than the usual exponential fault distribution.

The rest of the paper is organised as follows. In Section 2 the general model of the system is introduced, together with the hypotheses on the component's behaviour and the performance figures for the fault discrimination mechanism. In Section 3 the basic single-threshold scheme is presented and analysed, under the simplistic exponentially-distributed faults assumption. Section 4 is devoted to modelling and evaluating the basic scheme under more realistic distribu-

tions of the fault occurrences. Section 5 introduces and analyses a double-threshold scheme, aimed to ease a trade-off between conflicting goals in the design of the basic scheme. In Section 6 previous work is briefly reviewed and some comparison with heuristics shown. Last, Section 7 offers some concluding remarks.

2 Preliminaries

In this section we first describe the view of the relevant parts of the system and of the error detection and signalling subsystem having an impact on the fault discrimination, then we describe the assumptions made in our analyses, last we deal with the figures of merit which are relevant in performing the discrimination.

2.1 The System

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), establishing the finest resolution for error detection and fault diagnosis.

A *fault passivation subsystem* is supposed to be present, which is in charge of processing the signals produced by the fault discrimination mechanisms which we are going to describe.

An *error detection subsystem* has the responsibility to judge if a component is behaving correctly or erroneously because of a fault. Computing systems are normally equipped with several low-level error detection mechanisms, whether in the value domain (e.g., overflow checking, divide-by-zero, program code signatures) or in the time domain (e.g., time-outs, watchdog timers). Apart from the above "embedded" error detectors, error notification may be generated by testing programs, which can be activated periodically, or whenever spare CPU time is available. In systems designed to provide for fault-tolerance, the error processing mechanisms, which in some form must be present to mask the effect of faults, can be easily geared with supplemental error signalling tools; e.g., a standard TMR configuration, whose duty is to deliver an output value as long as it is produced by at least two agreeing processors, may well deliver

also a pointer to a disagreeing processor, should unanimity not occur. Another source of diagnostic information may be made up by distributed agreement algorithms, augmented with disagreement indications. In general, the judgement does not reflect in a perfect way the state of the component. For instance, if error detection is implemented by applying test procedures, a faulty component may test as correct, whether because the test procedure is not thorough enough and fails to exercise the faulty functions of the component, or the adjudication of test outcomes may misinterpret erroneous test results as correct. The error detection results into signals, which are assumed to have binary values for this paper's purposes. We draw from this signal stream to feed the fault discrimination mechanism.

2.2 Basic assumptions

The analytical modelling and quantitative evaluations of the fault discrimination mechanisms introduced in the following Sections will be carried out in two phases: i) the conventional assumption of exponential fault distribution is taken, then ii) more realistic scenarios, including increasing rate of intermittent fault manifestations and bursty transient fault intervals, will be considered.

In what follows, reference will be made, by default, to a generic system component. The items 1 to 7 below state the general assumptions pertaining to phase i):

1- All fault-related events (error detectors triggering, testing routines completion, etc.) occur at discrete points in time; two successive points in time differ by a (constant) *time unit* (or *step*). For each component, a default *state-indicator* signal (e.g., "I'm alive and well"), bound to be overridden by real error signals, is waved at each step.

2- The faults affecting the hardware directly supporting the discrimination mechanisms are not considered.

3- The signal on the components behaviour (both the judgement explicitly given by the error signalling mechanism and the default state-indicator signal) is correct with a probability c.

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 internal fault; therefore, a component experiencing an intermittent fault may also be hit by a transient fault, but not by a permanent fault.

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 an intermittent fault does repeatedly give rise to an error with a constant probability q at each step.

7- A transient fault lasts only for one 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 ii) is defined by keeping the hypotheses 1 to 5 above, while substituting assumptions 6 and 7 with the following:

6'- A component affected by an intermittent fault does repeatedly give rise to an error, at each time unit after the first one, with a probability expressed by a generic function $q(d)$, where d is the discrete time variable.

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

2.3 Relevant figures of interest

The goal of our fault discrimination mechanisms may be informally expressed as to try to balance between two conflicting requirements:

i) Signal all components affected by permanent or intermittent faults (referred to in the following as *faulty components*, or *FC*s) as quickly as possible. In fact, gathering information to discriminate between transient and intermittent faults takes time, giving way to a longer fault latency, which rises the requirements on the error processing subsystem.

ii) Avoid signalling components other than FCs (referred to in the following as *healthy components*, or *HC*s); depriving the system of valid resources may be hurting not less than relying on a lame component.

The performance of the mechanisms with respect to the above goal is given quantitative measures by the following figures:

1) the total elapsed time XD, after the (permanent or intermittent) fault occurrence in a component u, in which it is maintained in use and relied upon, because its condition has not yet been recognised. In place of XD other related measures can be given, e.g.:

a) the average $E[XD]$, or

b) the Cumulative Distribution Function (CDF) F_D of XD: $F_D(d) = P(XD \leq d)$.

2) Some measure of the penalty inflicted by the discrimination mechanism on the utilisation of HCs. This can be expressed as the fraction of component life NU, (with respect to its expected life as determined by the expected occurrence of a permanent/intermittent fault) in which the otherwise healthy component u is not effectively used in the system.

3 A basic threshold-based filtering mechanism and its analysis

3.1 α-count, an on-line threshold-based fault identification mechanism

Like 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 (even the default defined in Section 2.2) related to the correctness of a given component, as time goes on, 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, albeit satisfactory, formulation of the filtering function α is the following:

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

$$\alpha^{(0)} = 0$$

where α is a score associated to each not-yet-removed component u to record information about the failures experienced by that component, and $J^{(L)}$ indicates the L-th signal on the generic component u: $J^{(L)} = 1$ upon failure, $J^{(L)} = 0$ otherwise.

When the value of $\alpha^{(L)}$ exceeds a given threshold α_T, the component u is diagnosed as affected by a permanent or a dangerously frequent intermittent fault; this event produces an *α-signal* , sent to the fault passivation subsystem. The values to be assigned 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 judgements of the used error signalling mechanism. As a first flavour of the meaning of K and α_T, note that $\lceil \alpha_T \rceil$ represents the minimum number of consecutive failures sufficient to consider a component so bad to be removed, while K represents the ratio by which α is decreased after a success (related to how long memory of previous failures is retained).

The filtering function α may be given formulations different from the expression (1) above, in the seek of peculiar 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.

$$\alpha^{(L)} = \begin{cases} \alpha^{(L-1)} + 1 & \text{if} \quad J^{(L)} = 1 \\ \alpha^{(L-1)} - \text{dec} & \text{if} \quad J^{(L)} = 0 \text{ and } \alpha^{(L-1)} - \text{dec} > 0 \\ 0 & \text{if} \quad J^{(L)} = 0 \text{ and } \alpha^{(L-1)} - \text{dec} \le 0 \end{cases} \qquad 0 \le \text{dec} \qquad (2)$$

With this function, the decay time of the current value is proportional to the value itself, while in the multiplicative form (1) it depends (roughly) only on the parameter K. This would result into different optimal settings for the mechanism's parameters, and also into differences in be-

haviour; however, the modelling and the analysis will be worked out for the (1) form only, since it is straightforward to adapt the methods used to the form (2) (and of course to other variants of the α function as well).

Because of the exponential fault distributions, it is trivial to see that a FC will always, eventually, be identified as such; unfortunately, the same argument prevents excluding the possibility that a HC could be wrongly signalled as faulty.

3.2 Modelling the single-threshold α–count

In the rest of this Section, a first phase of the evaluation of the single-threshold α–count is carried out, using the hypotheses 1 to 7 of Section 2.2. The behaviour of the mechanism has been modelled by Stochastic Activity Networks (SAN) [10]. Two SANs have been derived: the first, F_SAN-1, modelling the mechanism acting on a faulty component, used to derive E[XD], which in the present single-threshold context assumes the meaning of the average delay between an intermittent fault occurrence and its signalling; the second, H_SAN-1, modelling the mechanism acting on a healthy component, which allows evaluating the average of NU as a measure of the utilisation penalization. As required by SAN models, a discrete approximation of the real valued variable α is used in the subsequent SANs. The resulting precision had to be traded off between the price of increasing the time and computational effort necessary to obtain the solution.

3.2.1 Single-threshold α–count acting on a Faulty Component

A faulty component u, by definition, is affected either by a permanent or by an intermittent fault. The estimation of the number of steps needed to recognise a permanent fault is trivial: the probability of $J^{(L)} = 1$ is very close to 1 (it is 1•c), hence the expected time to the threshold crossing is approximately bounded by $\lceil \alpha_T \rceil$, considering that the value of α will be in general greater than 0 when the permanent fault will occur. The case of intermittent faults, which requires longer time for the identification, is analysed through F_SAN-1 (shown in Figure 1). Inside F_SAN-1, α is represented by its approximation variable α_inf$\leq\alpha$; α_inf will then reach α_T in an average number of steps $\overline{D}$ higher than, or equal to, E[XD] the average number of steps

which would be required by the real α. Moreover, to enforce $\overline{D}$ to be an upper bound, we assume that at the first occurrence of a permanent/intermittent fault α_inf= 0, while in general α_inf will take positive, albeit small, values.

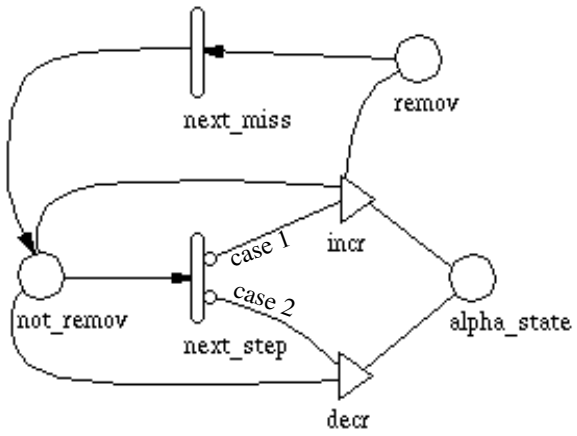

Figure 1. F_SAN-1 Single-threshold α−count acting on a Faulty Component with exponential fault occurrence

F_SAN-1 has three places: i) *not_remov* (the presence of a token in this place means that u is not yet flagged as faulty); ii) *remov* (a token here marks the actual identification of u as FC, with the attending emission of the α-signal); iii) *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 for a proper restart in *next_miss*.

The probabilities associated to *case 1* and *case 2* in the transition *next_step* are expressed as:

$$P(\text{Case 1}) = (q + q_t)c + (1 - q - q_t)(1 - c)$$
$$P(\text{Case 2}) = (q + q_t)(1 - c) + (1 - q - q_t)c$$

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 properly detected by the error signalling mechanism (with probability c); while the second term represents the incorrect detection of a failure. The second expression has a symmetric explanation.

The average $\overline{D}$ 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 Single-threshold α−count acting on a Healthy Component

The SAN depicted in Figure 2, H_SAN-1, represents the behaviour 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 would be required by α. The use of a lower approximation of the number of steps to the identification of a FC leads to the evaluation of the upper bound $\overline{NU}$ to E[NU].

H_SAN-1 has four places. Places *not_remov* and *alpha_state* have the same meaning as in H_SAN-1 (*alpha_state* maintains the current value of α_sup), whereas two places are used to distinguish whether the component identified as being a FC is really subject to an intermittent/permanent fault or not. Place *f_rem* holds a token when the component becomes a 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 α−count upon the issuing of the α−signal, respectively. The two output gates *incr* and *decr* compute α_sup and check its value against the threshold α_T.

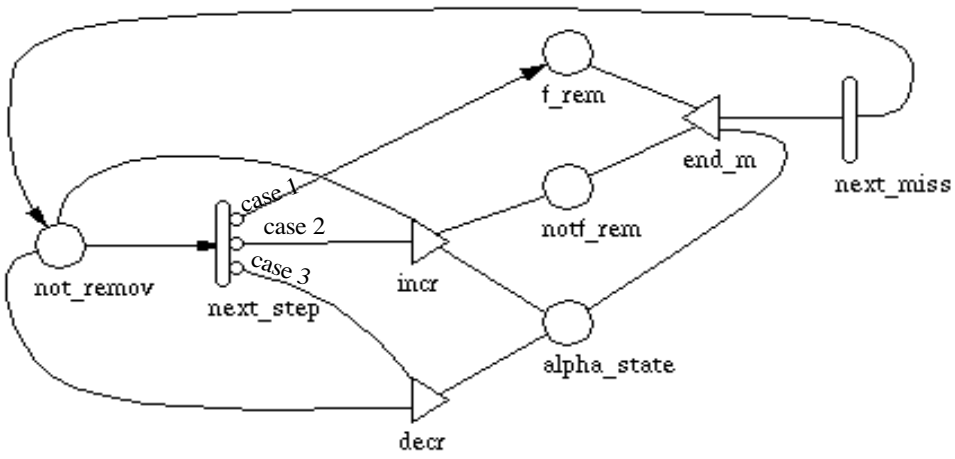

Figure 2. H_SAN-1 Single-threshold of α−count acting on a Healthy Component, with exponential fault occurrence.

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 a FC, *case 2* and *case 3* represent respectively the receipt of $J^{(L)} = 1$ and of $J^{(L)} = 0$ when u is a HC.

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

From its definition, NU is computed as the difference between the number of steps to the expected occurrence of a permanent/intermittent fault (E_life) and the expected number of steps in which the component is utilised by the system (E_util) divided by the former. In this model, E_life is given by: $E\_life = \frac{1}{(q_i + q_p)}$, while E_util is obtained as the ratio between the time spent in *not_remo*v 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_remo*v and the steady state probability of having one token in either *notf_rem* or *f_rem* :

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

Hence:
$$\overline{NU} = \frac{\frac{1}{(q_i + q_p)} - \frac{P(\#\, not\_removed = 1)}{P(\#\, not\_removed = 0)}}{\frac{1}{(q_i + q_p)}} = 1 - \left(\frac{P(\#\, not\_removed = 1)}{P(\#\, not\_removed = 0)} \bullet (q_i + q_p)\right).$$

3.3 Evaluation of single-threshold α–count

The SAN models in Figures 1 and 2 have been analytically solved by using UltraSAN [11] to evaluate $\overline{D}$ and $\overline{NU}$. For the sake of simplicity in notation, D and NU will be used in the sequel in place of $\overline{D}$ and $\overline{NU}$, respectively.

Symbol	Definition	Value
q_t	probability of transient fault per time-unit	10^{-5}
$q_p + q_i$	probability of intermittent or permanent fault per time unit	10^{-6}
q	probability of activation of an intermittent fault per time unit, given that the component is affected by an intermittent fault	0.1
c	probability that a signal on the components behaviour is correct	$1-10^{-5}$
K	the ratio by which α is decreased upon receiving $J^{(L)}=0$	0.99
α_T	threshold for identifying a component as affected by a permanent or intermittent fault	2

Table 1. Symbols, definitions and default values

Table 1 recalls the symbols used for all the parameters, with their definition and the default values used in the plots unless explicitly specified. The values for q_t , $q_p + q_i$ are derived assuming that the probability of intermittent or permanent fault be 10^{-4}/hour, that the probability of transient faults be 10^{-3}/hour, and that time be discretized into 1/100 hour units.

The parameters K and α_T are internal to α–count, i.e., under the control of the designer. Their values have to be tuned, depending on the other parameters values, in order to get the best behaviour. Here we show the behaviour of α–count at varying values of K and α_T, to provide insights on how to devise proper values for tuning the internal parameters. An extended analysis to the sensitivity to the other (external) parameters can be found in [3].

Figure 3 shows the plots of D and NU in terms of K for different values of α_T. Observe first that, while increasing values of K improve D (which gets lower values) and worsen NU, the opposite effects may be observed regarding α_T. At low values of K, D markedly improves as K increases, up to a region around a value K_D (.99 in case of Figure 3); for $K>K_D$ variations of D become smaller and smaller. In addition, D increases for increasing values of α_T; however, it becomes less sensitive to α_T above the region around K_D.

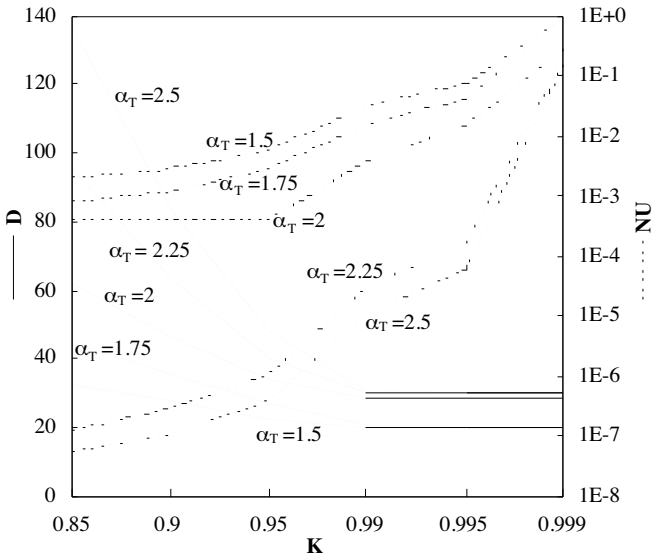

Figure 3. Values of D and NU as a function of K for varying α_T

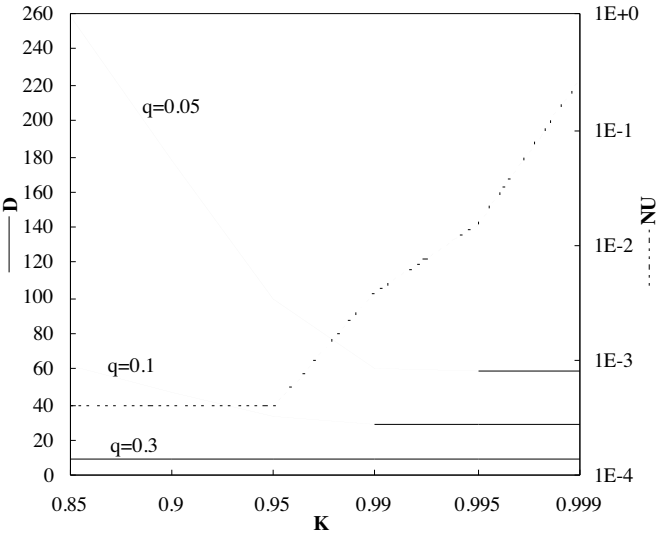

Figure 4. Values of D and NU as a function of K for varying q

NU grows very slowly for low values of K up to a region around a value K_{NU}, whereas it becomes worse and worse for $K > K_{NU}$. Moreover it is very sensible to the value of α_T. Contrary to the effect α_T has on D, increasing values of α_T improve the figures obtained for NU. If $K_D < K_{NU}$, values of K in the interval $[K_D, K_{NU}]$ determine values for D and NU close to their optimum. It appears thus that, if no other constraints are given, values for K should be chosen in such interval. However, the existence of such interval depends on other parameters as well. For

example, as shown in Figure 4, D is heavily influenced by q: the lower is q (i.e. the higher the time between intermittent fault activation), the higher is K_D.

Since a low NU and a low D are conflicting goals, the definition of the "best behaviour" for α–count, thus the proper tuning of its parameters, requires the definition of their relative importance. So, even in cases where the range for values to be assigned to K can be identified, as in the example of Figure 3, still it has to be decided whether D or NU should be optimised: the former case asks for low values of α_T while high values optimise the latter one. More generally, tuning of parameters for a specific system can be done once the system designer has given constraints on the desired behaviour of the mechanism, e.g. "D must be optimised 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 has been mainly directed to understand the influence of the internal parameters on the performance measures. To make it easier to grasp the essentials of the mechanism's behaviour the simpler 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, until possibly be stuck into permanent faults. Transient faults, for their part, may occasionally happen with abnormal frequency (e.g. in the course of a storm). In this Section, two refinements will then be added to the original fault model. First, the activation rate of intermittent faults is considered increasing in time after the first occurrence, that is assumption 6' of Section 2.2 will replace here the assumption 6. Since this change impacts only the behaviour of faulty components, only the delay in identifying a FC will be re-evaluated. Second, a piecewise exponential distribution of transient faults is introduced, where short, higher-frequency fault bursts emerge from the "noise floor" of the average transient fault rate, that is assumption 7' of Section 2.2 will replace here the assumption 7. In presence of bursts, the evaluation of D by the simple exponential model results into a conservative estimation, since the occurrence of bursts can only speed up spotting a dying component. Instead, the probability

of identifying a HC as a FC gets increased. As a consequence, only the figures relative to this case will be re-evaluated in the following. The modifications of the fault model will be introduced one at a time, to simplify both the analysis and the comprehension of their effects.

4.1 A second model of single-threshold α–count acting on a FC

F_SAN-2, depicted in Figure 5, which represents the behaviour 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 the states of the model and the time and computational effort necessary to obtain the solution. In order to cope with this potential state explosion problem, α–count is analysed in a finite time interval of length Max_d, i.e., for 0 ≤d≤ Max_d steps from the fault occurrence, even though the identification of the FC has not by then been attained. F_SAN-2 is described by difference with respect to F_SAN-1.

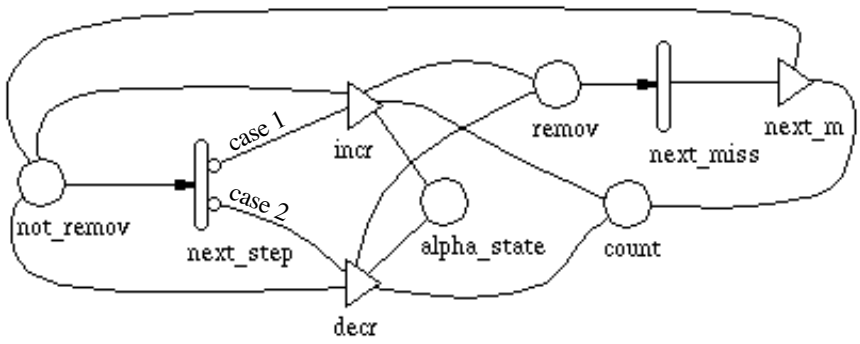

Figure 5. F_SAN-2: Single-threshold α–count acting on a FC under time dependent error rates of intermittent faults

The place *count* is initialised 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* , as the case might be, deposit a token in the place *remov* and set to 0 the value of α_inf for a proper restart in *next_miss*. Upon firing of *next_miss*, the gate *next_m* resets to 1 the number of tokens in *count* and *not_remov* for a proper restart of the network. The expressions for the probabilities of the two cases of *next_step* are derived from those used in F_SAN-1, by the substitution of the constant-valued q with a

time-dependent function q(d), implemented as a marking dependent expression $q(\text{MARK(count)})$.

In the following analysis we shall compute a lower bound $\bar{F}_D(d)$ for d≤Max_d of the Cumulative Distribution Function (CDF) F_D of XD: $F_D(d) = P(XD \le d)$. The values of $\bar{F}_D(d)$ for d≤ 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

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

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

$$\bar{F}_D(d) = \frac{F_V(d) - F_V(0)}{1 - F_V(0)}$$

Notice that V may take also the value Max_d+1. This represents the probability that the identification of the FC u is not performed within Max_d steps from the fault occurrence.

4.2 A second model of single-threshold α–count acting on a HC

The behaviour of single-threshold α–count acting on a HC considering more realistic distributions of occurrences of transient faults is modelled by H_SAN-2, depicted in Figure 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 occurrence of a transient fault per time unit is q_{ta}, and of abnormal periods, having random lengths T_b, characterised 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.

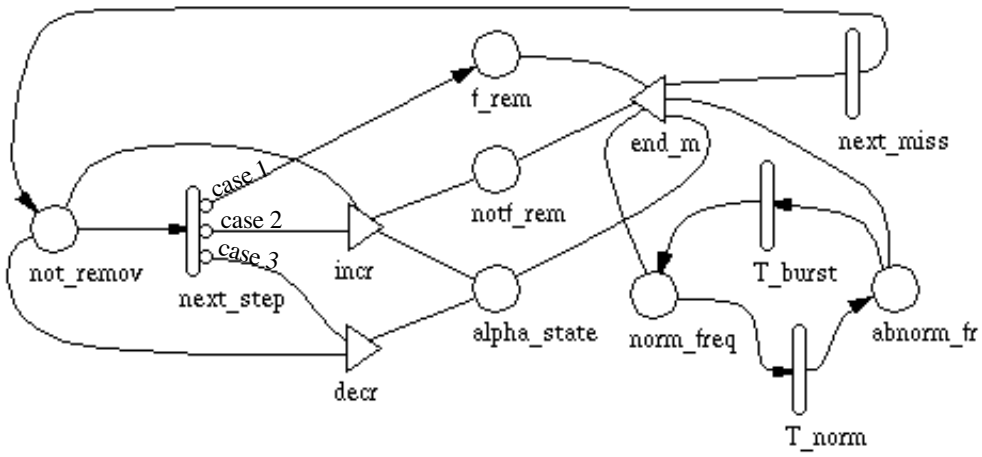

Figure 6. H_SAN-2 Single-threshold α−count acting on a Healthy Component, with transient fault bursts.

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} in place of q_t depending on the marking of periods, 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 $\overline{NU}$ to NU, the utilisation loss of the component, is obtained as in H_SAN-1.

4.3 Evaluation of single-threshold α−count using the new models

The SAN models of Figures 5 and 6 have been analytically solved by using UltraSAN [11], and $\overline{F}_D(d)$ and $\overline{NU}$ have been evaluated. The measure $1-\overline{F}_D(d)$, instead of $\overline{F}_D(d)$, has been plotted to help appreciate the effects of different discrete-time functions for the probability q(d). For the sake of simplicity in notation, $F_D(d)$ and NU will be used in the sequel in place of $\overline{F}_D(d)$ and $\overline{NU}$, respectively.

Since intermittent faults tend to manifest themselves as errors occurring at increasing rates, a function commonly used to model this behaviour is the (discrete) Weibull distribution [2, 12].

Thus, although F_SAN-2 allows to use arbitrary functions of discrete time, the Weibull discrete-time hazard function will be used in the following evaluations , i.e.:

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

To allow a significant comparison with the exponential rate distribution used in Section 3, the parameters q_w and β for a family of Weibull 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 6 fault activations 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 6 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 3 months, respectively).

Other parameters, common to the exponential model (e.g. K, $q_p + q_i$, etc.), will be given the default values defined in Table 1 unless otherwise specified. The values of the remaining parameters are specified in the appropriate Figures.

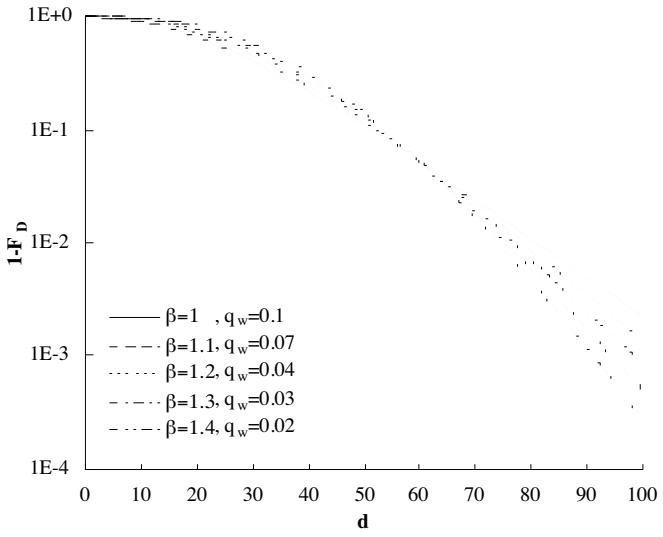

Figure 7. Values of $1\text{-}F_D(d)$ for varying β (and q_W)

Figure 7 shows the plots of $1\text{-}F_D(d)$ for several values of β and q_W. Note first that the crossover, around d=60, of the Weibull plots with the exponential plot (represented by β =1) is due to the

assumptions on the averages of both distributions. It is apparent that the exponential model over-evaluates the probability of spotting a faulty component at the left of the crossover, and, vice-versa, gives lower estimates for larger values of the detection delay. In fact, in the Figure one full order-of-magnitude difference between the exponential curve and the Weibull with β=1.4 is observed for d=100. Therefore it can be concluded that, whenever the real intermittent faults do activate with a rising rate, the accuracy attained by the enhanced model gets a significant improvement.

Figure 8(a) shows that the presence of transient fault bursts significantly impacts the utilisation loss caused by the premature tagging of a HC as faulty. In the figure the values 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 behaviour (shown by the curve labelled $q_{tb} = 10^{-5}$). The departure from the exponential behaviour grows, as expected, with increasing q_{tb}; e.g., for q_{tb} three orders of magnitude over the normal fault rate, NU gets more than one order of magnitude worse. This is most remarkable observing that the average fault rate is kept constant, and that the burst length is a tiny fraction of the normal interval. For low values of q_{tb} NU grows exponentially with T_b, with higher rates of rise associated to higher values of q_{tb}. Higher values of T_b determine higher values of NU; this influence decreases with decreasing values of q_{tb}.

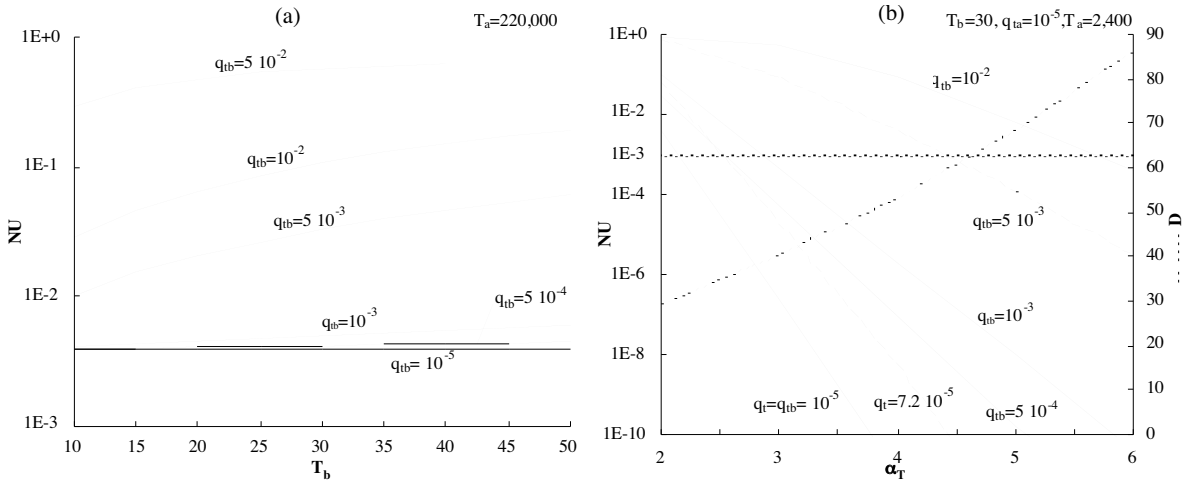

Figure 8. Values of NU against T_b, q_{tb} (a), and against α_T, q_{tb} (b)

As expected, in Figure 8(b) the values of NU worsen with increasing values of q_{tb}, for any given value of α_T. Note that, even in the pure exponential case, if a target value for NU is given, it can be attained by choosing a mating value for α_T, picked up from the Figure 8(b). Lower values of NU require higher values of α_T. This situation gets worse in presence of bursts: a measure of this effect can be observed by comparing the curves labelled with $q_{tb}=5*10^{-3}$ and $q_t=7.2*10^{-5}$, plotted for distributions having the same mean. Moreover, for increasing values of q_{tb}, a desired value of NU requires increasing values of α_T (in the figure, the example of NU $=10^{-3}$ is shown). So far, so good; unfortunately, however, higher values of α_T bring higher values of D, with the attending higher risks in running the system 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 point that it can become hard to find a satisfying trade-off in the original α-count mechanism. A way out has been worked out, as described in the next Sections, around the idea of decoupling the negative 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 run application programs just for diagnostic purposes, that is, its output will only be gathered by the error signalling subsystem, and its associate α-count mechanism will continue to operate. In other words u is considered "suspected", 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 affected by a permanent/intermittent fault, and the α- signal is issued, as in the original α-count, to the fault passi-

vation subsystem. When instead α, after some wandering in the (α_H - α_L) "limbo", becomes lower than or equal α_L[2], then the "suspected" component will be brought back in full service.

With this scheme, the lower threshold can be kept at a low level, to limit the detection delay XD, while losing not too much on the probability of throwing away good components. Actually, HCs being hit by occasional fault bursts may be counted off into a non-utilisation period (i.e. until their α-count goes back under the lower threshold), but the probability of irrevocable ousting can be kept as low as desired by means of an appropriately high value for α_H[3]. Another adverse effect, which has to be taken into account in the dimensioning of the system, is due to the necessity of treating the errors showing up in the limbo dwellers: in fact, for continued observation of their behaviour, they need to be nursed exactly as the on-line components, putting their share of burden on the error-processing subsystem. The models to be used in the analysis of the double-threshold mechanism are introduced below.

5.2 Double-threshold α–count acting on a HC

H_SAN-3, in Figure 9, models the behaviour of the double-threshold α–count acting on a HC. H_SAN-3 is an extension of H_SAN-2; as in the latter, a bursty fault behaviour is modelled, and the computation of an upper bound $\overline{NU}$ to NU, the utilisation 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 "suspected" (one token in *susp*) with those where the component is used and provides results to the user (no tokens in *susp*),

---

[2]In practice, a threshold value α_{L0}, with $\alpha_{L0} \le \alpha_L$, 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.

[3]An infinite value would, in fact, keep all units running, whether in the "limbo" or not.

2) the gate *incr* has been given the additional tasks 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; and

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).

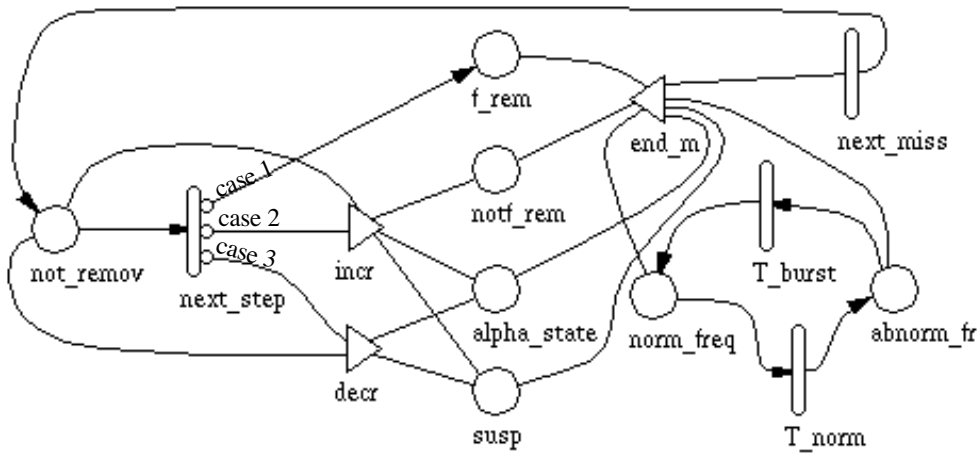

Figure 9. H_SAN-3 Double-threshold α–count acting on a HC, with transient faults bursts.

NU 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, determined as the steady state probability of "#not_remov=1 and #susp=0" divided by the steady state probability of "#notf_rem=1 or #f_rem=1":

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

Then:

$$\overline{NU} = \frac{\frac{1}{(q_i + q_p)} - \frac{P(\#\,not\_remov = 1 \text{ AND } \#\,susp = 0)}{P(\#\,not\_remov = 0)}}{\frac{1}{(q_i + q_p)}} = 1 - \left(\frac{P(\#\,not\_remov = 1 \text{ AND } \#\,susp = 0)}{P(\#\,not\_remov = 0)} \bullet (q_i + q_p)\right)$$

5.3 Double-threshold α−count acting on a FC

The behaviour of the double-threshold α−count acting on a FC, considering time dependent error rates of intermittent faults, is modelled using F_SAN-3 (see Figure 10), a SAN derived from F_SAN-2 (see Section 4.1) which allows to compute lower bounds for XD, the number of steps the FC is kept in full service after the permanent/intermittent fault occurrence. As done for F_SAN-2, the analyses have been limited to a finite time interval of length Max_d.

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 being in the state "not suspected") 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 is attained, or 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 re-setting to 1 the number of tokens in *count_notsusp* .

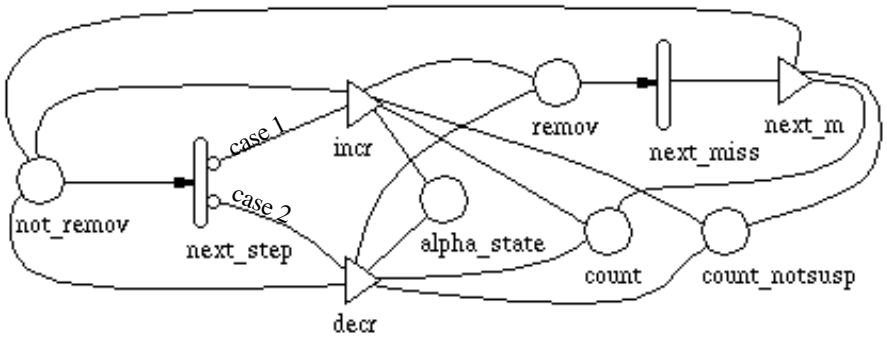

Figure 10. F_SAN-3, Double-threshold α−count acting on a FC under time-dependent error rates of intermittent faults

F_SAN-3 allows to compute a lower bound $\bar{F}_D(d)$ for d≤Max_d of the CDF F_D of XD as follows. Let:

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

where YD is the number of steps for α_inf to reach α_H. Let also:

$\bar{F}_D(d) = P(XD \le d | YD \le Max\_d) \cdot P(YD \le Max\_d)$ be a lower bound to $F_D(d)$ for d≤Max_d.

To obtain a closed expression for $\bar{F}_D(d)$ 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:

$$V = \begin{cases} \#\,count & \text{if } \#\,removv = 1 \\ 0 & \text{if } \#\,removv = 0 \end{cases} \qquad W = \begin{cases} \#\,count\_notsusp & \text{if "}\#\,removv = 1\text{" and "}\#\,count \le Max\_d\text{"} \\ 0 & \text{otherwise} \end{cases}.$$

and $F_W(d)$ be the CDF of the variable *W* , $F_V(d)$, the CDF of the variable *V*. Then it is easily seen that:

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

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

To check how strict is the lower bound thus obtained, 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 in Figures 9 and 10 have been analytically evaluated by UltraSAN, and estimations for the measures $\overline{NU}$ (indicated with NU for short) and $\bar{F}_D(d)$ (indicated with $F_D(d)$) have been derived. The basic parameter settings are still those reported in Table 1, if not otherwise specified. The values of new parameters pertaining the double-threshold α–count are indicated directly in the figures.

To discern the added features of the double-threshold scheme over the original single-threshold α-count, consider what could be the typical procedure a system designer would follow to set up the internal parameters of the mechanism. Given the external parameters, i.e. the fault rates and

their characterisations, in the single-threshold case a near-optimal value of K can be determined from Figure 3 (around the knee in the curve of D). Then, a figure like Figure 8 can be generated, where the target values of NU and D are plotted against the remaining internal parameter α_T: as it can be observed, their values are to be traded off each other, according to their relative importance in the system design, with no slack available to the designer.

Now, the performance figures resulting from the application of the single-threshold α-count are compared with those obtainable by the double-threshold variant. In the considerations that follow, the range $[\alpha_L, \alpha_H]$ is always assumed to encompass the value of α_T. A proper setting for the parameters α_H and α_L has to be chosen. Observing that high values of α_T give low values of NU suggests to start picking up from Figure 8 a (probably high) value for α_T which fits the target NU, and assigning it to α_H. Then, the value to be assigned to α_L has to come out as a good trade-off between a quite light negative influence on the utilisation factor of a HC and a satisfactorily low value for the probability to keep on-line a FC. This is what is analysed in the next two Figures 11 and 12.

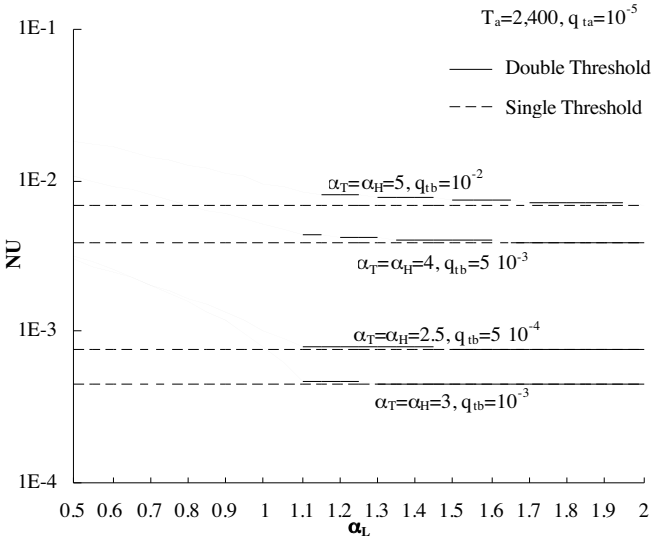

Figure 11. Values of NU against α_L for different values of α_H and q_{tb}, and comparison with the corresponding single-threshold case whith $\alpha_T=\alpha_H$.

Figure 11 shows the effects of the variation of the lower threshold on the utilisation loss NU of a HC . In this figure, four couples of plots have been depicted, each couple being formed by 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}) have been chosen (by inspection of Figure 8 (b)) to give values of NU around 10^{-3}, considered significant for typical applications.

Note that, since $\alpha_T=\alpha_H$ in each plot couple, the difference between the two curves displays how much is lost of a HC utilisation, wrt using the single threshold with the "best" threshold value in the range [α_L, α_H]. It is apparent that the utilisation loss is negligible in this respect with values of the lower threshold as low as 1.1. Around this value the curves exhibit a pronounced knee, which comes to no surprise, since the elemental increment in α-count is, in fact, 1. A first-approximation choice for α_L is then around 1.1. Anyway, at even lower values of α_L there is no dramatic loss, e.g. for α_T =3, (and $q_{tb}=10^{-3}$) at α_L = 0.5 the utilisation loss is around $4*10^{-3}$: with such a low α_L setting, most permanent faults would result into the neutralisation of the affected component with virtual no delay.

Having set all internal parameters, it only remains to check if the probability to keep on-line a faltering component has been lowered enough. This can be seen from Figure 12, where F_D is plotted against d for different α_L, with α_H=3 and Max_d=90.

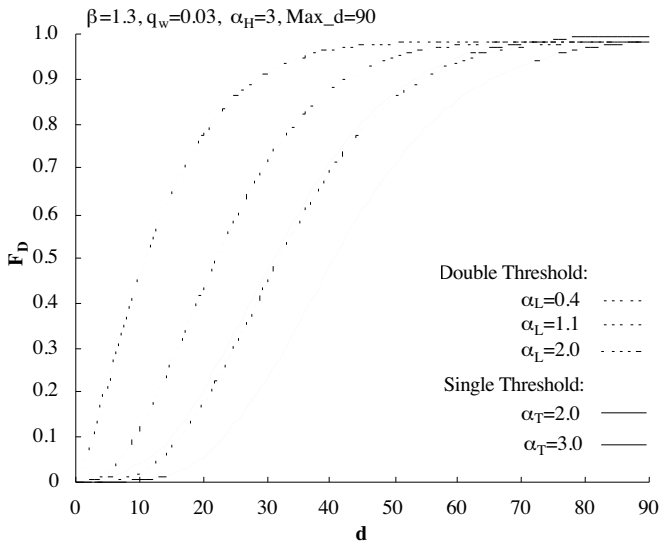

Figure 12. Values of 1-F_D against d for different α_L

A refinement step can be started here, looking up the benefits which would be gained by further shrinking α_L: if a smaller value seems worth the effort, the price in terms of NU can be looked up again in Figure 11, and so on.

Other considerations about Figure 12 follow. First, the plot relative to α_T=3 allows to appreciate the improvement of the double-threshold α-count over the single-threshold wrt the risky utilisation of a FC when α_T=α_H (which is the best assignment to α_T wrt the utilisation loss of a HC, as derived from the previous Figure 11). As expected, the lower the value of α_L, the higher is the improvement in the value of F_D. For example, in correspondence of d=40 the probability of still relying upon a FC is about 0.55 for the single-threshold, with α_T=3, while that exhibited by the double-threshold with α_H=3 and α_L=1.1 is approximately 0.15. In Figure 11 for this same setting of α_H, α_L and α_T the loss of utilisation of a HC is negligible; it can be concluded, in this case, that the double-threshold is clearly more appropriate in critical application contexts.

Second, the difference between the plot relative to α_T=2 and that relative to α_L=2 represents the loss incurred by the double-threshold, in terms of F_D, wrt the single-threshold enjoying the best α_T setting (in the range [α_L - α_H]) for F_D. As seen in Figure 12, while the single-threshold looks better, such difference is negligible. By the way, such a "better" behaviour regarding F_D is purely speculative, as it implies worsening NU.

6 Related work

As recalled in the Introduction, the necessity of discriminating transient faults to prevent excessive downtime in computing (sub)systems has spawned several techniques. A selection of those presented in the literature will be considered in the sequel, broadly classified in two groups: (A) techniques designed to support human intervention, and (B) algorithm-based, automatic mechanisms. A few of them will be directly related with α-count for the sake of simple comparisons.

(A) This group includes the more capable solutions (in terms of diagnostic resolution); they mainly entail the analysis of complex, bulky error logs, by means of (possibly) sophisticated

statistical analysis tools. Their procedures, however, are seldom expressed in algorithmic form, and require human supervision. Of course, this means that they are normally applied off-line.

In [15] trend analysis upon system error logs is applied, trying 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 [7] some heuristics, collectively dubbed 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 inter-arrival patterns of the failures (that can be time-varying). Among these, for example, there is the 2-in-1 rule, which signals a warning when the time of inter-arrival of two failures is less than 1-hour, and the 4-in-1 rule which fires when four failures occur within a 24-hour.

In [4] a comprehensive and sophisticated methodology for the automatic detection of permanent faults is presented. Error rate is used to build up error groups, i.e. sets of errors occurring in a time interval where an higher than normal rate is observed. The main part of the subsequent analysis exploits the quite detailed information given by the error record structure, available in the error logs generated by the targeted computer systems: simple probabilistic techniques are recursively used to seek similarities (correlations) among records, that eventually 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.

It has to be noted that the system logs routinely show detailed error reports: instead of simply giving a crash notice, specific codes are used to identify the (sub-)component involved, the type of the event, the application process active, the system time, and so on. This of course enables sophisticated analyses, possibly allowing more precise diagnosis, or permanent failure prediction; the point is, how to rate the performance of such complex analyses? The choice made for α-count leans exactly to trade complexity against predictability, accepting a simple mechanism for the sake of allowing performance figures to be forecast. Nevertheless, α-count could be used as a basic tool to implement a possibly approximated version of the above

heuristics. For example, the 2-in-1-like rule [7] 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) two failures are considered too many within, and Δ, $0<\Delta<1$, is a parameter to represent the accuracy of the approximation to the 2-in-1-like rule. By assigning 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 immediately precedent time unit the value of α associated to u is greater than Δ. With the same assumptions as above, it is easily seen that parameters of the single-threshold α-count satisfying the relation $\left(\left(K^y+1\right)K^y+1\right)K^y+1=\alpha_T$ make up an approximation of the 4-in-1-like rule [7] being in this case $4\cdot y$ the trigger frame.

(B) A number of mechanisms, based on counting/windowing algorithms simple enough to be implemented in a real system, operating in the field, are reported in this group.

In TMR MODIAC, the architecture proposed in [8] and analysed in [9], two failures experienced in two consecutive operating cycles by the same hardware component being part of a redundant structure make the other redundant components to consider it as definitively faulty.

In the 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 [14], describing the automatic diagnostics of the IBM 3081, the idea is used to trigger the intervention of a maintenance crew; in later models, as in the ES/9000 Model 900 [13], a retry-&-threshold scheme is adopted to cope with transient errors.

In [5] if, after a fault, a channel of the core FTP can be restarted successfully (restoring its internal state from other operating channels), then it is brought back in operation; “however, it is assigned a demerit in its dynamic health variable; this variable is used to differentiate between transient and intermittent failures”.

In [1] a list of “suspected” processors is generated during the redundant executions; then, a few schemes are suggested for processing this list, from taking down the processors in it for off-line diagnosis, to assigning weights to processors participating in the execution of a job and

failing to produce a matching signature with that of the accepted result, and taking down for diagnostics those whose weight exceeds a certain predetermined threshold.

All of the solutions just recalled implement schemes simpler than α-count; in fact, when given proper parameters their functionalities may easily be obtained by an α-count instance, e.g. the approach proposed in [8] can be emulated by the single-threshold an α-count with K=0 α_T=2.

7 Conclusions

A family of mechanisms, collectively named α-count, have been introduced in this paper to discriminate intermittent and permanent faults against low rate, low persistency transient faults. They are mainly characterised by: a) tunability through internal parameters, to warrant wide adaptability to a variety of system requirements; b) generality with respect to the system context, in which they are intended to operate, to ensure wide applicability; c) simplicity of operation to allow high analysability through simple models. The behaviour of α-count has been extensively analysed, in terms of performance figures, to be used in the framework of the whole system design.

Embedding mechanisms inspired to the α-count scheme in the design of a real system will give both the occasion to gauge its efficacy on the field, and to test the ductility conferred by its tuning parameters. A field trial of α-count is expected to be carried out in the context of the Esprit GUARDS project, where a generic multiprocessor architecture for real-time, dependable systems is under development. Error signals will feed instances of α-count applied to several components thus providing field data.

Other uses of α-count, taken as a system building block, can be devised. As an 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 all module's α-count values, looking for the dreadful correlated errors: a common pattern of rising counts on several dials could be for example an alarming symptom.

As a final remark, α-count is able to discriminate between event streams originated by different causes, provided they exhibit sufficiently different characteristic rates. Many other uses of the mechanisms may therefore be conceived: remaining in the field of dependable systems, α-count could be used to track errors *per se* (i.e. not to hunt faults), to ponder whether or not trigger a costly error processing protocol (normally requiring time-consuming system restart and state restoration procedure).

Acknowledgements

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, especially David Powell and Christophe Rabejac.

References

[1] P. Agrawal, "Fault Tolerance in Multiprocessor Systems without Dedicated Redundancy," IEEE Transactions on Computers, Vol. C-37, pp. 358-362, 1988.

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

[3] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico and F. Grandoni, "Discriminating Fault Rate and Persistency to Improve Fault Treatment," in Proc. 27th IEEE FTCS - International Symposium on Fault-Tolerant Computing, Seattle, USA, 1997, pp. 354-362.

[4] R. K. Iyer, L. T. Young and P. V. K. Iyer, "Automatic Recognition of Intermittent Failures: An Experimental Study of Field Data," IEEE Transactions on Computers, Vol. C-39, pp. 525-537, 1990.

[5] J. H. Lala and L. S. Alger, "Hardware and Software Fault Tolerance: A Unified Architectural Approach," in Proc. 18th International Symposium on Fault-Tolerant Computing, Tokyo, Japan, 1988, pp. 240-245.

[6] J. C. Laprie, "Dependability - its Attributes, Impairments and Means," in "Predictably Dependable Computing Systems", B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood Ed., Springer-Verlag, 1995, pp. 1-28.

[7] T.-T. Y. Lin and D. P. Siewiorek, "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis," IEEE Transactions on Reliability, Vol. 39, pp. 419-432, 1990.

[8] G. Mongardi, "Dependable Computing for Railway Control Systems," in Proc. DCCA-3, Mondello, Italy, 1993, pp. 255-277.

[9] M. Nelli, A. Bondavalli and L. Simoncini, "Dependability Modelling and Analysis of Complex Control Systems: an Application to Railway Interlocking," in Proc. EDCC-2 European Dependable Computing Conference, Taormina, Italy, 1996, pp. 93-110.

[10] W. H. Sanders and J. F. Meyer, "A Unified Approach for Specifying Measures of Performance, Dependability and Performability," in "Dependable Computing for Critical Applications, Vol. 4: of Dependable Computing and Fault-Tolerant Systems", A. Avizienis and J. Laprie Ed., Springer-Verlag, 1991, pp. 215-237.

[11] W. H. Sanders, W. D. Obal, M. A. Qureshi and F. K. Widjanarko, "The UltraSAN Modeling Environment," Performance Evaluation Journal, special issue on Performance Modeling Tools, Vol. 24, pp. 89-115, 1995.

[12] D. P. Siewiorek and R. S. Swarz, "Reliable Computer System - Design and Evaluation," Digital Press, 1992.

[13] L. Spainhower, J. Isenberg, R. Chillarege and J. Berding, "Design for Fault-Tolerance in System ES/9000 Model 900," in Proc. 22th International Symposium on Fault-Tolerant Computing, Boston, Massachusetts, USA, 1992, pp. 38-47.

[14] N. N. Tendolkar and R. L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," IBM J. Res. Develop., Vol. 26, pp. 78-88, 1982.

[15] M. M. Tsao and D. P. Siewiorek, “Trend Analysis on System Error Files,” in Proc. 13th International Symposium on Fault-Tolerant Computing, Milano, Italy, 1983, pp. 116-119.

AFFILIATION OF AUTHORS

Andrea Bondavalli is with CNUCE-CNR, Via S. Maria, 36, 56126 Pisa, Italy;

Silvano Chiaradonna is with CNUCE-CNR, Via S. Maria, 36, 56126 Pisa, Italy

Felicita Di Giandomenico is with IEI-CNR, Via S. Maria, 46, 56126 Pisa, Italy

Fabrizio Grandoni is with IEI-CNR, Via S. Maria, 46, 56126 Pisa, Italy

For communications, contact F. Grandoni, IEI-CNR, Via S. Maria, 46, 56126 Pisa, Italy

ADDRESSES:

Andrea Bondavalli	Phone: +39-50-593327
CNUCE-CNR	Fax: +39-50-904052
Via S. Maria 36	E-mail :A.Bondavalli@cnuce.cnr.it
I-56126 Pisa, Italy	WebServ: http://bonda.cnuce.cnr.it
Fabrizio Grandoni	email grandoni@iei.pi.cnr.it
IEI-CNR	Phone (pers.) +39-50-593444
Via S. Maria,46	Phone (oper.) +39-50-593400
I-56126 Pisa, Italy	fax +39-50-554342
Silvano Chiaradonna	E-mail:S.Chiaradonna@guest.cnuce.cnr.it
CNUCE-CNR	Phone: +39-50-593331
Via S. Maria 36	Fax: +39-50-904052
I-56126 Pisa, Italy	
Felicita Di Giandomenico	email digian@ieiserv.iei.pi.cnr.it
IEI-CNR	Phone (pers.) +39-50-593443
Via S. Maria,46	Phone (oper.) +39-50-593400
I-56126 Pisa, Italy	fax +39-50-554342