Balancing System Availability and Lifetime with Dynamic Hidden Markov Models

Jacopo Panerati*, Samar Abdi†, Giovanni Beltrame*
*Département de Génie Informatique et Génie Logiciel, École Polytechnique de Montréal, Montréal QC, Canada
†Faculty of Engineering and Computer Science, Concordia University, Montréal QC, Canada
{jacopo.panerati, giovanni.beltrame}@polymtl.ca, samar@ece.concordia.ca

***Abstract*—Electronic components in space applications are subject to high levels of ionizing and particle radiation. Their lifetime is reduced by the former (especially at high levels of utilization) and transient errors might be caused by the latter. Transient errors can be detected and corrected using memory scrubbing. However, this causes an overhead that reduces both the availability and the lifetime of the system. In this work, we present a mechanism based on *dynamic hidden Markov models* (D-HMMs) that balances availability and lifetime of a multi-resource system by estimating the occurrence of permanent faults amid transient faults, and by dynamically migrating the computation on excess resources when failure occurs. The dynamic nature of the model makes it adaptable to different mission profiles and fault rates. Results show that our model is able to lead systems to their desired lifetime, while keeping availability within the 2% of its ideal value, and it outperforms static rule-based and traditional *hidden Markov models* (HMMs) approaches.**

I. Introduction

At high altitude or in space, without the protection of the earth's magnetic field and atmosphere, integrated circuits are exposed to high level of radiation and heavy ion impacts that can disrupt the correct circuits' behavior.

In this work we provide a mathematical framework to establish an adaptive system capable of managing both device aging (as accelerated by ionizing radiation) and *single event upsets* (SEUs), or soft errors, usually caused by the transit of a single high-energy particle through the circuit.

Detection of transient errors and protection against SEUs can be obtained in several ways [1], but there are no clear guidelines on how to identify permanent faults. One way to do it is to retry the same computation multiple times, and after a certain number of errors in a given interval of time, declare the component as permanently faulty [2].

This sort of on-line testing requires additional resources and inherently reduces the availability of the system. Moreover, the repetition of computation increases the strain on the electronics, reducing their lifetime. In this work we propose a framework that provides a system with the ability to decide, in case a fault is detected, if it is worthwhile to perform additional testing or if the component should be classified as permanently damaged.

The framework we introduce is a *failure detection mechanism* based on dynamic hidden Markov models . The main aspects of novelty of this work are:

- the extension of the static hidden Markov model framework with a dynamic transition model. This allows for the correct application of HMMs to the modeling of more general, *non-memoryless* failure processes;
- the D-HMM integration with a compact statistical modeling of transient and permanent faults occurring in electronic components exposed to ionizing and particle radiation;
- the definition of the lifetime-availability trade-off faced by failure detection mechanisms;
- the comparison, through simulation performed with real-life parameters, of the proposed D-HMM approach against traditional HMMs and rule-based systems.

The rest of the paper is structured as follows: we review relevant work in the field of fault detection in Section II; Section III presents the theoretical background needed to understand our methodology, which is exposed, along two simpler alternatives, in Section IV;finally, the setup used to evaluate the approach, the results discussion, and the conclusions are presented in Section V, Section VI, and Section VII, respectively.

II. Related Work

This section reviews some important research work in the field of fault detection, automated classification and recovery. We also cover work dealing with the modeling of fault occurrences and failure times.

The need for computing systems tolerant to both software and hardware faults is not a novel one: reliability engineering provides us with a plethora of tools, mainly statistical ones, for the definition and analysis of such systems.

Creating fault-tolerant systems for space applications, however, is particularly challenging because of the several different types of faults that can arise when computers operate outside the atmosphere. Smolens *et al.* [3] called the attention on the

This work was partially supported by the *Regroupement Stratégique en Microsystèmes du Québec* (ReSMiQ)

*École Polytechnique de Montréal, 2900 Boulevard Edouard-Montpetit, Montreal, QC H3T 1J4, Canada. ✆ +1 514-340-4711 # 7123

978-1-4799-5356-1/14/$31.00

fact that "aggressive CMOS scaling" results in an accelerated wear out of transistors and wires, and it inevitably leads to "shorter and less predictable lifetimes for microprocessors".

On the other hand, Karnik and Hazucha [4] studied how radiation particles interact with silicon and how these interaction should influence the design of VLSI systems. Radiation, in fact, can induce SEUs, impact system reliability, and it poses "a major challenge for the design of memories and logic circuits [...] beyond 90nm".

Cassano *et al.* [5] observed that SRAM-FPGAs represent a flexible and powerful resource for the creation of adaptive systems. However, their application in the context of aerospace creates the need for methodologies to detect (and cope) with both permanent and transient faults. In [5], they presents "a software flow for the generation of hard macros for [...] the diagnosing of permanent faults due to radiation".

Analogous remarks regarding the use of SRAM-FPGAs in space applications were made by Jacobs *et al.* [6]. In [6], we can find "a reconfigurable fault tolerance (RFT) framework that enables system designers to dynamically adjust a system's level of redundancy and fault mitigation based on the varying radiation incurred at different orbital positions". What is most relevant to our work, however, is the introduction of an "upset rate modeling tool" used to capture time-varying radiation effects in a given orbit.

The methodology we propose here is motivated by all of these works. Our transient fault model is more naive than the one described in [6], even if our approach could be easily extended to use it; on the other hand, unlike [6], our system is also capable of dealing with permanent faults. With respect to [5], we observe that our approach, not only detects both permanent and transient faults, but it does that while maximizing lifetime and availability.

III. Theoretical Background

In the context of dependable computing, a *fault* is defined as the hypothesized cause of an *error*, which is itself defined as the deviation from the correct and desirable behavior of a service [7]. With regard to persistence, faults can be classified in two categories:

- *transient faults*, whose negative effects on the system are assumed to be limited in time;
- *permanent faults*, which are assumed to last forever since the moment they appear, and may lead to the eventual *halt failure* of the system and the impossibility of providing a service if redundant resources or reconfiguration capabilities are not available.

In space, computing systems are exposed to high levels of radiation that pose a serious hazard to their proper functioning and survival. Ionizing and particle radiation, in fact, can be held responsible for a variety of undesirable outcomes. In particular, we distinguish:

- *single event upsets* (SEUs), these are transient faults that cause changes in the content of individual memory elements, i.e. *bit flips*;

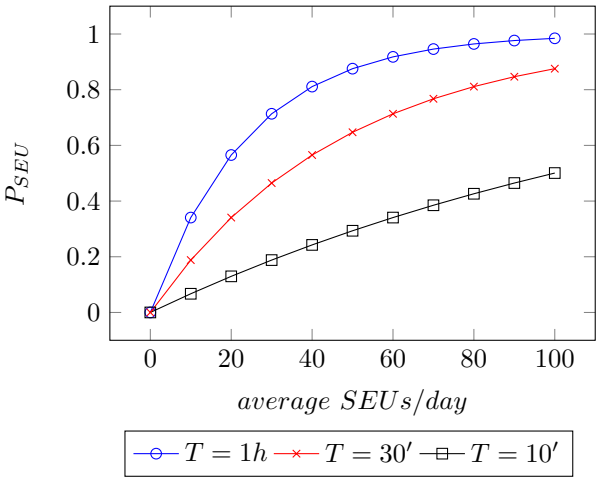

Fig. 1. Probability of observing at least one SEU in scrubbing periods of different durations, as the average ratio of SEUs per day increases.

- long-term damages, caused by the *total ionizing dose* (TID), that can have disruptive effects on current CMOS technologies and lead to performance degradation, permanent faults, and system failure [5].

Data scrubbing is an error detection and correction technique that consists in periodically re-reading and re-writing the content of a memory, using a "safe" copy known to be correct [8]. Exploiting data scrubbing, we are given the ability to detect faults, and, if they are not permanent, to correct them, at constant time intervals of duration T (scrubbing period).

A. Transient Faults Modeling

We employ probability theory to model the occurrence of transient (SEUs) and permanent (i.e. system failures given by TID) faults in space computing systems. The impacts of high-energy particles that cause SEUs are known to be independent and they usually happen at a constant average rate, given by the orbit or mission phase of the system.

We define the probability of observing at least one SEU in a scrubbing period of size T, given a constant average rate of SEU_r, as P_{SEU}. The probability of observing a given number of events that are known to occur at a constant average rate r is described by the Poisson distribution of parameter r, $Pois(r)$. Therefore, P_{SEU} does not depend on the actual time or history of the system and can be computed as:

$$\begin{aligned} P_{SEU} &= P(C_{SEU}(T) \geq 1 \mid C_{SEU}(T) \sim Pois(SEU_r \cdot T)) \\ &= 1 - P(C_{SEU}(T) = 0 \mid C_{SEU}(T) \sim Pois(SEU_r \cdot T)) \\ &= 1 - \text{pmf}_{Pois(SEU_r \cdot T)}(0) \\ &= 1 - e^{-SEU_r \cdot T} \end{aligned} \tag{1}$$

where $C_{SEU}(T)$ is defined as the number of SEUs observed in a period T, and pmf is the probability mass function. Figure 1 shows that this probability quickly increases with SEU_r, if T is large.

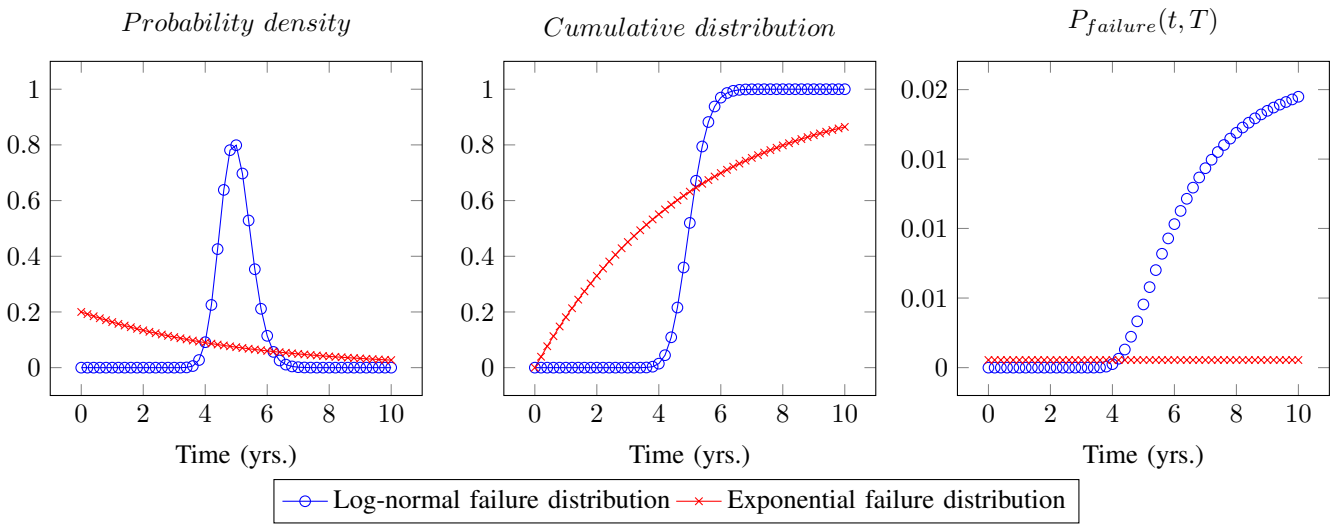

Fig. 2. Probability density function, cumulative distribution function, and probability of failure in the last scrubbing period (see Equation 2, $T = 1$ day), for log-normal and exponential failure distributions with $MTTF = 5$ years.

B. Permanent Faults Modeling

To model permanent faults we start by defining the probability of a permanent fault occurring in a component by time t, i.e $failure \leq t$, given that the component was still functional at the end of the previous scrubbing period $t - T$, i.e $failure > t - T$, as $P_{failure}(t, T)$.

This probability can be computed, using the Kolmogorov definition, as:

$$\begin{aligned} P_{failure}(t,T) &= P(failure \leq t \mid failure > t-T) \\ &= \frac{P(failure \leq t \wedge failure > t-T)}{P(failure > t-T)} \\ &= \frac{CDF_{failure}(t) - CDF_{failure}(t-T)}{1 - CDF_{failure}(t-T)} \end{aligned} \tag{2}$$

where $CDF_{failure}$ is the cumulative density function of the random variable $failure$ describing the time at which the failure happens.

In literature, several probability distributions are used to model failure times [9]. One of the most frequently used, because of its simplicity and the convenient *memorylessness* property, is the *exponential* distribution.

If we model the failure time using an exponential distribution with mean equal to expected mean time to failure (MTTF) of the system, Equation 2 becomes:

$$\begin{aligned} P_{failure}(t,T) &= \frac{1 - e^{-t/MTTF} - (1 - e^{-(t-T)/MTTF})}{1 - (1 - e^{-(t-T)/MTTF})} \\ &= 1 - \frac{1}{e^{T/MTTF}} \end{aligned} \tag{3}$$

Because the exponential distribution is memoryless, this value does not depend on current time t.

However, the exponential distribution representation is somewhat imprecise because it lacks the ability to capture the increasing failure probability due to accumulated wear in the component [9]. A common alternative used to overcome this limitation is a *log-normal* failure distribution:

$$P_{failure}(t,T) = \frac{\Phi\left(\frac{\ln t - \mu}{\sigma}\right) - \Phi\left(\frac{\ln(t-T) - \mu}{\sigma}\right)}{1 - \Phi\left(\frac{\ln(t-T) - \mu}{\sigma}\right)} \tag{4}$$

where Φ is the CDF of the normal distribution, and the log-normal parameters μ and σ can be computed from the MTTF and the variance of the failure time as follows:

$$\mu = \ln\left(\frac{MTTF^2}{\sqrt{var_{MTTF} + MTTF^2}}\right)$$

$$\sigma = \sqrt{\ln\left(1 + \frac{var_{MTTF}}{MTTF^2}\right)}$$

We can observe that in Equation 4, $P_{failure}(t, T)$ is no longer independent of the actual time t.

Figure 2 summarizes the comparison of exponentially and log-normally distributed failure times with the same MTTF of 5 years. The rightmost chart, in particular, shows how a memoryless and a non-memoryless distribution function differently describe the occurrence of a recent failure.

C. Hidden Markov Models

A hidden Markov model is a mathematical framework capable of describing the evolution over time of a stochastic system that can only be indirectly observed through stochastic sensors. An HMM can be considered as a special case of a Dynamic Bayesian Network (DBN) [10].

In an HMM, time is discrete and, at each time time step t, the state of the system is fully described by a discrete probability distribution function over a single random variable S_t. This variable cannot be observed, but its initial probability distribution $P(S_0)$ is known. At each time step, one can observe *evidences* O_t that solely depend on the current probability distribution over all the possible states $P(S_t)$. Moreover,

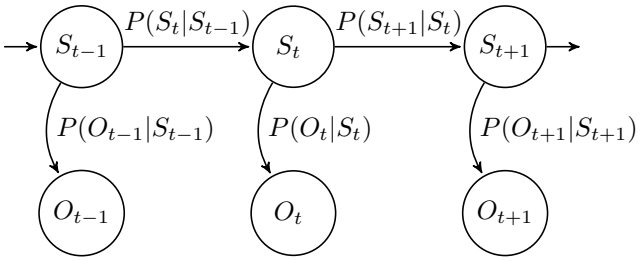

Fig. 3. Structure of a hidden Markov model.

the process is 1-step Markovian, meaning that $P(S_t)$ only depends on $P(S_{t-1})$. Figure 3 shows the structure of an HMM over 3 time steps.

In summary, HMMs can be formally described as:

- $P(S_0)$, an initial probability distribution over all the possible states;
- $T_{ij} = P(S_t = j \mid S_{t-1} = i)$, a transition model, governing the evolution of the system;
- $E_{ij} = P(O_t = j \mid S_t = i)$, a sensor model, that links the current state and possible observations.

1) State Prediction and Estimation using HMMs: HMMs can be used to predict the probability of the system being in a specific state at time t, given all the evidence up to time $t-1$, using the following equation (that can be recursively applied, having its base case in the initial probability distribution $P(S_0)$):

$$P_t(S = j \mid O_{1:t-1}) = \sum_i P_{t-1}(S = i \mid O_{1:t-1}) \cdot T_{ij} \quad (5)$$

Moreover, an HMM prediction can be refined after the observation of the current evidence by applying:

$$P_t(S = i \mid O_t = j,\ O_{1:t-1}) = \alpha\ P_t(S = i \mid O_{1:t-1}) \cdot E_{ij} \quad (6)$$

where α is a normalization parameter.

IV. Proposed Approach

In our approach, we use a multi-resource system model, and three methodologies to balance its lifetime and availability: a simple rule-based approach, a standard HMM and the proposed dynamic hidden Markov model formulation.

A. System Model

For our analysis, we introduce a simple computing system model, shown in Figure 4, with the following assumptions:

- the system is composed by N identical resources R_i's;
- each resource R_i is, alone, capable of providing the service required from the system;
- the system only uses one resource R_i at a time (to maximize system lifetime: in fact, resource wear depends on the resource being turned on and not on its level of utilization [11]);
- when a resource is active, it is subject to transient faults with constant average rate SEU_R;
- the failure time of each resource follows a log-normal probability distribution with mean equal to one N-th of the mean time to failure of the entire system ($MTTF$), starting from the time the resource is used for the first time. This means that the resource is not subject to wear as long as it is inactive;
- the active resource performs data scrubbing with a constant period T;
- whenever the scrubber detects the occurrence of and error, all the computation performed in the last period is discarded;
- whenever the scrubber detects the occurrence of and error, a *failure detection mechanism* is in charge of deciding whether to *rollback* the computation on the same resource R_i or to declare the resource dead (in permanent fault mode) and *migrate* the computation on the following resource R_{i+1};
- migration happens at a cost of a migration time $T_{migr.}$;
- the decision of declaring the death of a resource is irrevocable;
- after the last resource R_N is declared dead, the system is itself declared dead.

We can think of this model as an homogeneous multi-core system or any other N-times modular redundant computing system. It is clear from these premises that a *failure detection mechanism* that hastily declares the failure of resources will negatively impact the total lifetime of the system. However, a mechanism lingering for too long on obvious decisions might reduce the actual availability of the system.

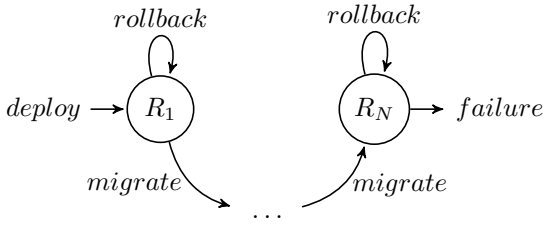

Fig. 4. Structure of an N-resource system.

B. Rule-based Failure Detection

To decide whether a resource should be considered in a state of permanent fault, we can employ different failure detection mechanisms. For this purpose, several previous research works exploit rule-based systems of different complexity [2, 6].

The simplest rule-based approach, implemented in this work as a reference (Algorithm 1), is based on the assumption that a resource failure will cause consecutive faults. If the number of consecutive faults observed by the mechanism is larger than a predefined threshold, the resource is assumed to have failed, and the computation is migrated to another resource, if available. The threshold can be as little as 2, meaning that two consecutive faults are enough to assume a permanent failure, or much larger. A threshold of 32, coupled with a scrub period

T of 1 hour means that we are willing to wait over a day before declaring the status of permanent fault.

Algorithm 1 Rule-based Algorithm

```
1:  consecutive_faults = 0;
2:  available = 1;
3:  while available do
4:      wait(scrub_time);
5:      fault = observe_fault();
6:      if fault then
7:          consecutive_faults++;
8:          if consecutive_faults ≥ threshold then
9:              available = 0;
10:         end if
11:     else
12:         consecutive_faults = 0;
13:     end if
14: end while
```

C. HMM-based Failure Detection

Algorithm 2 describes the behavior of an HMM-based failure detection mechanism. In this case, the mechanism retains a *belief state*, i.e. a probability distribution over three possible states of the resource: (1) available, (2) experiencing a SEU or (3) failed.

At each scrubbing period, this belief state is updated using the transition model given in Table II, and Equation 5. Then, after the result of the scrubbing operation, the belief state is filtered using the sensor model in Table I, and Equation 6. Finally, Equation 5 is used again to predict the future state of the resource.

TABLE I
HMM SENSOR MODEL

	$P(O_t)$	
S_t	Fault	No Fault
Available	0	1
SEU	1	0
Failure	1	0

TABLE II
HMM TRANSITION MODEL

	$P(S_t)$		
S_{t-1}	Available	SEU	Failure
Available	$1 - P(SEU \vee Failure)$	P_{SEU}	$1 - \frac{1}{e^{T/MTTF}}$
SEU	$1 - P(SEU \vee Failure)$	P_{SEU}	$1 - \frac{1}{e^{T/MTTF}}$
Failure	0	0	1

The decision of performing a migration is taken if the probability that the system will be in permanent failure in the near future is greater than a predefined level of confidence.

Tables II and I show that we are assuming the sensor model to be perfect (only 0s and 1s in Table I), the environment to be *static* (constant values in Table II), and that permanent faults cannot be recovered (only 0s and a 1 in the last line of Table II).

Algorithm 2 HMM-based Algorithm

```
1:  belief_state = init_belief();
2:  available = 1;
3:  while available do
4:      wait(scrub_time);
5:      fault = observe_fault();
6:      if fault then
7:          belief_state = update_belief(transition_model);
8:          belief_state = filter_belief(sensor_model);
9:          predict_state = predict_belief(transition_model);
10:         if predict_state.failure ≥ threshold then
11:             available = 0;
12:         end if
13:     else
14:         belief_state = update_belief(transition_model);
15:         belief_state = filter_belief(sensor_model);
16:     end if
17: end while
```

D. Dynamic HMM-based Failure Detection

A major limitation of the HMM-based mechanism is that the transition model is constant over time. This means that the model could easily capture the behavior of a resource with an exponentially distributed failure time, but it would fail to represent the log-normal distribution that we assumed in Subsection IV-A.

HMMs are meant to describe Markovian processes, i.e., processes that propagate one time step at a time. However, nothing prevents us from changing their static nature and make the transition model *dynamic*. We implemented the new transition model shown in Table III: the first two elements of the last column, representing the probability of encountering a permanent failure in a single time step, are no longer constants, but rather a function of the global time variable. This allows us to exploit Equation 4 and properly model the log-normal failure time distribution.

It is worth noting that the algorithm implementing the modified dynamic HMM-based mechanism differs from Algorithm 2 only by an additional step used to re-evaluate the transition model. The sensor model is still assumed to be perfect, as presented in Table I. As for the HMM-based mechanism, the decision of performing a migration is taken if the evidence shows that the system will be in permanent failure in the near future with a certain level of confidence.

TABLE III
DYNAMIC HMM TRANSITION MODEL

	$P(S_t)$		
S_{t-1}	Available	SEU	Failure
Available	$1 - P(SEU \vee Failure)$	P_{SEU}	$P_{failure}(t)$
SEU	$1 - P(SEU \vee Failure)$	P_{SEU}	$P_{failure}(t)$
Failure	0	0	1

V. Experimental Setup

In order to perform the simulation and validation of our methodology we implemented the algorithms and the system model described in Section IV using the MATLAB-compatible, GNU Octave programming language. Simulations where carried out on a quad-core Intel i7 desktop running at 3.2GHz with 32GB of RAM.

A. Parameters

The parameters that define the environment are those outside the control of the designer of the failure detection mechanism. These parameters regulate the occurrence of faults and the maximum lifetime of the system:

- SEU_r - in our simulations, the average rate of single event upsets *per day* can take the value of 16.5 or 62. These are the number of daily SEUs expected in a Virtex-4 FX60 in a low-Earth orbit (LEO) and in a highly-elliptical orbit (HEO), respectively [6];
- $MTTF$ - we choose components so that the mean time to failure of the system is fixed to 5 years, each resource of the system has a MTTF equal to one N-th of 5, depending on the number of resources in the system;
- var_{MTTF} - the variance of a resource lifetime is arbitrarily fixed to the 10% of the MTTF;
- $T_{migr.}$ - the migration time is assumed constant and equal to 10 minutes.

Furthermore, we perform our simulations screening multiple values for the parameters that can be chosen by the designer of the failure detection mechanism. These parameters affect the actual lifetime of the system and its availability:

- N - the number of resources varies from 3 to 10;
- T - we consider 3 different scrub periods of 10, 30, and 60 minutes each;
- finally, we consider 5 different thresholds for each failure detection mechanism: 2, 4, 8, 16, 32 for the rule-based method, and 0.5, 0.75, 0.88, 0.94, 0.97 for HMM and D-HMM.

B. Performance Metrics

As mentioned in Subsection IV-A, a good failure detection mechanism does not shorten the total system lifetime while maintaining an optimal system availability. Therefore, we consider lifetime and availability as the two metrics needed to assess the performance of our mechanisms.

The system lifetime L is equal to the time t when the mechanism declares the permanent failure of the last (N-th) resource in the system. This can be compared to the expected lifetime obtained by a *clairvoyant* mechanism given by equation:

$$\mathbb{E}_{cv}[L] = N \cdot MTTF + (N-1) \cdot T_{migr.} \tag{7}$$

An optimistic mechanism would record longer lifetimes, postponing the declaration of failure of computing resources as much as possible. This would happen at the price of a lower system availability.

The total availability can be defined as the fraction of scrubbing periods in which no faults are detected:

$$A = 1 - \frac{|detected\ faults| \cdot T}{L} \tag{8}$$

However, this metric is biased towards small scrubbing periods T, and does not allow for a fair comparison of results obtained with varying SEU rates. Therefore, in this work, we use a normalized measure of availability defined as:

$$\widetilde{A} = \frac{A}{\mathbb{E}_{cv}[A]} \tag{9}$$

where the expected availability $\mathbb{E}_{cv}[A]$ is computed as:

$$\mathbb{E}_{cv}[A] = 1 - \frac{SEU_r \cdot N \cdot MTTF + (N-1) \cdot T_{migr.}}{N \cdot MTTF + (N-1) \cdot T_{migr.}} \tag{10}$$

VI. Discussion

Figures 5 and 6 show the results obtained with $N = 10$ resources and a scrubbing period $T = 1$ hour, for a system operating in Low Earth Orbit (LEO, 16.5 SEUs per day) and a Highly Elliptical Orbit (HEO, 62 SEUs per day), respectively.

One can notice that the rule-based approach fails to achieve the desired lifetime of 5 years even at LEO, unless a very high threshold on the number of acceptable consecutive faults is used. Moreover, small thresholds result in short and highly variable lifetimes for both the rule-based and the HMM system. Regardless of the mechanism used, one can observe the presence of a compromise between availability and lifetime.

We also remark that the D-HMM approach performs better than HMM, especially in terms of lifetime. This can be explained by the fact that the D-HMM truly captures the accumulated wear of electronic devices. The very small deviations from the ideal normalized availability of 1.0 are justified by the observation that any failure detection mechanism can make an impact on this metric only when a failure actually happens, and its significance is limited by the number of resources N. We envision that much larger gains will be in reach in the many-cores era, when system will have tens or hundreds of resources. Figure 7 presents the lifetime-availability trade-offs obtained by the three failure detection mechanisms in the form of Pareto curves for LEO and HEO. The number of resources is again $N = 10$, and data scrubbing is performed with a period $T = 10$ minutes. To map the *ideal* situation in the origin of the axes $(0, 0)$ instead of $(+\inf, +\inf)$, we plot using the reciprocals of the availability and lifetime metrics. Each point represents the performance of a mechanism (with their associated threshold). In the LEO scenario the points on curve produced by the the proposed D-HMM dominate all other solutions (HMM and rule-based).

Finally, Table IV summarizes the lifetimes and normalized availabilities (see Equation 9) for the rule-based, HMM-based, and D-HMM-based approaches considering all five possible thresholds, and with number of resources ranging from 3 to 10. Data are reported for both the LEO and the HEO, with a scrubbing period $T = 1$ hour. Results show that D-HMM consistently outperforms HMM and rule-based approaches in

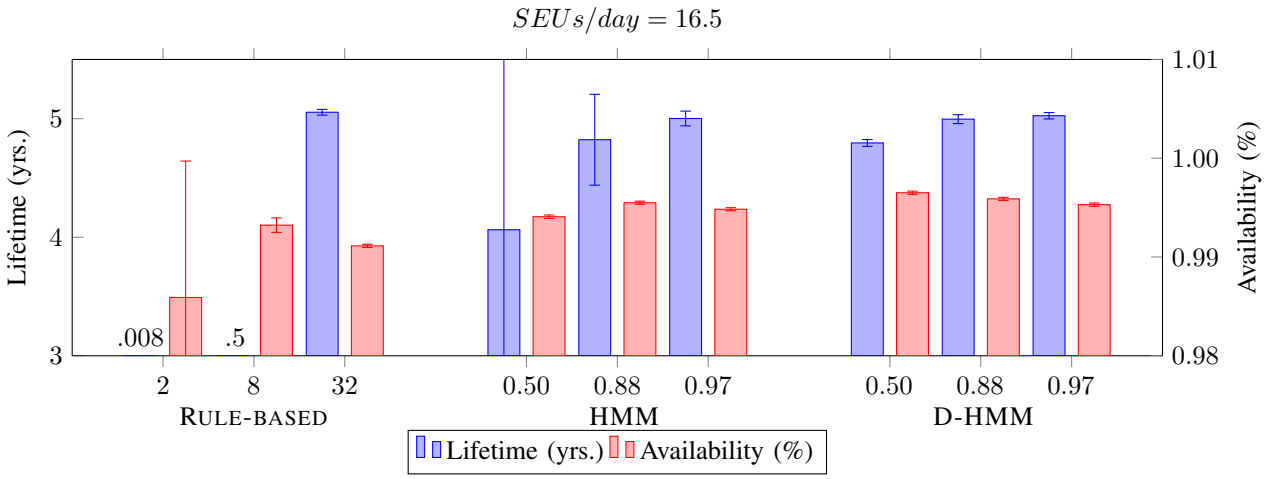

Fig. 5. Comparison of system lifetimes and normalized availabilities (see Equation 9) for the rule-based, HMM-based, and dynamic HMM-based approaches with different thresholds, assuming $N = 10$ resources and a scrubbing period $T = 1h$, when $SEUs/day = 16.5$.

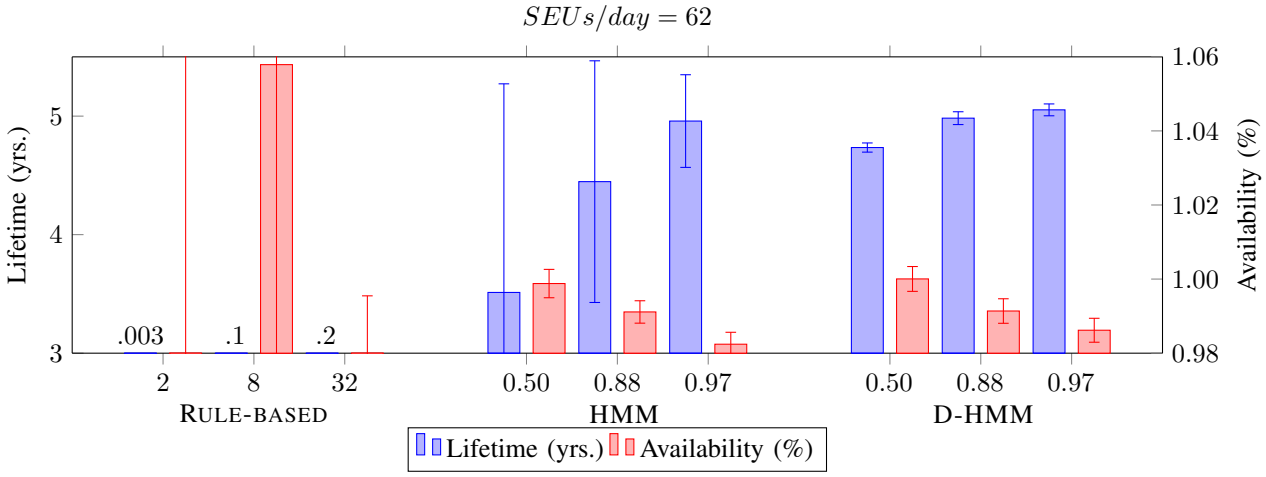

Fig. 6. Comparison of system lifetimes and normalized availabilities (see Equation 9) for the rule-based, HMM-based, and dynamic HMM-based approaches with different thresholds, assuming $N = 10$ resources and a scrubbing period $T = 1h$, when $SEUs/day = 62$.

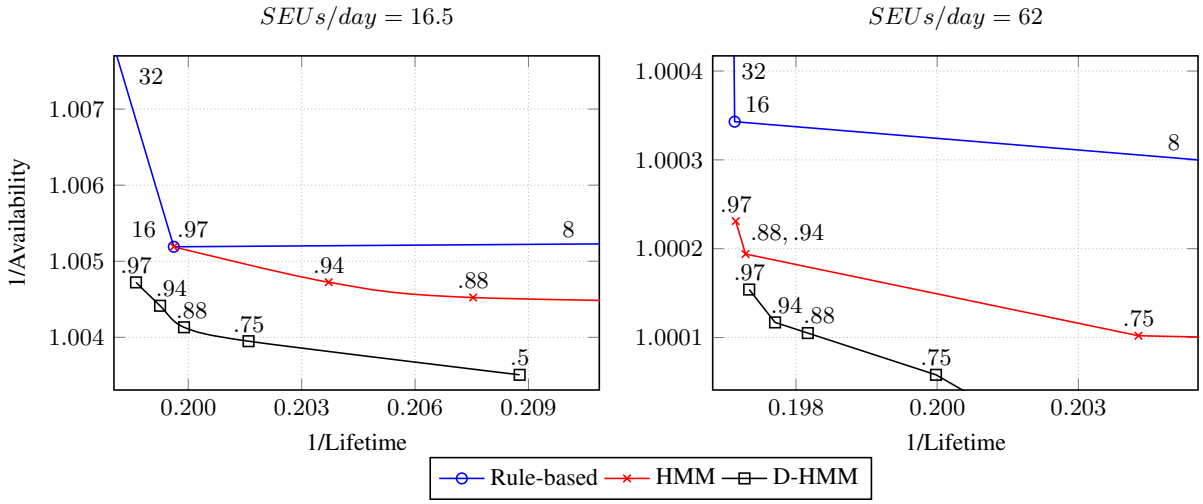

Fig. 7. Pareto curves obtained by the rule-based, HMM-based, and dynamic HMM-based approaches with different thresholds, assuming $N = 10$ resources and a scrubbing period $T = 10'$, for two levels of $SEUs/day$.

TABLE IV
LIFETIMES AND NORMALIZED AVAILABILITIES (SEE EQUATION 9) FOR THE RULE-BASED, HMM-BASED, AND DYNAMIC HMM-BASED APPROACHES WITH DIFFERENT THRESHOLDS, NUMBER OF RESOURCES IN THE SYSTEM, AND LEVELS OF $SEUs/day$, WHEN THE SCRUBBING PERIOD $T = 1h$

		$N=3$				$N=5$				$N=10$			
		$SEUs/day = 16.5$		$SEUs/day = 62$		$SEUs/day = 16.5$		$SEUs/day = 62$		$SEUs/day = 16.5$		$SEUs/day = 62$	
	thr	L (yrs)	$\widetilde{A}$ (%)	L (yrs)	$\widetilde{A}$ (%)	L (yrs)	$\widetilde{A}$ (%)	L (yrs)	$\widetilde{A}$ (%)	L (yrs)	$\widetilde{A}$ (%)	L (yrs)	$\widetilde{A}$ (%)
Rule-based	2	0.00	0.943	0.00	0.589	0.00	0.903	0.00	0.823	0.01	0.986	0.00	0.925
	4	0.01	0.977	0.00	0.806	0.02	0.957	0.00	0.857	0.04	1.004	0.01	0.942
	8	0.16	1.000	0.00	0.907	0.31	0.995	0.01	0.796	0.53	0.993	0.01	1.058
	16	4.74	0.997	0.01	0.916	4.79	0.998	0.02	0.876	5.00	0.995	0.04	0.983
	32	5.01	0.996	0.06	0.987	5.01	0.996	0.09	0.963	5.05	0.991	0.17	0.974
HMM	0.50	3.47	0.998	3.23	0.991	4.19	0.997	3.43	1.001	4.06	0.994	3.51	0.999
	0.75	4.49	0.997	4.47	0.992	4.55	0.998	4.50	0.997	4.55	0.996	4.35	0.995
	0.88	4.74	0.997	4.67	0.991	4.71	0.998	4.91	0.994	4.82	0.995	4.45	0.991
	0.94	4.87	0.997	4.87	0.990	4.79	0.998	5.04	0.993	4.91	0.995	4.77	0.987
	0.97	4.87	0.997	4.88	0.989	4.92	0.998	5.05	0.992	5.00	0.995	4.96	0.982
D-HMM	0.50	4.75	0.998	4.73	0.996	4.80	0.998	4.77	1.002	4.80	0.997	4.73	1.000
	0.75	4.89	0.998	4.92	0.995	4.92	0.998	4.89	0.999	4.96	0.996	4.91	0.995
	0.88	4.93	0.998	4.96	0.994	4.96	0.998	4.95	0.998	5.00	0.996	4.98	0.991
	0.94	4.97	0.998	4.99	0.993	4.98	0.998	5.00	0.996	5.01	0.996	5.03	0.988
	0.97	4.99	0.998	5.02	0.993	4.99	0.998	5.04	0.995	5.03	0.995	5.05	0.986

terms of availability, lifetime, and stability of the results (i.e. lower variance).

VII. CONCLUSIONS

In this paper we proposed a failure detection mechanism for electronic components used in space applications that are exposed to both ionizing and particle radiation and, therefore, may experience unpredictable transient faults (SEUs) and permanent faults due to wear out. Our methodology is based on hidden Markov models, however, we extend the classical framework through the use of a dynamic transition model in order to cope with failure time probability distributions that are not memoryless.

We defined lifetime and a normalized availability measure as the two metrics that should be maximized by a properly functioning failure detection mechanism, and we showed that a trade-off between the two is inevitable. We simulated our approach using a simple yet powerful model, performing error injection with realistic parameters, in order to assess its validity. Moreover, we compared the performance of our approach against the performance of a rule-based approach, showing improvements in both the goal metrics and reductions in their variance,and a non-dynamic hidden Markov model approach, showing how the framework benefits from our enhancements.

REFERENCES

[1] E. Petersen, *Single event effects in aerospace*, 2011.

[2] S. Gupta, A. Ansari, S. Feng, and S. Mahlke, "Adaptive online testing for efficient hard fault detection," in *2009 IEEE Int. Conf. Comput. Des.* IEEE, Oct. 2009, pp. 343–349.

[3] J. C. Smolens, B. T. Gold, J. C. Hoe, B. Falsafi, and K. Mai, "Detecting emerging wearout faults," in *In Proceedings of the IEEE Workshop on Silicon Errors in Logic - System Effects*, 2007.

[4] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in cmos processes," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 128–143, April 2004.

[5] L. Cassano, D. Cozzi, S. Korf, J. Hagemeyer, M. Porrmann, and L. Sterpone, "On-line testing of permanent radiation effects in reconfigurable systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 717–720.

[6] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam, "Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive fpga-based space computing," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 4, pp. 21:1–21:30, Dec. 2012.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, Jan 2004.

[8] M. Garvie and A. Thompson, "Scrubbing away transients and jiggling around the permanent: long survival of fpga systems through evolutionary self-repair," in *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International*, July 2004, pp. 155–160.

[9] E. A. Elsayed, *Reliability Engineering*, 2nd ed. Wiley Publishing, 2012.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[11] J. Henkel, T. Ebi, H. Amrouch, and H. Khdr, "Thermal management for dependable on-chip systems," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, Jan 2013, pp. 113–118.