This paper has appeared in the proceedings of 27th IEEE FTCS - International Symposium on Fault-Tolerant Computing.

Discriminating Fault Rate and Persistency to Improve Fault Treatment

A. Bondavalli*, S. Chiaradonna**, F. Di Giandomenico** and F. Grandoni**

* CNUCE Istituto del CNR, Via S. Maria, 36, 56126 Pisa, Italy,
** IEI Istituto del CNR, Via S. Maria, 46, 56126 Pisa, Italy

Abstract

In this paper the consolidate identification of faults, distinguished as transient or permanent/intermittent, is approached. Transient faults discrimination has long been performed 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 the count-and-threshold scheme, through the analysis of its behaviour and the exploration of its effects on the system. To this goal, the scheme is mechanized as a device named α-count, endowed with a few controllable parameters. α-count tries to balance between two conflicting requirements: to keep in the system those components that have experienced just transient faults; and to remove quickly those affected by permanent or intermittent faults. Analytical models are derived, allowing detailed study of α-count's behaviour; the actual evaluation, in a range of configurations, is performed by standard tools, in terms of the delay in spotting faulty components and the probability of improperly blaming correct ones.

1 Introduction

The problem of handling the consequences of faults in computing systems is not limited to special, high reliability systems: in any environment a specified, albeit feeble, level of operability is to be assured, in spite of adverse internal or external conditions. In fact, even the commonplace general purpose departmental computing system, while by no means equipped to automatically survive to hardware or software faults, nonetheless is expected to provide "reasonably" continuous service: upon a crash, the user is accustomed to wait several minutes or so, after that the system is supposed to go up again (thanks, operator!). This is tantamount to state that, upon the occurrence of a fault, some action to treat this condition has to be taken (whether by humans or by automatic mechanisms) to heal the system. In systems where dependability figures are explicit and important requirements the treatment of faults needs to be carried out in a systematic way: in fact, this is a hefty part of fault-tolerant systems theory.

Fault treatment [5] consists of fault diagnosis and fault passivation. Fault diagnosis has the purpose of locating the source of the fault, i.e. the (hardware or software) component(s) affected, and understanding the nature of the fault. Fault passivation is carried out by disallowing the component identified as being faulty further execution, possibly repairing or replacing it. If error processing means are employed, sufficient to cope with the fault (e.g. in case of transient faults, provided their likelihood of recurring is low enough), passivation may be omitted. After reconfiguration, the system may be no longer capable of delivering the same service as before, and may offer a degraded service.

In [5] 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, owing to their short persistency, their adverse effects rapidly disappear, and, provided their frequency does not exceed some system-specific threshold, no further action is taken, other than restarting the system or masking the resulting error if fault-tolerance is provided.

In most application fields a large fraction of faults experienced by current computing systems is transient in nature [11]. Since the relative cost of treatment is very different, it is important to recognise such a characterisation. However, in many systems it is difficult to distinguish on the spot permanent from transient faults.

An idea which has long been exploited, e.g. in several IBM mainframes (see below), has been to count the number of fault events up to a given threshold, whose crossing would be used to consider permanently faulty the system component at hand.

Actually, a wide range of techniques spanning from simple retry to rather sophisticated off-line error log audit and trend analysis have been used, or studied in the literature, with reference to specific systems.

The present work aims to contribute to the usefulness of the count-and-threshold scheme, through the analysis of its behaviour and the exploration of its effects on the system. To this goal, the scheme is mechanized as a device named α-count, endowed with a few controllable parameters. Analytical models are derived, allowing detailed study of its behaviour; the actual evaluation, in a range of configurations, is performed by standard tools.

The rest of the paper is organised as follows. In Section 2, several solutions presented in the literature are reviewed, and the system context where α-count is intended to operate is given. Section 3 is devoted to the definition of α-count. In Section 4 the models of α-count behaviour are introduced. Numerical evaluation through identified figures of merit is conducted in Section 5. An exercise on implementing known heuristics by means of α-count is carried out in Section 6. Finally, conclusions and directions for future work are given in Section 7.

2 Problem statement

2.1 Background

Some significant transient-faults screening techniques are recalled below, broadly classified into two groups: (A) on-line mechanisms and (B) off-line analysis procedures.

(A): In TMR MODIAC, the architecture proposed in [7] and analysed in [8], two consecutive failures experienced by the same hardware component being part of a redundant structure make the other redundant components to consider it as definitively faulty.

[13], describing the automatic diagnostics of the IBM 3081, shows up how the idea of thresholding the number of errors accumulated in a time window is used to trigger the intervention of a maintenance crew. [12] reports how a retry & threshold scheme is adopted in the IBM ES/9000 Model 900 to dodge transient errors.

In [4] 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.

(B): In [14] trend analysis upon system error logs is applied, in 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 [6] 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 [3] 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.

2.2 System context

The techniques reviewed above are given as solutions to specific applications, characterised by different dependability requirements, from plain general purpose computing centers to safety-critical systems. Those in the (B) group are way more powerful of those in the (A) group; however, because of their complexity they have been applied in off-line diagnosis and their applicability to on-line use is only envisioned by their authors for future work.

The α-count mechanism proposed here aims to be applied on-line in a wide set of application fields, with the attending dependability requirements. It can be adapted to different application requirements via parameters geared to tune its behaviour to different system figures, such as CPU or input devices fault rates. The definition of α-count will be given with reference to highly dependable, fault tolerant systems, since less demanding ones may be seen as simplified instances thereof.

Like in several well known threshold schemes, α-count gathers information about erroneous behaviour of each system component as time goes on, from any available error detection device, weighing down error signals as they get older. The rationale is to decide the point in time when keeping a system component on-line is no longer beneficial; if the rate of detected errors, filtered according to some tuning parameters, exceeds a proper threshold, a signal is raised toward a fault passivation subsystem to take action.

The basic concept of α-count has been presented in [2]. There the mechanism had a slightly different definition: it was considered in a specific fault-tolerant context, paired to the error processing structure, and the evaluation of the whole resultant structure was carried out, by means of simulation techniques in a restricted system configuration, with no provision to single out performance figures for the mechanism alone. Here a general formulation of α-count is given, without any mandatory reference to the surrounding system, other than the assumptions listed below. The mechanism is modelled and evaluated, and its performance is given in terms of specific figures of merit, which will be later introduced.

Computing systems are normally equipped with several low-level error detection mechanisms, whether in the value domain (e.g., overflow checking, divide-by-zero) or in the time domain (e.g., time-outs, watchdog timers). 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 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 agreement algorithms, augmented with disagreement signals. The α-count mechanism may be easily plugged in all the above sort of systems.

The description of α-count will be given with reference to a model, where the system is partitioned into units enclosed in well-defined boundaries, which may be pointed to as affected by faults; some error signalling mechanism is available, able to raise fault hints on misbehaving units; units are activated at regular points in time, and each activation has a fixed duration. A fault passivation subsystem is present, which is in charge of processing the signals coming from α-count. Faults manifest themselves according to the following assumptions:

1- Hardware faults only are considered. A component affected by a permanent fault produces an incorrect service at each activation; a component affected by an intermittent fault does repeatedly give rise to an error with a probability q at each activation; a component affected by a transient fault produces an incorrect service, but the fault lasts only for one activation.
2- The faults affecting the hardware supporting the α-count are not considered.
3- During each activation, a unit may be affected at most by a single fault related to a given phenomenological cause; therefore, a unit experiencing an intermittent fault may also be hit by a transient fault, but not by a permanent fault.
4- The judgement on the components behaviour given by the error signalling mechanism is correct with a probability c, which depends on the scheme used by this mechanism to boil down its judgement. Each judgement is assumed independent in the probabilistic sense from the others.
5- In modelling and evaluating the α-count mechanism, the probabilities q_p, q_i and q_t of the occurrence of a permanent, intermittent or transient fault, respectively, during a unit activation, are assumed constant in time for any given unit. Fault occurrences in different activations are assumed statistically independent each other.

3 A threshold-based fault identification mechanism

The requirements for α–count can be stated as:

i) signal all components affected by permanent or intermittent faults (referred to in the following as faulty units, or FUs) as quickly as possible, by sending a message to the fault passivation subsystem;
ii) avoid signalling units other than FUs (referred to in the following as healthy units, or HUs). HU are so

designated because of the external, temporary nature of transient faults.

3.1 Basic description

The judgement issued by the error signalling mechanism on the behaviour of the generic component u_i is symbolically expressed as a binary value. Let $J_i^{(L)}$ indicate the L-th judgement on the generic component u_i; then $J_i^{(L)} = 0$ means success while $J_i^{(L)} = 1$ means failure. The α-count keeps track of fault occurrences in each component while execution proceeds. A score α_i is associated to each not-yet-removed component u_i to record information about the failures experienced by that component. α_i is initially set to 0, and accounts for the pertinent L-th judgement as follows:

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

When the value of $\alpha_i^{(L)}$ grows bigger than or equal to a given threshold α_T, the component u_i is diagnosed as affected by a permanent/intermittent fault; this event is signalled to the fault passivation subsystem. The effects of component removal upon system performance and reliability figures depend on the parameters K and α_T. As a first flavor of their meaning, note that $\lceil \alpha_T \rceil$ represents the minimum number of consecutive failures sufficient to consider a component permanently faulty, while K represents the ratio by which α_i is decreased after a success.

The values to be assigned to the parameters so that the strategy works best, i.e., it recognises FUs as soon as possible and lowers the probability of identifying HUs as faulty, depend on the expected frequency of permanent, intermittent and transient faults and on the probability c of correct judgements of the error signalling mechanism used, as it will be shown below.

3.2 Basic properties

The features of α–count with respect to its ability to fulfil its requirements will now be discussed. To start, it is easy to show that the α–count mechanism is asymptotically able to identify all components affected by permanent or intermittent faults.

More formally, if α_T is set to any finite positive integer A, and the component u_i is a FU, then α_i will eventually grow bigger than or equal to A. This corresponds to identify the component u_i as faulty (with the consequent signal to the fault passivation subsystem). In fact, a sufficient condition for a generic item $J_i^{(L)}$ to be 1 is that a fault has been activated in u_i and the judgement issued by the error signalling mechanism is correct.

Since intermittent faults, by their nature, are more difficult to discover than permanent faults (they alternate incorrect and correct behaviour according to the probability q), this worst case will be conservatively assumed. Recall that independence has been assumed both among successive activations of an intermittent fault and among the judgements provided by the error signalling mechanism in different executions. Then, upon the occurrence of an intermittent fault in u_i, the sequence of judgements starting from $J_i^{(\bar{L})}$ (the first judgement after the occurrence of the fault) is composed of independent binary values each having at least a probability $q \cdot c$ to be a 1.

A (trivially) sufficient condition for α_i to grow bigger than or equal to A, is to observe A consecutive 1's. This condition becomes also necessary if the parameter K is set to 0. Having no limits on the length of the sequence, obviously a sequence of A consecutive 1s will eventually be observed.

Using a similar argument it can be shown that α–count cannot guarantee avoiding to signal some HU as faulty, i.e., a (normally small) probability exists that a HU will be identified as faulty and the fault passivation mechanism improperly invoked.

4 Analytical modelling of α–count

Besides the above asymptotic results, performance figures of interest in practice are: i) the average delay D_i (expressed in terms of number of activations of u_i) from the (permanent or intermittent) fault occurrence in u_i to its identification and signalling, and ii) the probability HR_i that a HU u_i be (wrongly) identified as FU.

To evaluate D_i and HR_i, the behaviour of α-count has been modelled by means of the Stochastic Activity Networks (SAN) [9] D_SAN and HR_SAN, respectively. In SAN models, such as D_SAN and HR_SAN, real valued variables, like α_i, need to be represented by a discrete approximation which can be chosen as close to α_i as desired at the price of increasing the time and computational effort necessary to obtain the solution.

4.1 An upper bound to the expected delay in identifying a FU

D_SAN, the SAN depicted in Figure 1, represents the behaviour of α–count acting on the FU u_i. Inside D_SAN, α_i is represented by the discrete variable α_inf_i taken as the value of α_i, truncated to the 4th decimal figure. Being $\alpha\_inf_i \le \alpha_i$, α_inf_i will then reach α_T in an average number of steps $\overline{D}_i$ higher than, or equal to, D_i the average number of steps for α_i to reach α_T.

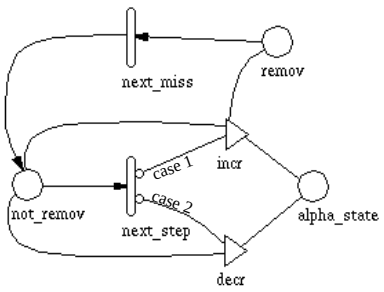

Figure 1. D_SAN: model for evaluating $\overline{D}_i$

D_SAN has three places: i) *not_remov* (the presence of a token in this place means that u_i is still active in the system); ii) *remov* (a token here signifies the actual identification of u_i as FU, with the attending emission of the *remove* message); iii) *alpha_state* (used to maintain the current value of α_inf_i). The exponentially distributed timed activity (with rate 1) *next_step* represents the issue of a judgement; it has two cases: *case 1*, representing the occurrence of $J_i^{(L)} = 1$, and *case 2*, representing that of $J_i^{(L)} = 0$.

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 *remove* message; *next_miss* is also exponentially distributed with rate 1. The two output gates *incr* and *decr* perform the basic steps for computing α_inf_i. When α_inf_i reaches the value α_T, the gate *incr* deposits a token in the place *remov* and sets to 0 the value of α_inf_i for a proper restart in *next_miss*.

Since D_SAN represents the case of a FU, the component u_i is affected either by a permanent or by an intermittent fault. The case of permanent fault is trivial: the probability of $J_i^{(L)} = 1$ is very close to 1, hence the expected delay in identifying u_i as FU is approximately equal to α_T. In the worst case, given by the occurrence of intermittent faults, we have:

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

The first term of the first expression represents a failure of u_i (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 a success wrongly detected as a failure.

The value of $\overline{D}_i$ is determined by the ratio between the steady-state probabilities of the places *not_remov* and *remov* 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 FU.

4.2 An upper bound to the probability of identifying a HU as FU

HR_i, the probability of identifying a HU, u_i, as faulty, can be thought of as the probability that the number of steps to the tagging of u_i as a FU taken by the α–count mechanism is less than the number of activations before u_i really becomes a FU, i.e. is affected by a permanent or an intermittent fault.

The SAN depicted in Figure 2, HR_SAN, represents the behaviour of α–count acting on the HU u_i and allows to compute an upper bound $\overline{HR}_i$ to HR_i. α_i is represented in HR_SAN by the discrete variable α_sup_i, chosen as the nearest four-decimal figure value greater than α_i. Being $\alpha\_sup_i \geq \alpha_i$, α_sup_i will then reach α_T in an average number of steps lower than, or equal to, the average number of steps for α_i to reach α_T. Therefore, by using a lower approximation of the number of steps to the identification of a FU, an upper bound $\overline{HR}_i$ to HR_i is obtained.

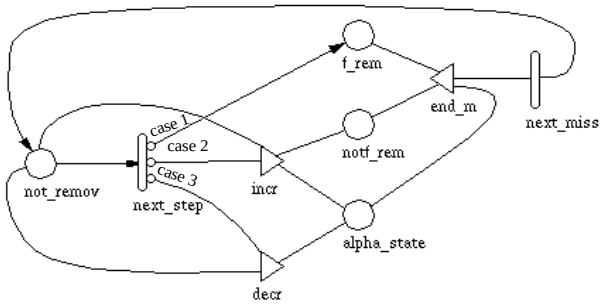

Figure 2. HR_SAN: model for evaluating $\overline{HR}_i$

HR_SAN has four places. Places *not_remov* and *alpha_state* have the same meaning as in D_SAN (*alpha_state* maintains the current value of α_sup_i), whereas two places are used to distinguish whether the component identified as being a FU is really subject to an intermittent/permanent fault or not. Place *f_rem* holds a token when the component becomes a FU, place *notf_rem* holds a token when α_sup_i crosses the threshold α_T while u_i is still a HU. Activities *next_step* and *next_miss*, which are exponentially distributed timed activities with rate 1, represent, as in the previous SAN, the issue of a judgement and the restart of α–count upon the issuing of the *remove* message, respectively. The two output gates *incr* and *decr* compute α_sup_i and check its value against the threshold α_T.

In HR_SAN the activity *next_step* has three cases. *case 1* represents the occurrence of a permanent or in-

termittent fault in u_i, which thus becomes a FU, *case 2* and *case 3* represent respectively the issue of $J_i^{(L)} = 1$ and of $J_i^{(L)} = 0$ when u_i is an HU.

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

The component u_i is identified as a FU when a token is either in *notf_rem* or *f_rem.* Thus the value of $\overline{HR}_i$ is determined by computing the ratio between the steady-state probability that one token is in the place *notf_rem* and the steady state probability of having a token in either *notf_rem* or *f_rem*.

5 Evaluation of α–count

The SAN models, depicted in Figures 1 and 2, have been analytically solved by using UltraSAN [10] to evaluate $\overline{D}_i$ and $\overline{HR}_i$. For the sake of simplicity in notation, D and HR will be used in the sequel in place of $\overline{D}_i$ and $\overline{HR}_i$, respectively; moreover they have been plotted together to help appreciate the effects of common parameters on both quantities.

Table 1 recalls the symbols used for all the parameters, together with their definition and the default values used in the plots unless explicitly specified.

Symbol	Definition	Value
q_t	probability of transient fault per execution	10^{-5}
$q_p + q_i$	probability of intermittent or permanent fault per execution	10^{-6}
q	probability of activation of an intermittent fault per execution, given that the component is affected by an intermittent fault	0.1
c	probability that a judgement given by the error signalling is correct	$1\text{-}10^{-5}$
K	the ratio by which α_i is decreased after a success	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

Among all the parameters that are into play, K and α_T are those internal to α–count, i.e., those 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. So the first part of the analysis, showing the behaviour of α–count at varying values of K and α_T, aims at providing insights on how to devise proper values for the internal parameters. Then the analysis is extended to the sensitivity to the other (external) parameters. This extension contributes to the normal multi-step refinement design process.

5.1 Tuning of parameters

Figure 3 shows the plots of D and HR for different values of α_T and varying K.

Observe first that, while increasing values of K improve D (which gets lower values) and worsen HR, the opposite effects may be observed regarding α_T. For 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.

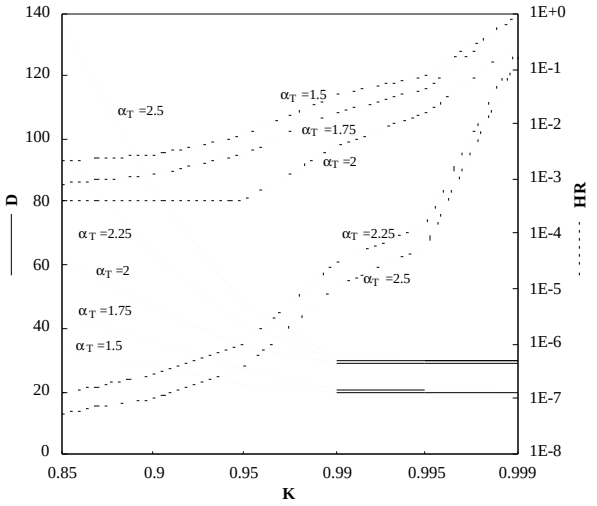

Figure 3. Values of D and HR as a function of K for varying α_T

HR grows very slowly for low values of K up to a region around a value K_{HR} whereas it becomes worse and worse for $K > K_{HR}$, 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 HR. If $K_D < K_{HR}$, values of K in the interval $[K_D, K_{HR}]$ determine values for D and HR close to their optimum. It appears thus that, if no other constraints are given, values for K should be chosen in such interval.

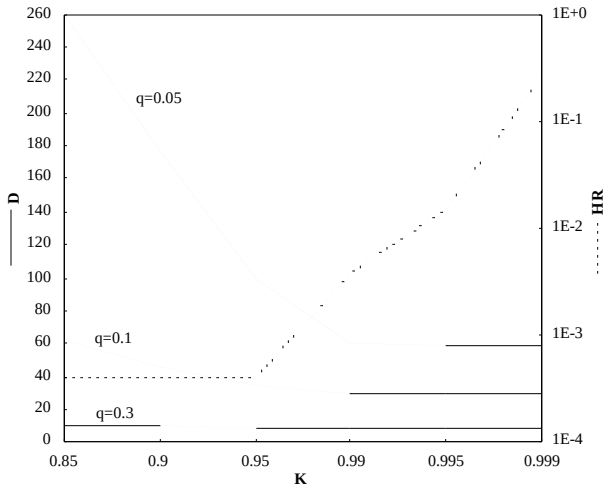

Figure 4. Values of D and HR as a function of K for varying q

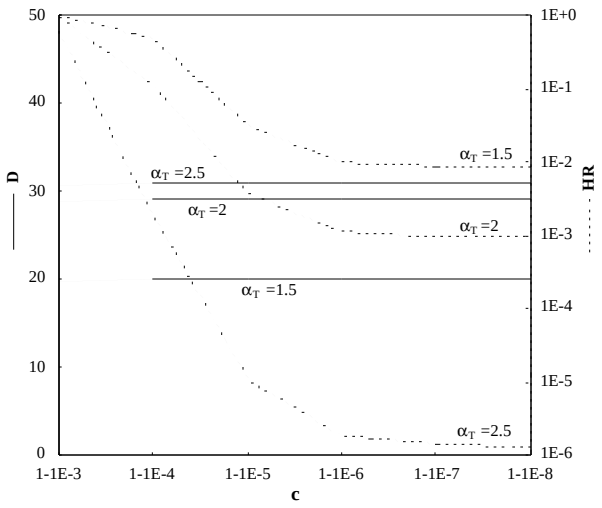

Figure 5. Values of D and HR as a function of c at varying α_T

However, the existence of such interval depends on other parameters as well. In fact, 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 HR and a low D are conflicting objectives, the definition of the "best behaviour" for α–count, thus the proper tuning of its parameters, requires the definition of their relative importance. So, returning to our example depicted in Figure 3, even though the range for values to be assigned to K has been identified, still it has to be decided whether D or HR 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 HR must take values lower than a given threshold".

5.2 Sensitivity to external parameters

Figure 5 shows the plots of D and HR as a function of c, for different values of α_T.

Increasing values of α_T (obviously) increase D and determine smaller values of HR. While c does not affect D, higher values of c lead to lower (better) values for HR even though this improvement becomes less important for higher values of c. It can be noted that the same value of HR can be obtained by different combinations of c and α_T. If the value of c was not high enough to reach the desired value for HR with a given α_T, a designer could set α_T to an higher value at the price of a worse D. Moreover HR improves at increasing values of c up to some value c^*, almost independent from α_T, after which HR becomes constant.

The value c^* is determined by the cumulative fault probability, $q_p + q_i + q_t$, as shown in Figure 6 where HR is plotted as a function of this probability (with a constant percentage of 91% of transient faults, as used in the previous figures) for different values of c. It is apparent to see that for any given value of the cumulative fault probability a value exists for c such that higher values of c determine no further improvement on HR. Figure 6 shows that for a given value for c (e.g. $1\text{-}10^{-6}$), the value for HR has an absolute minimum (corresponding to the value 10^{-6} of abscissae in the example). While it is clear that HR increases at increasing cumulative fault probability (more frequent errors are observed), it is much less clear why decreasing values of the abscissae determine higher HR. The reason comes from the definition of HR_i, that is the probability of identifying a HU as faulty. What happens is that, decreasing the cumulative fault probability, despite the time to the identification of a component as faulty becomes longer, the component is more likely a HU when this identification occurs. This is due to the constant value of c which becomes more and more pre-

dominant in the probability of observing errors. Thus, whenever the value for HR is close to the minimum for a given c, employing components with lower cumulative fault probabilities without improving c at the same time, may even turn to a worsening of HR.

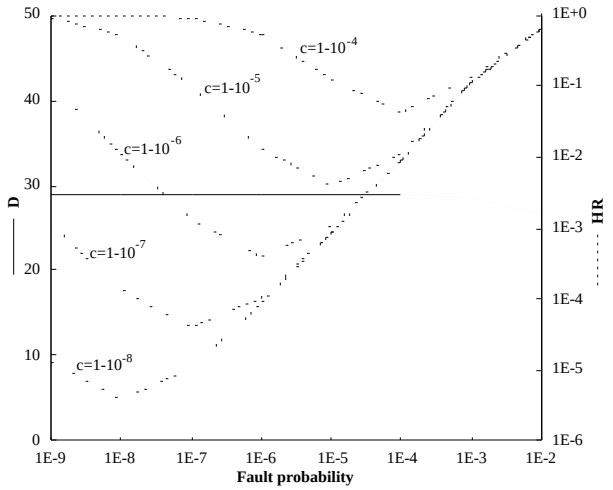

Figure 6. Values of D and HR as a function of the cumulative fault probability

Looking at the right end side of Figure 6, it can be seen that as the fault rate increases, even units experiencing only transient faults are tagged as FU, with a probability which grows toward unity. This suggests a more general characterisation of the mechanism not bound to hypotheses on the nature and timing distributions of faults. Actually α-count is simply based on some parameters, such as a threshold and averaging parameters, that make up a decision mechanism which fires whenever the number of errors signalled about a unit, accumulated and weighed in the recent past, reaches a value where the normal fault handling capabilities can be impaired and incorrect service provided. Therefore, whenever the discriminating criterion to decide if a unit is to be taken off-line is its error occurrence rate (regardless of the actual nature of the faults and of the distribution in time of their occurrence), α-count can be usefully employed.

6 Mapping of a few heuristics

The α-count, despite its simplicity, may be used in a range of applications. As an exercise, approximate implementations of three common-sense rules already presented in the literature to solve the same problem are shown below. A few heuristics, clearly expressible as mathematical formulas, are chosen (see Section 2.1 for the original formulations).

(1) The approach proposed in [7] can be easily emulated by α-count, setting K to 0 and α_T to 2.

(2) Assuming that the duration of an activation is one time unit, the 2-in-1-like rule [6] can be obtained 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 is a parameter to represent the accuracy of the approximation to the 2-in-1-like rule. Δ is to be set to a positive value, well smaller than 1. By assigning K the value computed with the above formula, when a second error hits component u_i after less than x successful activations, the threshold α_T will be exceeded, since at the immediately precedent activation the value of α_i is greater than Δ.

(3) With the same assumptions as above, it is easily seen that parameters of 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 [6] being in this case $4\cdot y$ the trigger frame.

7 Conclusions

In this paper, a mechanism designed to discriminate intermittent and permanent against low rate, low persistency transient faults, named α-count, has been described, with the aim of improving fault treatment, and so the overall system performance. α-count draws from the long practiced count-and-threshold idea, coupled with a simple decay algorithm to take into account the timing traits of intermittent faults.

The system context in which α-count is intended to operate has been kept very general: this mechanism provides parameters to tune its behaviour to system figures, to warrant wide applicability. A model for α-count has been derived and extensive analyses of its stochastic behaviour have been conducted in terms of the figures of merit D_i (delay to fault identification) and HR_i (probability of improper fault identification); the simplicity of the mechanism results into simple models.

The behaviour of faults in real systems sometime escapes the quite simple model defined in Section 2.2: e.g., the constant rate assumption is not the best as far as intermittent faults are concerned, as they tend to become more frequent after the first occurrence, until possibly be stuck into permanent faults. Even transient faults, in fact, may occasionally happen with abnormal frequency (e.g. in the course of a storm). In this regard, though, we have shown that the mechanism can well adapt to those cases where the discriminating criterion to decide if a unit is to be taken off-line is its error occurrence rate.

The quantities D_i and HR_i can not be considered to have a conclusive value in the determination of the overall dependability figures of a whole system. However, the analysis tools given above for α-count may be used to

embed the mechanism as a black box in the design of the encompassing system; the box will be characterised by its parameters D_i and HR_i. From the design goals of the system-at-large (e.g., dependability figures) the required values for D_i and/or HR_i should be derived, which in turn determine the α-count parameters K and α_T.

Directions for future work may be identified essentially along four lines. First, more extensive comparisons with heuristics developed to discriminate transient faults would help to prove how much generality the mechanism is to be credited, as well as to understand what improvements are necessary to capture real-world system characteristics. A second refinement would be to consider more realistic time distribution of fault occurrences. Of course, for a generic time distribution of the error events, it may be very difficult to obtain accurate evaluations making it hard to devise an optimal setting of the internal parameters, e.g. the setting of α_T may be required to be found by trial and error. Then we plan to release the assumption of considering only hardware faults. Last, embedding the α-count mechanism 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.

Acknowledgements

This work has been partially supported by the CEC in the framework of the ESPRIT 20718 GUARDS project. The authors are involved in this project through PDCC. The authors wish to thank the reviewers and Ram Chillarege for their valuable comments and suggestions.

References

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

[2] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico and L. Strigini, “Rational Design of Multiple-Redundant Systems: Adjudication and Fault Treatment,” in “Predictably Dependable Computing Systems”, B. Randell, J. C. Laprie, H. Kopetz and B. Littlewood Ed., Springer-Verlag, 1995, pp. 141-154.

[3] 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.

[4] J. H. Lala and L. S. Alger, “Hardware and Software Fault Tolerance: A Unified Architectural Approach,” in Proc. FTCS-18, Tokyo, Japan, 1988, pp. 240-245.

[5] 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.

[6] 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.

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

[8] M. Nelli, A. Bondavalli and L. Simoncini, “Dependability Modelling and Analysis of Complex Control Systems: an Application to Railway Interlocking,” in Proc. EDCC-2, Taormina, Italy, 1996, pp. 93-110.

[9] W. H. Sanders and J. F. Meyer, “A Unified Approach for Specifying Measures of Performance, Dependability and Performability,” in Proc. DCCA-1, 1991, pp. 215-237.

[10] 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.

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

[12] L. Spainhower, J. Isenberg, R. Chillarege and J. Berding, “Design for Fault-Tolerance in System ES/9000 Model 900,” in Proc. FTCS-22, Boston, Massachusetts, USA, 1992, pp. 38-47.

[13] 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.

[14] M. M. Tsao and D. P. Siewiorek, “Trend Analysis on System Error Files,” in Proc. FTCS-13, Milano, Italy, 1983, pp. 116-119.