Efficient Transient Simulation of Failure/Repair Markovian Models

Juan A. Carrasco
Departament d'Enginyería Electrònica
Universitat Politécnica de Catalunya
Diagonal 647, plta. 9, 08028 Barcelona, Spain

Abstract

Simulation methods have recently been developed for the solution of the extremely large markovian dependability models which result from complex fault-tolerant computer systems. This paper presents efficient simulation methods for the estimation of transient reliability/availability metrics for repairable fault-tolerant computer systems which combine estimator decomposition techniques with an efficient importance sampling technique recently developed. Comparison with simulation methods previously proposed for the same type of metrics and models shows that the methods proposed here are orders of magnitude faster.

1 Introduction

This paper considers the evaluation of repairable fault-tolerant computer systems which from the user's point of view can be seen as either up or down. Several dependability metrics such as the steady-state availability, the availability, the expected interval availability, the mean time to failure and the reliability can be appropriate to evaluate the up/down behavior of these systems. For the computation of these metrics, the system can be conceptualized as made up of components which change their state as a result of failure and repair processes. These processes may involve in general several components and the system is up or down depending on the unfailed/failed state of the components. Examples of this conceptual view are provided by the modeling languages described in [1, 2]. Under the assumption of exponential failure and repair time distributions, the behavior of the system can be described by a continuous-time Markov chain (CTMC) X, whose transitions represent either failures or repairs of components. The state space of X can be partitioned into the set of up states U and the set of down states D. Let $u \in U$ be the state in which all components are unfailed. We assume that all components are repairable and that at least one failed component is under repair in any state $x \neq u$. Failure propagation, the impact of the operational configuration of the system into failure and repair processes, and limited repairmen introduce dependencies in the model such that, in general, X has to be solved using general-purpose, state-level methods. Current state-of-the-art numerical methods [3, 4] can solve CTMC's with tens of thousands of states. However, in general, the size of X grows exponentially with the number of components of the system and both its generation and numerical solution are precluded when the system has more than a few components.

Simulation is an approach which by nature is not limited by the size of the model. However, repair rates are typically several orders of magnitude higher than failure rates, which makes the probability of X following a failure transition from any state $x \neq u$ very small. Then, if as it is typical, the system has redundancy to tolerate most single faults, the structure of X is such that only very rarely will X enter D. This makes direct Monte Carlo simulation unfeasible in practice due to the extremely large number of events which would have to be sampled to get estimates with reasonable confidence intervals.

Two types of techniques have been proposed to speed up direct Monte Carlo simulation. Importance sampling techniques exploit heuristic knowledge about the model to modify the sampling distributions so that the rare contributing paths be sampled with significant probabilities. Failure biasing and forced transition are two such techniques which were initially proposed in the context of the nuclear domain [5, 6], and have been recently further developed and applied with success to the simulation of models of fault-tolerant computer systems [7–11]. Another importance sampling technique called failure distance biasing which is typically much more efficient than failure biasing has been recently proposed [12]. Estimator decomposition techniques exploit the regenerative behavior of X around u to formulate the metric of interest in terms of lower level metrics which can be estimated more efficiently. Such techniques have been recently used for the estimation of the steady-state availability [8, 10] and the mean time to failure [9, 10]. However, currently proposed methods [5, 10] for transient metrics such as the availability, the expected interval availability and the reliability do not use such techniques and, as it will be discussed, can be highly inefficient.

This paper presents new simulation methods for the availability, the expected interval availability, and the reliability combining estimator decomposition techniques with the importance sampling technique described in [12]. The methods are robust and orders of magnitude faster than methods previously proposed

CH3021-3/91/0000/0152/$01.00 © 1991 IEEE

for the same metrics. The rest of the paper is organized as follows. Section 2 reviews importance sampling theory and previous simulation methods. Section 3 argues qualitatively the inefficiency of previous methods and presents the estimator decomposition techniques on which the new simulation methods are based. Section 4 examines the problem of deriving confidence intervals for the estimates. Section 5 presents experimental results illustrating the robustness and efficiency of the proposed methods. Section 6 concludes the paper. The formulations on which the estimator decomposition techniques are based are derived in the Appendix.

2 Background

Since all metrics considered in this paper have usually values very close to 1, we use the complementary metrics (unavailability, expected interval unavailability, and unreliability) in our presentation. For all metrics we assume that X is initially in the state u. The unavailability at time t, $ua(t)$ is the probability that X is in D at time t. The expected interval unavailability at time t, $iua(t)$ is defined as $mdt(t)/t$, where $mdt(t)$ is the mean down time by time t, i.e., the expected value of the time spent by X in D during the interval $[0,t]$. It can be easily shown that:

$$mdt(t) = \int_0^t ua(\tau)d\tau \,. \qquad (1)$$

Finally, the unreliability at time t, $ur(t)$, can be defined as the probability that X has hit D during the interval $[0,t]$.

All the metrics previously defined can be expressed as the expected value of a function defined over the set of paths of X and, therefore, can be estimated by simulating X and taking the sample mean of the corresponding path function. However, only paths which hit D have non-null contributions and, given the structure of X, an extremely large number of events (transitions) would have to be sampled to generate the number of contributing paths required to achieve reasonable confidence intervals. The simulation can be speeded up by using importance sampling techniques.

The principles behind importance sampling [13] can be illustrated using a simple example. Assume we want to estimate the expected value of $Z(\Theta)$, where Θ is a random variable with density $f(\theta)$. Let $f^*(\theta)$ be a different density function such that $f^*(\theta) \neq 0$ whenever $Z(\theta)f(\theta) \neq 0$ and, for the values θ satisfying the latter condition, let $\Lambda^*(\theta) = f(\theta)/f^*(\theta)$ be the likelihood ratio. We have:

$$E[Z(\Theta)] = \int Z(\theta)f(\theta)d\theta = \int Z(\theta)\Lambda^*(\theta)f^*(\theta)d\theta \,,$$

which tell us that un unbiased estimator of $E[Z(\Theta)]$ can be obtained by sampling Θ using $f^*(\theta)$ and taking the sample mean of the function $Z^*(\theta) = Z(\theta)\Lambda^*(\theta)$. The variance after n samples is:

$$\sigma_Y^2 = \frac{1}{n}\left(\int Z^2(\theta)\frac{f^2(\theta)}{f^{*2}(\theta)}f^*(\theta)d\theta - E[Z(\Theta)]^2\right) ,$$

which is null when $f^*(\theta) = f(\theta)Z(\theta)/E[Z(\Theta)]$, i.e., when the values of Θ are sampled with likelihoods equal to their relative infinitesimal contributions to $E[Z(\Theta)]$. This is impossible in practice because $E[Z(\Theta)]$ is unkown, but when choosing a practical $f^*(\theta)$ we should try to match as closely as possible $f(\theta)Z(\theta)/E[Z(\Theta)]$. In the context of this paper, this means that we should drive the simulation effort to the contributing paths (those hitting D) and, furthermore, assign higher likelihoods to the paths with higher contributions to the metric under estimation. We next review briefly the simulation methods proposed in [5, 10] for $ur(t)$ and $iua(t)$.

The method proposed in [5] for the simulation of $ur(t)$ samples paths of X till either the path hits D or the sum of all sampled holding times exceeds t. Let λ_x denote the output rate from the state x and t_c the sum of the previously sampled holding times. The holding times in states $x \neq u$ are sampled from their exponential distribution $f(y) = \lambda_x e^{-\lambda_x y}$, but the holding times in u are sampled using the conditional distribution $f^*(y) = \lambda_u e^{-\lambda_u y}/(1 - e^{-\lambda_u(t-t_c)})$, which results when the holding time in u is restricted to be $< t - t_c$. This importance sampling technique is called *transition forcing* and is used to avoid sampling paths which will not hit D by time t and thus will have null contributions to $ur(t)$. Transitions are sampled using another importance sampling technique called *failure biasing*. Failure biasing is applied when the current state has both outgoing failure and repair transitions and consists in biasing the transition probabilities so that the probability of following a failure transition be $FBIAS$ and that of following a repair transition be $1 - FBIAS$, with the biasing probabilities within each class distributed proportionally to the unbiased probabilities. The biasing scheme is turned off as soon as a down state is hit. The heuristic behind failure biasing is to increase the sampling probabilities of the paths which enter D. A value $FBIAS = 0.5$ typically minimizes the required simulation effort.

The simulation method for $ur(t)$ proposed in [10] is sligthly different. The method simulates X till entry in D and does not sample the holding times in u. The number of visits k to u during a path is recorded and the contribution of the path to $ur(t)$ is computed by conditioning out the distribution of the total time spent in u, which follows a k-stage Erlang distribution with parameter λ_u. The method for $iua(t)$ proposed in [10] conditions out a predefined value L of holding times in u and samples successive holding times in u till the sum of all sampled holding times (including those in states $x \neq u$) exceeds t. Two importance sampling techniques in addition to failure biasing have been experimented in [10]. The first one is identical to failure biasing except that $FBIAS$ is distributed uniformly among the failure transitions. The second is an ellaboration of failure biasing in which a given probability is allocated to the transitions corresponding to failures of component types with already some component failed. The latter is only meaningful when redundancy is only provided between components of the same type (i.e., undistinguishable), quite a restrictive assumption which is not fullfilled, for instance, by

the large example presented in Section 5.

Failure distance biasing is a more ellaborated biasing scheme which exploits the concept of *failure distance*. See [12] for a more detailed description of the scheme, including implementation and optimization issues. In failure distance biasing components are classified into *failing* and *non-failing*, depending on whether they fail on their own or can only be failed by propagation of failures of other components. The *failure distance* from an operational state x is defined as the minimum number of failing components whose failure in x would take the system down. A failure transition is said *dominant* if it reduces the failure distance and *critical* if it reduces the failure distance by more than one. The *criticality* of a failure transition is defined as the amount by which it reduces the failure distance. Failure distance biasing is done in steps. At each step a subset of transitions is split into two subsets, which are biased in relation to one another (keeping as in failure biasing the relative values of the transition probabilities within each subset), and one of the subsets is passed to the next step. If one of the subsets is empty, the step is skipped. The first step is failure biasing. In the next step, the subset of dominant failure transitions is assigned a probability *DBIAS* in relation to the current set (failure transitions) and passed to the next step, and the subset of non-dominant transitions is assigned a relative probability $1 - DBIAS$. The third step is repeated while the transitions in the current set have different criticalities and assigns the relative probability $1 - CBIAS$ to the subset of failure transitions with smallest criticality and *CBIAS* to the complementary subset, which is passed for the next application of the biasing step. Assume, for instance, that a state has outgoing repair transitions, non-critical dominant transitions, and critical failure transitions of criticalitites 2 and 3. These subsets of transitions would be sampled with probabilities $1 - FBIAS$, $FBIAS(1 - CBIAS)$, $FBIAS\,CBIAS(1 - CBIAS)$, and $FBIAS\,CBIAS^2$, respectively. The heuristic behind failure distance biasing is to sample more often the shortest paths entering D. By taking values of *DBIAS* and *CBIAS* more or less close to 1, the biasing scheme can focus more or less the simulation effort to these paths. Using an independent biasing parameter to deal with critical transitions is convenient since the actual importance of these transitions depends on the values of the "coverage" parameters of the model. The scheme has been compared in [12] with failure biasing for the simulation of the steady-state unavailability and has been shown to be typically much more efficient.

3 Estimator Decomposition

Failure distance biasing and, to a lesser extent, failure biasing and the related biasing schemes described in [10] tend to concentrate the simulation effort towards the shortest paths entering D. These paths have the more important contributions to $ua(t)$, $iua(t)$ and $ur(t)$ only when t is smaller or of the order of magnitude of the mean holding time in u, h_u. We present next an approximate analysis for $ur(t)$ which clarifies this point and provides further insight.

Let ν_u^D be the number of sojourns in u before X hits D and T_{uD} the time at which X hits D. We can formulate $ur(t)$ as:

$$ur(t) = \sum_{k=1}^{\infty} Q_k(t)\,,$$

where

$$Q_k(t) = P[T_{uD} \leq t \mid \nu_u^D = k]P[\nu_u^D = k]\,.$$

$Q_k(t)$ can be interpreted as the contribution to $ur(t)$ of the set of paths hitting D after k sojourns in u. Let γ be the probability that a path starting in u hits D before u. From the memoryless property of CTMC's, it follows that ν_u^D has a geometric distribution with parameter γ. Since repair rates are typically several orders of magnitude higher than failure rates and all states $x \neq u$ have at least one outgoing repair transition, the mean holding times in the states $x \neq u$ are usually much smaller than the mean holding time in u, h_u. In addition, given the structure of X, paths which visit many states $x \neq u$ before hitting D after a given number k of sojourns in u have much smaller probabilities than paths of the same type visiting few states $x \neq u$. This allows to neglect the holding times in the states $x \neq u$ and approximate T_{uD} conditioned to $\nu_u^D = k$ by the sum of k holding times in u. It follows that T_{uD} conditioned to $\nu_u^D = k$ has approximately a k-stage Erlang distribution with parameter $\lambda_u = h_u^{-1}$, and:

$$Q_k(t) \approx (1 - \sum_{r=0}^{k-1} \frac{(t/h_u)^r}{r!} e^{-t/h_u})(1-\gamma)^{k-1}\gamma\,.$$

The expected value of ν_u^D when the paths are sampled with probabilities proportional to their relative contributions (optimal distribution to which an efficient importance sampling scheme would tend to adjust) is $\overline{\nu_u^D} = \sum_{k=1}^{\infty} kQ_k(t) / \sum_{k=1}^{\infty} Q_k(t)$. Figure 1 plots $\overline{\nu_u^D}$ as a function of γ and t/h_u, using the approximation developped for $Q_k(t)$. It is clear that when t is smaller or of the order of magnitude of h_u only paths with few regenerative cycles around u have important contributions to $ur(t)$. However for small values of γ (which are typical in the models considered here), longer and longer paths have important contributions to $ur(t)$ as t/h_u becomes $\gg 1$.

The same can be argued to happen when simulating $ua(t)$ and $iua(t)$. The conclusion is that simulation methods for transient metrics using biasing schemes which tend to focuss the simulation effort towards short paths entering D can be expected to degradate when $t \gg h_u$ due to: a) an increase in the simulation effort per path, b) a decrease in the efficiency of the biasing scheme.

We present next formulations for $ua(t)$, $iua(t)$, and $ur(t)$ in terms of lower-level metrics which follow a

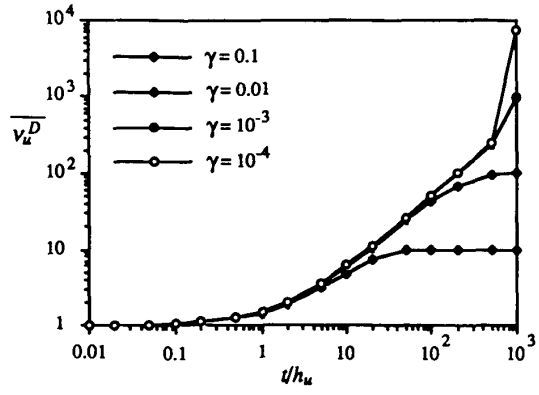

Figure 1: Approximate average number of sojourns in u per path during the simulation of $ur(t)$ when paths are sampled according to their contributions.

neat heuristic and can be simulated more efficiently. Let

- $g_1(t)$ the probability that given $X(0) = u$, $X(t) \in D$ and X has not hit u by time t,
- $g_2(t)$ the probability density function of the duration of a regenerative cycle around u,
- $h_1(t)$ the probability that given $X(0) = u$, X has hit D by time t without hitting u before,
- $h_2(t)$ the probability that given $X(0) = u$, X has hit either D or u by time t.

In the Appendix we formulate the Laplace transforms of $ua(t)$ and $ur(t)$ in terms of the Laplace transforms of the above low-level metrics. Denoting by $F(s)$ the Laplace tranform of $f(t)$, we obtain:

$$UA(s) = \frac{G_1(s)}{1 - G_2(s)} \tag{2}$$

$$UR(s) = \frac{H_1(s)}{1 - s(H_2(s) - H_1(s))} . \tag{3}$$

Remembering that $iua(t) = mdt(t)/t$, it becomes clear that the simulation of $iua(t)$ is virtually equivalent to the simulation of $mdt(t)$. Let $I_1(s) = G_1(s)/s$, using (1, 2), we have:

$$MDT(s) = \frac{UA(s)}{s} = \frac{I_1(s)}{1 - G_2(s)} . \tag{4}$$

Efficient numerical algorithms for numerical Laplace transform inversion exist (see, for instance, [14]) which allow to compute the value of a function $f(t)$ at a given abscissa t from a few values of its Laplace transform $F(s)$. Then, estimates for $ua(t)$, $ur(t)$, and $mdt(t)$ can be obtained from estimates for the Laplace transforms in the right-hand sides of (2, 3, 4) at the abscissae required by the inversion algorithm.

Each low-level metric transform $M(s)$ can be formulated as the expected value of a function $Z_r(s)$ defined over the set of paths to absorption of one of two transient discrete-time Markov chains (DTMC's), i.e., we can write:

$$M(s) = \sum_{r \in R} Z_r(s) P(r) ,$$

where R is the set of paths to absorption of the corresponding transient DTMC and $P(r)$ is the probability of the path r. Two transient DTMC's with initial state u are used. The first one, X', describes (ignoring holding times) the behavior of X till its first hit in u. The second one, X'', describes the behavior of X till its first hit in either u or D and has two absorbing states: a, representing hit in u, and d, representing hit in D. Let $L(r)$ denote the length of path r (number of transitions), $x_i(r)$ the state visited in path r after the ith transition ($x_0(r) = u$), λ_{ij} the transition rate from i to j, and $q_{ij} = \lambda_{ij}/\lambda_i$ the conditional transition probability from i to j. We have $P(r) = \prod_{0 \le i < L(r)} q_{x_i(r), x_{i+1}(r)}$. The path function required for each low level metric can be derived by considering that the holding times in the states $x_i(r), i = 0, \ldots, L(r) - 1$ conditioned to the execution of r are independent exponential random variables with parameters $\lambda_{x_i(r)}$. Table 1 gives the transient DTMC and the path function required for each low-level metric. In the table, $P_{x_i(r)/r}(s)$ denotes the Laplace transform of the probability that the associated transient CTMC is in $x_i(r)$ at time t, conditioned to the execution of r by the embedded DTMC. These Laplace transforms can be computed using:

$$P_{x_0(r)/r}(s) = \frac{1}{s + \lambda_{x_0(r)}}$$

$$P_{x_i(r)/r}(s) = \frac{\lambda_{x_{i-1}(r)}}{s + \lambda_{x_i(r)}} P_{x_{i-1}(r)/r}(s) \qquad i = 1, \ldots, L(r) - 1 ,$$

so that the computation of any path function of Table 1 has linear cost on the path length.

In the methods proposed here, an independent simulation stream is used to estimate the Laplace tranforms of each low-level metric. Given the structure of X, only the regenerative cycles around u with few transitions have significant probabilities and the duration of these cycles follows very closely an exponential distribution with mean h_u. Thus, all paths r of X' with significant contributions to $G_2(s)$ have a similar functional value $Z_r(s)$ and, according to importance sampling theory, direct Monte Carlo simulation can be expected to be very efficient. A similar reasoning based on the typical behavior of X supports the efficiency of direct Monte Carlo simulation of $H_2(s)$. Importance sampling techniques are however required for the efficient estimation of, $G_1(s)$, $H_1(s)$, and $I_1(s)$, since only the paths of X' and X'' hitting D have nonnull contributions to these metrics. In addition, the

Table 1: Transient DTMC and path function required for each low-level metric.

metric	DTMC	path function
$G_1(s)$	X'	$\sum_{0\le i<L(r),x_i(r)\in D} P_{x_i(r)/r}(s)$
$G_2(s)$	X'	$\prod_{0\le i<L(r)} \frac{\lambda_{x_i(r)}}{s+\lambda_{x_i(r)}}$
$H_1(s)$	X''	0 if $x_{L(r)}(r)=a$; $\frac{1}{s}\prod_{0\le i<L(r)} \frac{\lambda_{x_i(r)}}{s+\lambda_{x_i(r)}}$ if $x_{L(r)}(r)=d$
$H_2(s)$	X''	$\frac{1}{s}\prod_{0\le i<L(r)} \frac{\lambda_{x_i(r)}}{s+\lambda_{x_i(r)}}$
$I_1(s)$	X'	$\frac{1}{s}\sum_{0\le i<L(r),x_i(r)\in D} P_{x_i(r)/r}(s)$

probabilities of these paths (and their relative contributions) are typically strongly ranked according to the number of failure transitions they contain. This is in accordance with the heuristic behind failure distance biasing and we can expect the simulation of $G_1(s)$, $H_1(s)$, and $I_1(s)$ under that biasing scheme to be particularly efficient.

4 Derivation of Confidence Intervals

This section discusses the derivation of confidence intervals for the estimates of the metrics. Consider, for instance, the derivation of confidence intervals for $\widehat{ua}(t)$. Let $\widehat{G_1}(s)$ and $\widehat{G_2}(s)$ be sample mean estimates of $G_1(s)$ and $G_2(s)$ and define $\Delta G_1(s) = \widehat{G_1}(s) - G_1(s)$, $\Delta G_2(s) = \widehat{G_2}(s) - G_2(s)$. We have (2):

$$\widehat{UA}(s) = \frac{\widehat{G_1}(s)}{1-\widehat{G_2}(s)} = \frac{G_1(s)+\Delta G_1(s)}{1-G_2(s)-\Delta G_2(s)} .$$

$\Delta G_1(s)$ and $\Delta G_2(s)$ have null expected values and the same standard deviations as, respectively, $\widehat{G_1}(s)$ and $\widehat{G_2}(s)$. For any estimate of $\widehat{UA}(s)$ of reasonable quality, the relative standard deviations of $\widehat{G_1}(s)$ and $\widehat{G_2}(s)$ will be small so that we can approximate $\widehat{UA}(s)$ by its first-order Taylor expansion on $\Delta G_1(s)$ and $\Delta G_2(s)$, obtaining:

$$\widehat{UA}(s) \approx UA(s) + \frac{1}{1-G_2(s)}\Delta G_1(s) + \frac{G_1(s)}{(1-G_2(s))^2}\Delta G_2(s) .$$

Replacing $\Delta G_1(s)$ and $\Delta G_2(s)$ by, respectively, $\widehat{G_1}(s) - G_1(s)$ and $\widehat{G_2}(s) - G_2(s)$ and taking Laplace antitransforms we obtain for $\widehat{ua}(t)$ an expression of the form:

$$h(t) + \hat{f}_1(t) + \hat{f}_2(t) , \qquad (5)$$

where $h(t)$ is non-random and $\hat{f}_1(t)$, $\hat{f}_2(t)$ are random variables, which, denoting by $(Z_M)_r(s)$ the path function associated to the metric $M(s)$ as given in Table 1 and by $(Z^*_M)_r(s)$ the function affected by the likelihood ratio, can be interpreted as the sample mean estimates of, respectively, the path functions:

$$(W_1)_r(t) = \mathcal{L}^{-1}\left\{\frac{1}{1-G_2(s)}(Z^*_{G_1})_r(s)\right\} \qquad (6)$$

$$(W_2)_r(t) = \mathcal{L}^{-1}\left\{\frac{G_1(s)}{(1-G_2(s))^2}(Z_{G_2})_r(s)\right\} . \qquad (7)$$

A similar development holds for the other metrics, where the random terms in (5) can be interpreted as the sample mean estimates of the path functions:

$$(W_1)_r(t) = \mathcal{L}^{-1}\left\{\frac{1-sH_2(s)}{[1-s(H_2(s)-H_1(s))]^2}(Z^*_{H_1})_r(s)\right\} \qquad (8)$$

$$(W_2)_r(t) = \mathcal{L}^{-1}\left\{\frac{sH_1(s)}{[1-s(H_2(s)-H_1(s))]^2}(Z_{H_2})_r(s)\right\} \qquad (9)$$

for $\widehat{ur}(t)$ and:

$$(W_1)_r(t) = \mathcal{L}^{-1}\left\{\frac{1}{1-G_2(s)}(Z^*_{I_1})_r(s)\right\} \qquad (10)$$

$$(W_2)_r(t) = \mathcal{L}^{-1}\left\{\frac{I_1(s)}{(1-G_2(s))^2}(Z_{G_2})_r(s)\right\} \qquad (11)$$

for $\widehat{mdt}(t)$.

Then, $\hat{f}_1(t)$ and $\hat{f}_2(t)$ have asymptotically a normal distribution and approximate confidence intervals for the metrics can be derived by using the normal distribution with variance:

$$\hat{\sigma}^2 = \hat{\sigma}_1^2 + \hat{\sigma}_2^2$$

where $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ are estimates for the variances of, respectively, $\hat{f}_1(t)$ and $\hat{f}_2(t)$. These estimates can be obtained by sampling the path functions (6–11). The computation of these path functions requires the knowledge of $G_1(s)$ and $G_2(s)$, $H_1(s)$ and $H_2(s)$, or $I_1(s)$ and $G_2(s)$, respectively, which are precisely the quantities under estimation. This problem is solved as follows.

We organize each simulation stream in substreams, each having a length which approximately doubles the length of the previous substream for the same low-level metric. The streams for the two low-level metrics run in parallel with optimal allocation of events according to their contributions to the variance of the metric estimate. The first substreams use initial rough estimates for the Laplace transforms of the low-level metrics obtained running short presimulation substreams. The following substreams use the estimates for the Laplace transforms available at the end of the previous substreams.

The Laplace transforms $G_2(s)$, $H_2(s)$ are estimated by the sample means of the corresponding path function and, at the end of the kth substream, $\hat{\sigma}_2^2$ is computed by:

$$\hat{\sigma}_2^2 = \frac{s_k^2}{M},$$

where M is the total number of sampled paths and s_k^2 is the sample variance of $(W_2)_r(t)$ computed in the kth substream. Computing a sample variance using all sampled paths is not possible because, being different the estimates for the Laplace transforms of the low-level metrics used in each substream for the computation of $(W_2)_r(t)$, we are in fact sampling a different random variable $(W_2)_r(t)$ at each substream.

Simulation of $G_1(s)$, $H_1(s)$, $H_1(s)$ is optimized using an adaptive scheme in which the values of the biasing parameters are updated at the end of each substream to minimize the variance of the sampled path function. The estimates for these Laplace transforms are computed by weighting optimally the sample means obtained for the different substreams so as to minimize $\hat{\sigma}_1^2$. This is done to take into account the decrease in variance of the sampled path function resulting from the progressive optimization of the biasing parameters. The estimate for the variance $\hat{\sigma}_1^2$ at the end of the kth substream is computed by:

$$\hat{\sigma}_1^2 = \sum_{i=1}^{k} \alpha_i^2 \frac{s_k^2 + R_i - R_k}{M_i},$$

where α_i is the weight associated to the sample mean of the ith substream, M_i is the number of paths of the ith substream, s_k^2 is the sample variance of $(W_1)_r(t)$ obtained in the kth substream, and R_i is the value given by a symbolic estimator (which takes into account all the samples) of $E[((W_1)_r(t))^2]$ for the values of the biasing parameters used in the ith substream. See [12] for details. Using $s_k^2 + R_i - R_k$ is more robust than using s_i^2, which can be dangerous when the model is badly-behaved and the variance is only well estimated after a large number of sampled paths.

5 Experimental Analysis

In our implementation of the simulation methods we compute Laplace antitransforms using the inversion formula [14]:

$$f(t) = \frac{e^{at}}{t}\left\{\frac{F(a)}{2} + \sum_{k=1}^{\infty}(-1)^k Re[F(a + i\,k\,\frac{\pi}{t})]\right\},$$

where a is a parameter which is chosen large enough to make the relative discretization error smaller than $DERROR = 10^{-4}$. The convergence of the series is accelerated by the epsilon algorithm, and the accelerated series is truncated when the relative difference between the two last truncated values is smaller than $TERROR = 10^{-4}$. The value for a and the number of abscissae at which the Laplace transforms are bookkept are chosen after running the presimulation substreams and rough estimates for the Laplace transforms are available. For the presimulation we use oversized values for these parameters. Recovery procedures are implemented in case the values chosen after the presimulation turn out to be insufficient to achieve the specified accuracy. In all cases we have tested, between 20 and 30 terms have been enough to achieve convergence. More sophisticated inversion formulae [15] could be used to get estimates of the metrics at several values of t using slightly more Laplace transforms.

A number of experiments have been carried out using models of small size and comparing the results obtained by the simulation methods presented here with those obtained numerically by METFAC [16] and we have found that the methods are robust. This will be illustrated using the fault-tolerant database example described in [7]. The system is made up of two front-ends, two databases and two processing subsytems and each processing subsystem includes one switch, one memory and two processors. The system is up if at least one front-end, one database, and one processing subsystem are operational, and a processing subsystem is operational if the switch, the memory, and at least one of the processors are operational. All components fail with rates $1/2400\ hrs^{-1}$, except the processors which fail with rate $1/120\ hrs^{-1}$. A processor failure is propagated to one database with probability 0.01. There is a single repairman which uses a preemptive discipline with priority given first to databases and front-ends, next to memories and switches, and last to processors. Components of the same priority are chosen at random and all components have a mean repair time of 1 hour.

The proposed simulation methods were run for a wide range of values of t with a goal of a 99% confidence interval of $\pm 2\%$ and a limit of 500,000 events. Tables 2, 3, and 4 give the exact values computed by numerical methods, the estimates obtained by simulation, and the number of simulated events required to achieve the specified tolerance (including the events sampled in the presimulation substreams). In all cases, the exact value is within the predicted interval confidence and a small number of events is enough to achieve the specified tolerance.

Our methods differ from those previously proposed [5, 10] in the use of estimator decomposition techniques and the use of the failure distance biasing scheme. A comparison isolating the impact of each improvement will be presented using a large reallistic example.

The example is the fault-tolerant data processing system whose architecture is shown in Figure 2. A dual configuration of data processing units (DPU's) command control subsystems located at remote sites. Each control subsystem comprises two redundant control units (CU's) working in hot-standby redundancy. The system can be accessed through two redundant front-ends connected to the DPU's. The DPU's and CU's communicate using a redundant local area network (LAN) to which each DPU and each CU has access through dedicated communication processors (CP's). Components fail with constant rates $\lambda_{FE} = 2 \times 10^{-4}$, $\lambda_{DPU} = 10^{-3}$, $\lambda_{CU} = 2 \times 10^{-4}$, $\lambda_{CP} = 5 \times 10^{-5}$, and $\lambda_L = 10^{-4}$, respectively. Two failed modes are considered for the DPU's: "soft" and "hard". The first mode occurs with probability 0.9 and can be recovered by an operator restart; the second mode occurs with probability 0.1 and requires hardware repair. Coverage is assumed perfect for all faults except those of the DPU's, which take the system down with a probability of 0.01. Lack of coverage is modelled by propagating the failure of one DPU to the other DPU. There are three repair teams. The first repairs LAN's and CP's, with preemptive priority given to LAN's. The second repairs FE's, DPU's and CU's in "hard" failed mode, with preemptive priority given first to DPU's, next to FE's, and last to CU's. The third makes DPU restarts. Each team includes only one repairman. Failed components with the same repair priority are taken at random for repair. Components are repaired with rates $\mu_{FE} = 0.5$, $\mu_{DPU_h} = 0.5$, $\mu_{DPU_s} = 4$, $\mu_{CU} = 0.5$, $\mu_{CP} = 0.5$, and $\mu_L = 0.2$, respectively.

The system is considered up if one unfailed DPU can communicate with at least one unfailed CU of each control subsystem. Different LAN's can be used for communication between the active DPU and the active CU of each control subsystem, but the communication has to be direct, i.e., involving only one CP of each unit and one LAN. The resulting CTMC has about 4.6×10^{11} states, which clearly precludes both its generation and numerical solution.

Due to space limitations we will only present experimental results for $ur(t)$. Comparison of our methods for $iua(t)$ with that proposed in [10] gives qualitatively identical results.

The method proposed in [5] for $ur(t)$ (making abstraction of the biasing scheme used to sample transitions) will be denoted by F (forcing). The method proposed in [10] for $ur(t)$ (also making abstraction of the biasing scheme used to sample transitions) will be called PC (partial conditioning). The latter allows the computation of estimates at several t, but in our experiments we only computed the metric at a given value t and the simulation of the path under PC was aborted when the sum of the sampled holding times exceeded t, since in this case the path currently sampled has a null contribution.

We run F and PC with failure biasing (FB), the same methods with failure distance biasing (FDB), and the proposed method based on (3) and using failure distance biasing for several values of t with a goal of a 99% confidence interval of $\pm 2\%$ and a limit of 500,000 events. Our implementation of F and PC under both FB and FDB includes an adaptive optimization scheme for the biasing parameters. We turned biasing off in our method when $MAXREP = 2$ repair transitions are sampled in the same path. This improves the robustness of the biasing scheme without affecting its efficiency significantly. This was however not done under F and PC since it would turned the biasing scheme off prematurely when $t \gg u$ and long paths have to be sampled. The CPU time per simulated event in our method is about 50% higher than in F and PC due to the computation of the Laplace transforms and the invocations of the Laplace transform inversion algorithm, and we present CPU times instead of sampled events for comparison purposes. These times were measured on a SUN 3/260. Figure 3 shows the results, using estimates for the required CPU times when the goal confidence interval was not achieved after 500,000 events. The speedup of the proposed method over previous ones (F and PC with failure biasing) is between 3 and 4 orders of magnitude. In fact none of the previous methods can achieve the goal confidence interval in a practical amount of time whereas our method achieves it in about 2 minutes in all cases. F and PC with failure distance biasing are slightly better than the proposed method when t is smaller or of the order of magnitude of h_u (172 hours). As t increases, the performance of F and PC with failure distance biasing rapidly deteriorates for the reasons explained in Section 3. Figure 4 gives the average path lengths, clearly confirming the arguments given there.

6 Conclusions

We have proposed new methods for the transient simulation of repairable fault-tolerant systems which are robust and efficient for any value of t. For a typical large example speedups of 3 to 4 orders of magnitude over previously proposed methods have been reported. These speedups are the result of using estimator decomposition techniques and the more efficient failure distance biasing scheme. Previously proposed methods under failure distance biasing tend to be slightly more efficient than the method proposed here only

Table 2: $ua(t)$ simulation results for the fault-tolerant database system.

t	exact	estimate	events
0.01	1.284×10^{-10}	$1.304 \times 10^{-10} \pm 2.6 \times 10^{-12}$	18,643
0.1	1.219×10^{-8}	$1.235 \times 10^{-8} \pm 2.4 \times 10^{-10}$	18,208
1	7.399×10^{-7}	$7.389 \times 10^{-7} \pm 1.5 \times 10^{-8}$	21,530
10	3.425×10^{-6}	$3.365 \times 10^{-6} \pm 6.7 \times 10^{-8}$	61,476
100	3.431×10^{-6}	$3.373 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,611
10^3	3.431×10^{-6}	$3.373 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,611
10^4	3.431×10^{-6}	$3.373 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,611
10^5	3.431×10^{-6}	$3.373 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,611

Table 3: $ur(t)$ simulation results for the fault-tolerant database system.

t	exact	estimate	events
0.01	1.289×10^{-10}	$1.294 \times 10^{-10} \pm 2.6 \times 10^{-12}$	10,094
0.1	1.261×10^{-8}	$1.266 \times 10^{-8} \pm 2.5 \times 10^{-10}$	9,813
1	1.023×10^{-6}	$1.022 \times 10^{-6} \pm 1.9 \times 10^{-8}$	15,963
10	2.960×10^{-5}	$2.976 \times 10^{-5} \pm 5.9 \times 10^{-7}$	36,559
100	3.353×10^{-4}	$3.351 \times 10^{-4} \pm 6.7 \times 10^{-6}$	39,051
10^3	3.387×10^{-3}	$3.388 \times 10^{-3} \pm 6.7 \times 10^{-5}$	38,706
10^4	3.340×10^{-2}	$3.336 \times 10^{-2} \pm 6.6 \times 10^{-4}$	37,811
10^5	0.2881	$0.2862 \pm 5.6 \times 10^{-3}$	26,061

Table 4: $iua(t)$ simulation results for the fault-tolerant database system.

t	exact	estimate	events
0.01	4.287×10^{-11}	$4.354 \times 10^{-11} \pm 8.6 \times 10^{-13}$	18,643
0.1	4.124×10^{-9}	$4.182 \times 10^{-9} \pm 8.1 \times 10^{-11}$	18,572
1	2.842×10^{-7}	$2.859 \times 10^{-7} \pm 5.3 \times 10^{-9}$	18,622
10	2.639×10^{-6}	$2.608 \times 10^{-6} \pm 5.2 \times 10^{-8}$	54,592
100	3.351×10^{-6}	$3.307 \times 10^{-6} \pm 6.6 \times 10^{-8}$	61,866
10^3	3.423×10^{-6}	$3.372 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,645
10^4	3.430×10^{-6}	$3.386 \times 10^{-6} \pm 6.6 \times 10^{-8}$	66,517
10^5	3.431×10^{-6}	$3.373 \times 10^{-6} \pm 6.7 \times 10^{-8}$	62,606

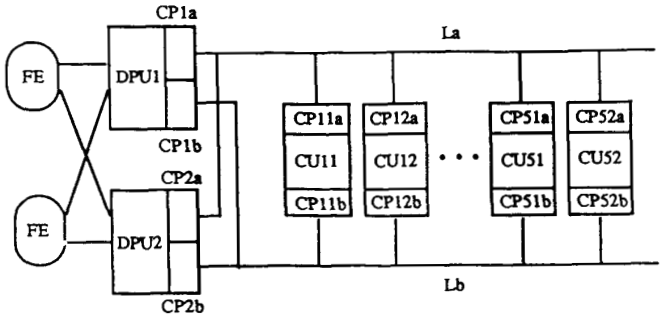

Figure 2: Fault-tolerant data processing system.

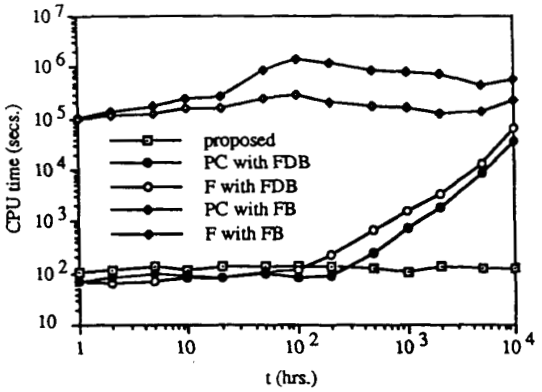

Figure 3: CPU times required to achieve a 99% ±2% confidence interval in the simulation of $ur(t)$ for the fault-tolerant data processing system.

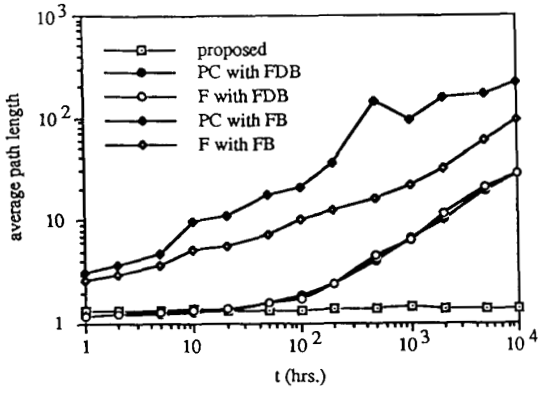

Figure 4: Average path lengths in the simulation of $ur(t)$ for the fault-tolerant data processing system.

when t is small or of the order of magnitude of the mean holding time in the state with all components unfailed h_u, but their performance rapidly deteriorates as t increases. This behavior is specially problematic for the reliability, since this metric is often used to characterize the long-term dependability behavior of the system and values of t several orders of magnitude higher than h_u will be typically of interest, particularly for large systems, where λ_u can be large.

APPENDIX

In this appendix we obtain the formulations (2), (3) for the Laplace transforms of, respectively, $ua(t)$ and $ur(t)$.

Let $p_i(t) = P[X(t) = i \mid X(0) = u]$ and $\phi^u_{ui}(t)$ the probability that, given $X(0) = u$, $X(t) = i$ and X has not hit u in $[0,t]$. Remembering the definition of $g_2(t)$ given in Section 3 and denoting by $g_2^{(k)}(t)$ the n-fold convolution of $g_2(t)$ with itself and by $*$ the convolution operator, we can write:

$$\begin{aligned} p_i(t) &= \sum_{k=0}^{\infty} P[X(t) = i \wedge X \text{ has hit } u \\ &\qquad k \text{ times in } [0,t] \mid X(0) = u] = \\ &= \sum_{k=0}^{\infty} g_2^{(k)}(t) * \phi^u_{ui}(t). \end{aligned} \tag{12}$$

The point unavailability $ua(t)$ can be computed as $\sum_{i \in D} p_i(t)$ and, according to its definition, $g_1(t) = \sum_{i \in D} \phi^u_{ui}(t)$. Then, using (12):

$$\begin{aligned} UA(s) &= \sum_{i \in D} P_i(s) = \sum_{k=0}^{\infty} G_2^k(s) \sum_{i \in D} \Phi^u_{ui}(s) = \\ &= \sum_{k=0}^{\infty} G_2^k(s) G_1(s) = \frac{G_1(s)}{1 - G_2(s)} \end{aligned}$$

The formulation for the Laplace tranform of $ur(t)$ can be obtained by resorting to semi-Markov process

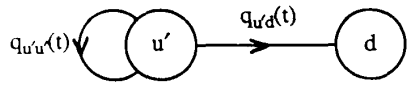

Figure 5: Transition graph of the semi-Markov process Y'.

theory [17]. Let Y be the transient CTMC with state space $U \cup \{d\}$ which is obtained from X by collapsing D into an absorbing state d and consider the semi-Markov process Y' whose transition graph is given in Figure 4 and is defined so that, provided $Y(0) = u$: a) $Y'(t) = u'$ if $Y(t) \in U$, b) $Y'(t) = d$ if $Y(t) = d$, and c) Y' makes a transition into u' whenever Y enters u.

The functions $h_1(t)$ and $h_2(t)$ defined in Section 3 can be expressed in terms of the kernel functions of Y' as:

$$h_1(t) = q_{u'd}(t) \tag{13}$$

$$h_2(t) = q_{u'u'}(t) + q_{u'd}(t)\,. \tag{14}$$

By construction, $ur(t)$ is the transition probability $p_{u'd}(t)$. Then, using semi-Markov process theory we can write:

$$ur(t) = p_{u'd}(t) = q_{u'd}(t) + \int_0^t p_{u'd}(t-\tau)\, dq_{u'u'}(\tau)\,.$$

Considering that $q_{u'u'}(0) = 0$ and taking Laplace transforms we obtain:

$$\begin{aligned} UR(s) &= Q_{u'd}(s) + sP_{u'd}(s)Q_{u'u'}(s) = \\ &= Q_{u'd}(s) + sUR(s)Q_{u'u'}(s)\,. \end{aligned}$$

Finally, solving in $UR(s)$ and using (13, 14):

$$UR(s) = \frac{Q_{u'd}(s)}{1 - sQ_{u'u'}(s)} = \frac{H_1(s)}{1 - s(H_2(s) - H_1(s))}\,.$$

References

[1] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg, and K.S. Trivedi, "The System Availability Estimator," *Proc. 16th Int. Symp on Fault-Tolerant Computing FTCS-16*, 1986, pp. 84–89.

[2] J.A. Carrasco, "Hierarchical, Object-oriented Modeling of Fault-Tolerant Computer Systems," *Proc. CompEuro 91 Conference*, 1991, pp. 452–456.

[3] W.J. Stewart and A. Goyal, "Matrix Methods in Large Dependability Models," IBM Research Report RC 11485, Yorktown Heights, Nov. 1985.

[4] A. Reibman and K. Trivedi, "Numerical Transient Analysis of Markov Models," *Computer Operations Research*, vol. 25, no. 1, pp. 19–36, 1988.

[5] E.E. Lewis and F. Böhm, "Monte Carlo Simulation of Markov Unreliability Models," *Nuclear Engineering and Design*, vol. 77, 1984, pp. 49–62.

[6] T. Zhuguo and E.E. Lewis, "Component Dependency Models in Markov Monte Carlo simulation," *Reliability Engineering*, vol. 13, 1985, pp. 45–62.

[7] A.E. Conway and A. Goyal, "Monte Carlo Simulation of Computer System Availability/Reliability Models," *Proc. 17th Int. Symp on Fault-Tolerant Computing FTCS-17*, 1987, pp. 230–235.

[8] A. Goyal, P. Heidelberg, and P. Shahabuddin, "Measure Specific Dynamic Importance Sampling for Availability Simulations," *Proc. 1987 Winter Simulation Conference*, A. Thesen, H. Grant and W. D. Kelton (eds.), pp. 351–357.

[9] P. Shahabuddin, V.F. Nicola, P. Heidelberger, A. Goyal, and P.W. Glynn, "Variance Reduction in Mean Time to Failure Simulations," *Proc. 1988 Winter Simulation Conference*, M. Abrams, P. Haigh, and J. Comfort (eds.), pp. 491–499.

[10] A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola, and P.W. Glynn, "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems," IBM Research Report RC 14772, Yorktown Heights, Nov. 1989.

[11] R. M. Geist and M. K. Smotherman, "Ultrahigh Reliability Estimates Through Simulation," *Proc. Ann. Reliability and Maintainability Symp.*, January 1990, pp. 1–6.

[12] J. A. Carrasco, "Failure distance-based Simulation of Repairable Fault-Tolerant Systems," *Proc. 5th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, 1991, pp. 337–351.

[13] J.M. Hammersley and D.C. Handscomb, *Monte Carlo Methods*, Metheun, London, 1975.

[14] K.S. Crump, "Numerical Inversion of Laplace Transforms Using a Fourier Series Approximation," *J. of the ACM*, vol. 23, no 1, January 1976, pp. 89–96.

[15] F. DeHoog, J. Knight, and A.N. Stokes, "An improved method for numerical inversion of Laplace transforms," *SIAM J. Sci. Stat. Comput.*, vol. 3, no. 3, pp. 357–366, 1982.

[16] J.A. Carrasco and J. Figueras, "METFAC: Design and Implementation of a Software Tool for Modeling and Evaluation of Complex Fault-Tolerant Computing Systems," *Proc. 16th Int. Symp. on Fault-Tolerant Computing FTCS-16*, 1986, pp. 424–429.

[17] E. Çinlar, *Introduction to Stochastic Processes*, Prentice-Hall, Englewood Cliffs, 1975.