IMPORTANCE SAMPLING TO EVALUATE REAL-TIME SYSTEM RELIABILITY: A CASE STUDY

G. Durairaj, I. Koren, C.M. Krishna
Dept. of Electrical and Computer Engineering
University of Massachusetts, Amherst
gopinathd@yahoo.com; {koren, krishna}@ecs.umass.edu

ABSTRACT

Real-time distributed computers are often used in life-critical applications. However, the complexity of such systems calls for extensive simulation studies to validate their performance and reliability before a design can be accepted and a prototype constructed. A simulator testbed has been built to model a variety of such systems quickly from a few basic building blocks.

Life-critical applications require reliability levels so high that brute-force simulation to validate these levels would take weeks of computer time. In this paper, we present studies we have conducted into the use of Importance Sampling in simulating real-time systems. While many theoretical studies have been published on this technique, there are practical studies available in the literature. This paper presents a interesting case-study of the use of Importance Sampling in an increasingly important branch of computer engineering.

Importance Sampling may not work for all cases and over all parameter ranges. In this paper we are interested in finding out whether (and how well) this scheme works for the case of distributed real-time systems and also the range of failure bias values for which it works well. Specifically we look at the implementation of two heuristics called 'forcing' and 'failure biasing' in the testbed. This was validated by comparing the reliability estimates with that of normal (very long) simulation. The effect of the failure bias on the dynamics of the scheme are also investigated to provide readers with some guidance on choosing appropriate bias values.

[0]Acknowledgement: This work was partially supported by the Defense Advanced Projects Agency and the Navy SPAWAR under contract N0039-94-C-0165. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Projects Agency, SPAWAR, or the US Government.

1 Introduction

Real-time, distributed, fault-tolerant computers are increasingly being used in critical applications like aircraft control, air traffic control, factory automation etc. Such computers need to be highly reliable and are expected to deliver critical outputs in a timely fashion, even in the presence of a few component failures.

Modeling the reliability of such computers is extremely complex. Reliability in real-time systems is a *dynamic*, rather than a *static*, concept [16]. In other words, real-time reliability translates into the system being able to avoid missing critical deadlines. The ability to meet deadlines is a complex function of a number of parameters: the hardware, the operating environment, the interconnection network and communication protocol, the executive software (including the task assignment and scheduling algorithms), and the fault-tolerance procedures used. It is practically impossible to obtain an analytical model that models reliability as a function of all these parameters and their interactions to high degree of accuracy. Thus analytical models can be used as rough guides of system reliability than definitive predictors. Instead, a designer has often to turn to simulation for reliability evaluations that are sufficiently accurate for fine-grain design decisions and prototype-building.

However, simulation has the drawback of requiring very long computation times, even with today's high-speed computers. This is especially true when reliability levels of 10^{-5} or lower must be validated: to obtain confidence intervals that are sufficiently small can require computational runs lasting many days or weeks. Several methods have been proposed in the simulation literature to reduce such long runs [23]. One fairly new method, which has been attracting some attention, is Importance Sampling.

A good analogy for understanding Importance Sampling is how processor lifetimes are often obtained experimentally. Since these lifetimes are of the order of thousands of hours, it would take a long time to obtain sufficient statistics on processor lifetime if we simply ran them to failure under normal conditions. Instead, one might measure their lifetimes under high temperature. The way in which failure is accelerated by high ambient temperature is fairly well known. Hence, we can obtain high-temperature processor lifetimes by experiment, and then use the computed acceleration factor to determine their lifetimes under room temperature.

Importance Sampling has the same philosophy. Instead of simulating the system under the specified parameters and then waiting a very long time before enough failure data can be recorded, we simulate the system under stressful conditions, which increases the rate at which failures happen. Once sufficient failure data have been recorded, we can then make a theoretical computation to predict what the failure probability or rate would have been if the system had been running under the specified parameter set (rather than the "stressed" parameters).

Importance Sampling has great potential for speeding up rare-event simulation. However, anecdotal evidence suggests that it is a somewhat temperamental technique, which can some-

times do quite badly in reducing the simulation variance. The purpose of this paper is to report a case study of the use of Importance Sampling to real-time systems.

The framework for our study is RAPIDS (**R**ecovery **P**olicies for Real-time **D**istributed **S**ystems), which is a simulator testbed for real-time systems. Our work consisted of doing the following:

- Implementing Importance Sampling in the RAPIDS simulator.
- Analysing the expected behavior of such a scheme.
- Validating the implementation.
- Investigating the tunable parameters in the scheme and providing guidelines on their use.

We do not make any new *theoretical* innovations in this paper; rather the contribution of this paper is in the case-study it provides for the use of Importance Sampling in real-time systems. Importance Sampling is a somewhat temperamental technique, and the guidance provided in this paper on the choice of biasing parameters and when to use Importance Sampling is likely to be of interest to other workers.

2 Technical Background

2.1 The RAPIDS Simulator

RAPIDS [1] is a simulator testbed that has extensive facilities to specify a distributed real-time system completely and to observe its behavior. The system to be modeled can be viewed as a collection of computing nodes that are connected by a communication network (see Figure 1). RAPIDS provides facilities for the user to specify the system by giving detailed information about the following:

- The computing nodes and their executive software (including the tasks scheduling algorithm and the checkpointing scheme.).
- The real-time tasks that run on the nodes (modeled as periodic, aperiodic or sporadic tasks).
- The algorithms to be used for task allocation.
- The messages that are exchanged by the nodes as part of the execution or maintenance (modeled as data and control messages).
- The interconnection network.

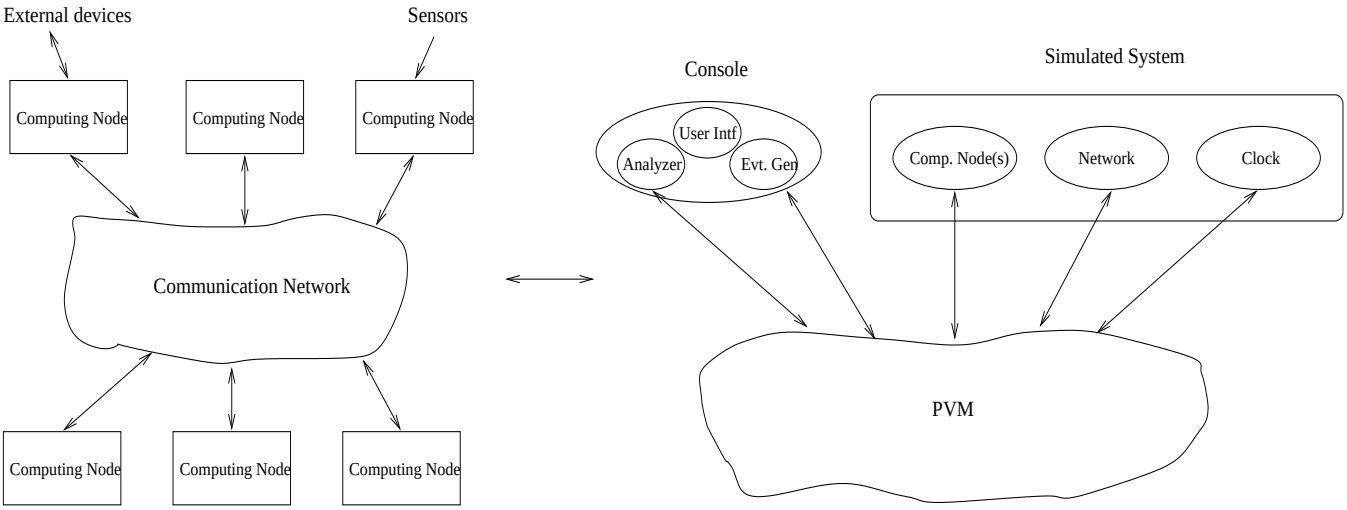

Figure 1: A typical distributed real-time system and the RAPIDS Implementation

- The fault recovery policy to be used when a fault is detected on one of the nodes [32].

The simulator models this system by having a set of processes that communicate with each other by means of a portable message-passing library called PVM [8]. There is a process to model each of the computing nodes, the network interconnect, the central clock and the console. The console process controls the entire simulation run and its responsibilities include generating the events for all the simulation entities (implemented as the 'Event Generator') and analyzing the results of a simulation run (implemented as the 'Analyzer'). Importance Sampling implementation will be done in this process. RAPIDS has a graphical user interface showing the status of the simulated system including:

- Summaries about the tasks meeting or missing their deadlines.
- Run time graphical information about
  - Task allocation
  - Task scheduling
  - Fault occurences

Additional implementationn details of the simulator can be found in [1].

2.2 Importance Sampling

In Importance Sampling, we change the probabilistic dynamics of the system for simulation purposes. The simulation is biased so that sample paths ending in system failure are generated disproportionately often. We then make adjustments to the sample outputs to unbias (i.e., to correct the biasing of) the estimator before computing it. A good reference for the mechanics of this technique is Hammersley and Handscomb [13]. Generalizations to stochastic systems are given by Glynn and Iglehart [9]. The heuristic of failure biasing, which we use in this paper, was first proposed by Lewis and Bohm [19] in the reliability estimation of nuclear reactors. In Goyal, Heidelberger and Shahabuddin [11] it was adapted to estimating the unavailability of highly reliable Markovian systems. In Shahabuddin *et al.* [24], it was used for the estimation of the MTTF using a regenerative method. Generalizations of these heuristics along with new ones have been investigated in the unifying paper of Goyal *et al.* [12]. The reader is directed to these references for a full description of the theory underlying the Importance Sampling technique. In the remainder of this section, we provide the minimum background required to understand our implementation of Importance Sampling in the RAPIDS simulator.

2.2.1 Introduction to Importance Sampling

Suppose X is a random variable with density function $p(x)$. We are interested in estimating the probability, θ, that X is in the set of failed states, A. This quantity is given by

$$\theta = \int_{-\infty}^{+\infty} 1_{\{x \in A\}} p(x) dx = E_p[1_{\{X \in A\}}]$$

where the subscript p denotes sampling from the density $p(\cdot)$ and $1_{x \in A}$ is the indicator of the set A. If we were using normal (i.e., traditional) simulation, we would draw n samples, $X_1, X_2, \cdots, X_n$, of X from the density $p(x)$ and compute $\theta = (1/n) \sum_{i=1}^{n} 1_{X_i \in A}$. Let σ be the estimate of the standard deviation of these readings. From elementary probability, we have $\sigma = \sqrt{\theta(1-\theta)}$. We can find, for any given $\alpha \in [0,1]$, the $100\alpha\%$ confidence intervals relating to our estimate of θ. The half-width of these confidence intervals is given by $z_{\alpha/2}\sigma/\sqrt{n}$, where $z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile point of the standard normal distribution.

If $\alpha = 0.9$, $z_{\alpha/2} = 1.282$. If our goal was to keep the confidence interval width to about 10% of the mean, we would have:

$$\begin{aligned} 1.282\sqrt{\theta(1-\theta)/n} &\leq 0.1\theta \\ \Rightarrow n &\approx 100 \times 1.282^2 \times (1/\theta - 1) \end{aligned}$$

n therefore grows very rapidly as θ decreases: for example, if $\theta = 10^{-6}$, $n \sim 10^8$, which would make for very long simulation runs. It is clear that normal simulation is not always practical for highly reliable systems.

The idea of importance sampling is to pick another suitable density function $p'(x)$ and then to write:

$$\begin{aligned}\theta &= \int 1_{\{x\in A\}}\frac{p(x)}{p'(x)}p'(x)dx \\ &= E_{p'}\left[1_{\{X\in A\}}\frac{p(X)}{p'(X)}\right] \\ &= E_{p'}[1_{\{X\in A\}}L(X)]\end{aligned}$$

where $L(x) = p(x)/p'(x)$ is called the *likelihood ratio* and the subscript p' indicates that we are taking the expectation over the density function $p'(\cdot)$. All this requires only that $p'(x) > 0$ for all $x \in A$ for which $p(x) > 0$.

The above equation is the key to the Importance Sampling technique. We pick a density, $p'(x)$, which has a greater chance of resulting in $X \in A$ than the original density, $p(x)$. Then, we obtain n samples, $X_1, \cdots, X_n$, and estimate θ through the equation $\theta = (1/n)\sum_{i=1}^{n} 1_{X_i\in A}L(X_i)$.

2.2.2 Implementation Heuristics

Simulation consists of constructing a state model of the system, and deciding when the system enters a given state when the next state transition will occur, and what the next state will be. To speed up the simulation, we use two techniques introduced by Lewis and Bohm [19] and Shahabuddin [25]. The first, called *forcing*, increases the rate at which state transitions occur. The second, called *balanced failure biasing*, biases the system towards more faults. Nakayama [20] provides a study of this, and a related, biasing technique.

Each of these techniques has a likelihood ratio associated with it. The overall likelihood ratio associated with using both of them is just the product of the individual ones.

Let us start by considering forced (or accelerated) state transitions. Let $f(t|t', k')$ denote the density function for the transition instant, t, given that in its last state transition, the system entered state k' at time t'. All times are absolute, global quantities. Forcing consists of replacing $f(\cdot)$ in the simulation by another density function, $\tilde{f}(\cdot)$, which makes for more state transitions. We now proceed to make this more concrete.

For each node i in the system, the Poisson fault arrival rate is given by λ_i. Permanently failed nodes cannot be repaired; however, if node i suffered a transient fault, it does recover and its recovery time is exponentially distributed with parameter μ_i. λ is the total failure rate out of the current system state and is equal to the sum of the failure rates of all active, non-faulty, nodes. μ is the total recovery rate out of the current state and is equal to the sum of the recovery rates of all the nodes currently suffering a transient fault. Finally γ is the total transition rate out of the current state, i.e., $\gamma = \lambda + \mu$. As a node goes faulty or gets repaired, the system state changes and these variables are updated.

Usually the failure rate of the nodes is very small when compared to the transient-failure recovery rate. Therefore γ is small when no failed components are present. In such a case, the transition rate is boosted by taking

$$\tilde{f}(t|t',k') = \begin{cases} \frac{\tilde{f}(t|t',k')}{1-e^{-\gamma(T-t')}} & \text{for } t' \le t \le T, \\ 0 & \text{otherwise} \end{cases}$$

where T is the mission time of the system. This heuristic is applied only when γ is small, $\gamma(T-t') \ll 1$ and there is only a small chance that an additional transition will take place before the end of mission time T.

The likelihood ratio associated with forced transitions is then given by

$$\frac{f(t|t',k')}{\tilde{f}(t|t',k')} = 1 - e^{-\gamma(T-t')}$$

The second acceleration technique is failure biasing. As mentioned previously, it consists of artificially increasing the chance that the system will suffer more node failures.

Define the state-transition probability, $\pi_{k',k}$ as the probability that if the system is in state k', its next state will be k. We say that (k',k) is a *failure transition* if the failure of some node results in the system making the transition from k' to k. Define F as the set of all the failure transition pairs. Then, in the normal simulation (and in the actual system), the probability that the system currently in state k' makes a transition associated with a node failure is given by $\Pi_{k'} = \sum_{(k',k)\in F} \pi_{k',k}$. Failure biasing consists of boosting this transition probability for each state, k'. To implement failure biasing, pick some suitably large quantity, ϕ, and define a state-transition transition matrix $\tilde{\pi} = [\tilde{\pi}_{k',k}]$ such that

$$\sum_{(k',k)\in F} \tilde{\pi}_{k',k} = \phi.$$

Clearly, this results in

$$\sum_{(k',k)\notin F} \tilde{\pi}_{k',k} = 1 - \phi.$$

The simulator determines which node is struck by a fault by maintaining an ordered list of all the n eligible nodes and generating a random number i uniformly distributed between 0 and $n-1$. i is the index of the node experiencing the fault. The likelihood ratio associated with this failure biasing is then given by

$$\frac{\pi_{k',k}}{\tilde{\pi}_{k',k}} = \frac{n\lambda_i}{\phi\gamma}$$

If it is a recovery, the simulator decides which node recovers from a transient by looking at the repair rates of the nodes currently affected by a transient fault. For this we order the eligible

nodes and determine the particular node i by using the following formula

$$\sum_{i'=1}^{i} \mu_{i'} < \frac{\mu(\varepsilon' - \phi)}{1 - \phi} \leq \sum_{i'=1}^{i+1} \mu_{i'}$$

The likelihood ratio associated with the forcing is then given by

$$\frac{\pi_{k',k}}{\tilde{\pi}_{k',k}} = (1 - \phi)^{-1} \frac{\mu}{\gamma}$$

The overall likelihood ratio associated with both failure biasing and forcing is just the product of the individual likelihood ratios for failure biasing and forcing, respectively.

3 Implementation

To implement Importance Sampling in our model (recall Figure 1), we do not alter the simulated system: we only need to modify the generation of the fault arrival/repair events and the way the reports are analyzed. We measure the unreliability of the system by repeating the simulations to get individual samples. The general approach to be followed is summarized in Figure 2.

The logical place to implement Importance Sampling is in the console. To be more precise, we can implement this in the event generator and the analyzer. The event generator has the following responsibilities:

- Decide the time of the next system state transition. Implement "forcing" to accelerate the state changes.
- Decide whether the next transition is a fault arrival or repair. Implement "failure biasing" to push the system towards more component faults.
- Calculate the likelihood ratio associated with each "change of measure" and store this value along with the event.

The analyzer has the following responsibilities:

- Receive reports from the simulated system.
- If it corresponds to one of the above mentioned "change of measure", update the current simulation weight.
- If the system fails within the mission time, set the simulation output to the current value of the likelihood ratio; else set the simulation output to zero.

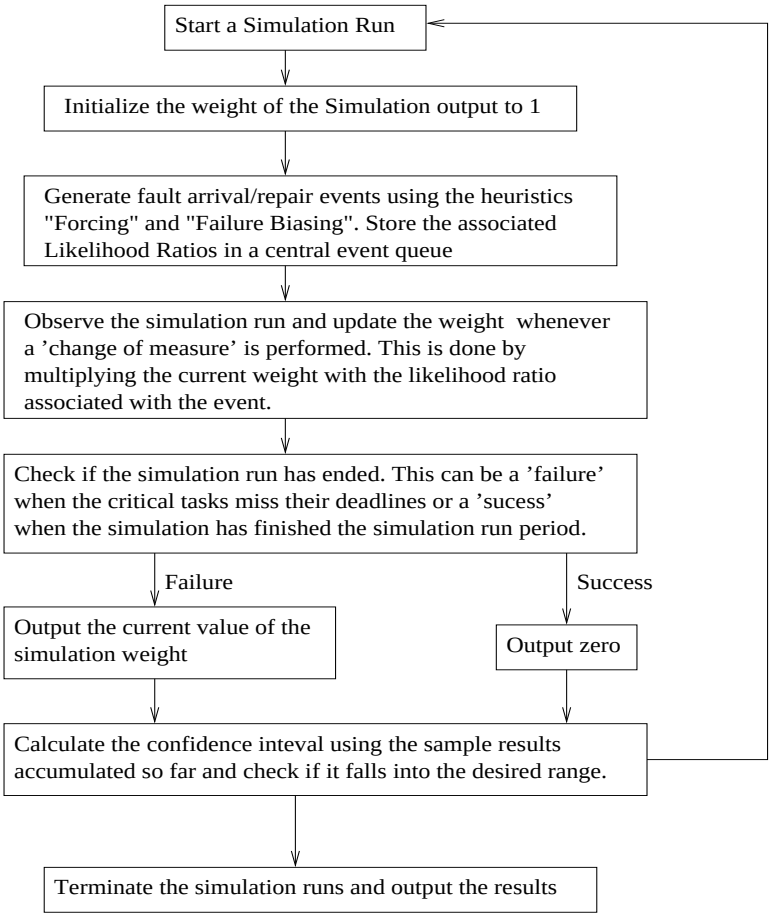

Figure 2: Flowchart of the implementation

3.1 Expected Behavior of Importance Sampling

Before we carry out the validation of the implementation, it is imperative to understand the expected behavior of the Importance Sampling scheme. The question is for what regions of unreliability values this approach works well. Intuitively we know that there should be some range of the unreliability values beyond which normal simulation will be better than Importance Sampling.

We will use the Relative Error (RE) defined below as the performance measure of the estimation scheme (normal simulation and Importance Sampling).

$$RE = \frac{z_{\alpha/2}\sigma}{\sqrt{n}\theta}$$

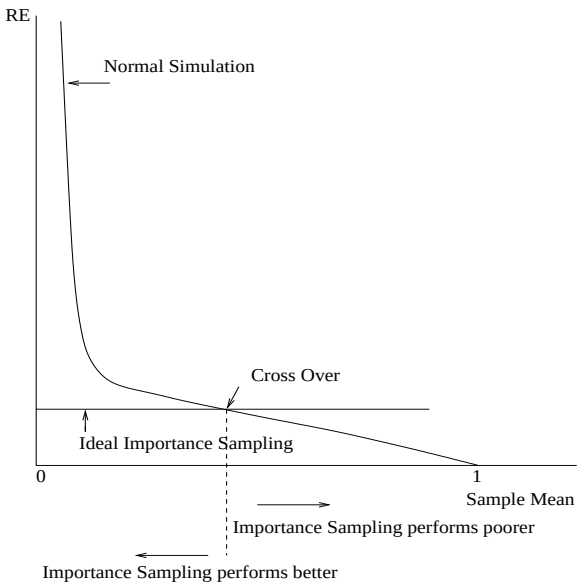

Figure 3: RE of Normal Simulation and Importance Sampling

For the case of the normal simulation, this becomes

$$RE = \frac{z_{\alpha/2}\sqrt{\theta(1-\theta)}}{\sqrt{n}\theta}$$

for a fixed number of samples, as the failure event becomes rarer (i.e., $\theta \to 0$) the RE $\approx z_{\alpha/2}/\sqrt{n\theta}$ becomes unbounded. Therefore, to obtain precise estimates, we need very large n.

In most cases, the failure rates of the system components are very small by comparison with the transient-failure recovery rate. Let $\lambda_{max} > 0$ be the maximum component failure rate in the system. We call λ_{max} the 'rarity' parameter of the system. For Importance Sampling, Shahabuddin [27] notes that the elements of the modified transition matrix, $[\tilde{\pi}_{k',k}]$, should be independent of the rarity of the failures. This is an ideal Importance Sampling scheme which will always lead to the bounded RE property. He also proves in this case that RE will have the form

$$RE = \frac{z_{\alpha/2}}{\sqrt{n}} \frac{\sqrt{a_2 + o(1)}}{(a_0 + o(1))}$$

where a_2 and a_0 are positive constants depending on the implementation.

For a fixed number of samples, we can represent the behavior of RE as shown in Figure 3, for the cases of normal simulation and Importance Sampling. This ideal Importance Sampling scheme will be able to estimate the sample mean with a small, fixed number of samples. We also

observe that there is a certain region of reliability values beyond which the normal simulation will be better than the Importance Sampling.

In practice, techniques such as Balanced Failure Biasing approximate the behavior of this ideal case. They achieve variance reduction (and hence RE reduction) by pushing the system towards more faults and hence reducing the reliability of the underlying system. They then unbias the sample output to get the correct value of the sample mean.

The RE offered by such heuristics may not be constant but rather a complicated function of the bias value and the type of the system under consideration. However, as the reliability of the system decreases, the bias value has to be kept arbitrarily low in order to maintain the RE close to that of the normal simulation. Hence beyond a point, Importance Sampling is not very useful. It is important to experimentally determine this range of values so that the user can switch to the normal simulation instead of the Importance Sampling scheme.

In the systems that we are interested in simulating, the most obvious failure path is the one where the system is unable to meet its critical task deadlines. This obviously depends on a variety of factors such as the number of computing nodes available, average task load on the computing nodes, the recovery overheads etc. Consider a rather crude example: let a system be composed of x computing nodes and to meet its critical task deadlines it needs at least y of them. Increasing the failure bias has the effect of pushing the system towards complete breakdown (in other words, more of the nodes are pushed towards failure). In many cases this will be overkill and the likelihood ratio associated with these sample runs will be very low. Thus most of the sample runs will have a likelihood ratio that is very small and a few of them will have very large values. This effect causes the RE to blow up in cases where the bias value is very high.

Because of these conflicting effects associated with the bias value, each system might have a optimal bias value that pushes the system towards the failure path most of the time and still does not make the RE blow up. Since it is not practical for us to locate this optimal value for each configuration, it is enough if we are able to guess a bias value that gives good results over a range of systems.

4 Experimental Results

4.1 Importance Sampling Evaluation

In this section, we consider four distinct and representative system configurations and compare the system unreliability predicted by (a) normal (i.e., traditional) simulations (without any bias), and by (b) Importance Sampling. We show that Importance Sampling performs poorest when the system unreliability is high, i.e., when the events to be captured by simulation are not very rare, and best when the system is highly reliable. Our experiments suggest that if the unreliability of the system is of the order of 0.01 or higher, Importance Sampling may give

Parameter	Configuration 1	Configuration 2	Configuration 3	Configuration 4
Number of nodes	6	7	8	8
Network interconnect	Token Ring	FDDI	Rectangular mesh	3D hypercube
Transient processor failure rate (per hour)	5	2	1	1
Permanent processor failure rate (per hour)	1	0.2	0.1	0.1
Mission time (seconds)	1000	1500	2000	2000
Average node utilization	0.4	0.3	0.25	0.25

Table 1: System Configurations

Configuration	Mean		Variance		No. of Samples		HW_90		AF
	NS	IS	NS	IS	NS	IS	NS	IS	
1	4.24E − 02	2.36E − 02	4.06E − 02	5.20E − 02	2,500	2,600	12.2%	24.3%	undefined
2	3.20E − 03	3.42E − 03	3.19E − 03	9.90E − 04	5,000	1,400	32.0%	31.6%	3.57
3	7.00E − 04	6.46E − 04	6.99E − 04	5.65E − 05	20,000	1,938	34.3%	33.9%	10.32
4	3.00E − 04	2.98E − 04	2.99E − 04	1.74E − 05	40,000	2,430	37.9%	36.4%	16.46

NS=Normal simulation; IS=Importance Sampling;
HW_90= half width of confidence interval as percentage of mean

$$\text{Accleration Factor, } AF = \frac{\text{Number of samples under normal simulation}}{\text{Number of samples under Importance Sampling}}$$

Table 2: Normal simulation vs. Importance Sampling with a bias of 0.3

misleading results. On the other hand, when it is of the order of 10^{-3} or lower, the results of Importance Sampling agree very well with those of traditional simulation.

The four system configurations are shown in Table 1. The output of each simulation was a single number: 0 if the system was still functional at the end of the mission duration, and 1 if it failed before the specified mission duration ended. The simulations were repeated until the width of the 90% confidence interval was about 30% of the sample mean. The results are shown in Table 2. Apart from the observations we have already made regarding the accuracy of the sample mean, the difference in the required number of samples (which translates into the number of simulation runs needed) is striking when the unreliability is of the order of 10^{-3} or lower. Indeed, the acceleration factor[1] (defined in Table 2) varies from 3.57 for configuration 2 to over 16 for configuration 4. As the system reliability increases, so too does the acceleration factor.

[1] AF is undefined for Configuration 1 because Importance Sampling fails to provide an accurate result.

Configuration	Failure Probability	Variance when Bias Value =				
		0.2	0.3	0.4	0.5	0.6
1	4.24E − 02	4.80E−02	5.20E − 02	5.50E−02	5.86E−02	-
2	3.20E − 03	9.40E−04	9.90E−04	1.02E−03	1.37E−03	-
3	7.00E − 04	6.04E−05	5.65E−05	6.21E−05	6.27E−05	7.83E−05
4	3.00E − 04	3.20E−05	1.74E−05	1.69E−05	1.83E−05	2.92E−05
5	2.10E−05	4.70E−07	2.30E−07	1.90E−07	1.98E−07	2.53E−07
6	5.02E−07	-	3.07E − 09	2.93E − 09	2.64E−09	2.86E−09

Table 3: Sample variance for different failure bias values

4.2 Selecting the Bias Parameter

As mentioned earlier, failure bias is an important parameter that alters the dynamics of the sample output. If it is too low, we don't push the system towards additional component failures fast enough. Because of this only a small percentage of the sample runs result in a system failure and the sample variance is high. If the failure bias is too high, the distortion that this causes affects the accuracy of the simulation output, and the sample variance is high. There is usually some optimal value of the failure bias that pushes the system towards additional faults but is not too high to result in inaccuracies.

Since the simulator has to work with a large variety of systems with varying unreliabilities, we want to identify nominal values of failure bias that result in a low sample variance for system configurations with different ranges of unreliability values. Intuitively, if the sample variance is low, the estimates converge faster and we need fewer samples to get the desired confidence intervals.

We expect the user to choose some nominal bias depending on the 'guessed' range of system unreliabilities. This will serve as a starting point and the failure bias can be tuned if repeated sample runs or experiments are needed.

We want to use the above mentioned system configurations (with varying unreliability values) and observe how the sample variance changes when we vary the failure bias values. We add two more system configurations (with decreasing unreliabilities) to extend the range of the unreliabilities. For this experiment, we vary the failure bias over a range from 0.2 to 0.6 in steps of 0.1 and observe how the sample variance changes in response. The results are tabulated in Table 3.

From Table 3, we observe that the optimal bias value (the one that produces the least sample variance) is different for each configuration and in general, the lower the unreliability of the system, the higher the value of this optimal bias. This is as expected.

For Configuration 1, the normal simulation is the best choice. For all the failure bias values,

Configuration	Fixed Recovery Action	RAMP Algorithm
2	3.42E−03	3.36E−03
3	6.46E−04	2.84E−04
4	2.98E−04	1.74E−04
5	2.10E−05	1.30E−05

Table 4: Comparison of Recovery Policies

the sample variance is substantially above that for the normal simulation. Configuration 2 seems to have an optimal value around 0.2, Configuration 3 around 0.3, both Configurations 4 and 5 an optimal value around 0.4, and Configuration 6 around 0.5. It seems to stabilize around 0.5 for configurations with higher reliabilities.

A bias value of around 0.4 seems to work well for most of the configurations. If the guessed unreliability of the system is quite high (on the order of 10^{-3}), we are better off by choosing a bias of around $0.2 - 0.3$.

4.3 Comparing Recovery Policies

Here, we present a small example of the use of Importance Sampling in selecting suitable recovery policies in a real-time system.

When a fault is detected on a node, the system has a choice of three basic recovery actions. They are Retry (Restart execution on the same node using the last checkpoint), Replace (Replace the faulty node with a spare, if one exists) and Disconnect (Distribute the tasks that were running on the faulty node among the other active nodes).

RAMP is a dynamic resource management algorithm [31] that suggests the optimal recovery action to be used whenever there is a fault in the system and a decision has to be made regarding the choice of the recovery actions. We want to compare the reliability of a system using RAMP against another that uses a fixed recovery policy. To compare the two, we use simulation, accelerated by importance sampling.

An intuitive fixed recovery action can be formulated such as the following:

- When the node fails, try a Retry first.
- If the Retry failed, then try to Replace the faulty node by a spare node.
- If a spare does not exist, then as a final resort, Disconnect the faulty node and distribute its load to the other active nodes.

We used such a fixed recovery action in Configurations 2 to 5.

Table 4 also contains the unreliability of the system when the RAMP recovery policy is used. From this we can quantify the extent to which the RAMP algorithm outperforms the intuitive fixed recovery action for all of the considered configurations. In cases like these, using normal simulation would have taken a prohibitively long time to yield the same result.

5 Conclusion

This paper has discussed the implementation of an efficient variance reduction technique called Importance Sampling in a simulator testbed.

Importance Sampling was successfully implemented on the RAPIDS testbed. It was validated by running a series of simulations for different configurations and comparing the results with that of a normal simulation.

We observed the behavior of the scheme and its performance (reduction in sample variance) by varying the failure bias. Increasing the failure bias causes the sample variance to reduce faster but only upto a limit. Beyond this limit (which is specific to the system under observation) an increase in failure bias causes unstability. This knowledge is used to provide some guidelines in choosing a good failure bias probability for a given system.

Our experiments indicate that Importance Sampling is a powerful mechanism to accelerate the simulation of highly reliable real-time systems. It must, however, not be used for less reliable systems.

References

[1] M. Allalouf, J. Chang, G. Durairaj, V.R. Lakamraju, O.S. Unsal, I. Koren and C.M. Krishna, "RAPIDS: A Simulator Testbed for Fault-Tolerant Real-Time Systems," Proc. of HPC'98, *Grand Challenges in Computer Simulation*, pp. 191-196, Boston, April 1998.

[2] S. Andradottir, D. Heyman and T. Ott, "On the Choice of Alternative Measures in Importance Sampling with Markov Chains," *Operations Research* vol.43, no.3, pp.509-519 1995.

[3] M. Berg and I. Koren, "On Switching Policies for Modular Fault-Tolerant Computing Systems," *IEEE Trans. Computers,* Vol. C-36, pp. 1052-1062, Sept. 1987.

[4] M. Boyd and S. Bavuso, "Simulation Modeling for Long Duration Spacecraft Control Systems," *1993 Proc. Annual Reliability and Maintainability Symposium,"* pp 106-113 1993.

[5] J. Carrasco, "Failure distance based simulation of repairable fault-tolerant systems," *Proc. of 5th International Conf. on Modeling Techniques and Tools for Computer Performance Evaluation,* pp 337-351.

[6] J. Carrasco, "Efficient Transient Simulation of Failure/Repair Markovian Models," *Proc. of 10th Symposium on Reliable and Distributed Computing,* IEEE Computer Society Press, pp 152-161.

[7] P. L'Ecuyer, "Efficiency Improvement and Variance Reduction," *Proc. of the 1994 Winter Simulation Conf.* pp. 122-132 1994.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, *PVM: Parallel Virtual Machine,* MIT Press, 1994.

[9] P. Glynn and D. Iglehart, "Importance Sampling for Stochastic Simulations," *Management Science*, vol. 35, no. 11, pp. 1367-1393, 1989.

[10] A. Goyal and S.S. Lavenberg, "Modeling and analysis of computer system availability," *IBM J. Res. Develop.,* vol.31 pp.651-664, 1987.

[11] A. Goyal, P. Heidelberger, P. Shahabuddin, "Measure Specific Dynamic Importance Sampling for Availability Simulations," *1987 Winter Simulation Conference Proceedings,* IEEE Press 1987.

[12] A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola and P.W. Glynn, "A Unified Framework for Simulating Markovian Models of Highly Dependable Systems," *IEEE Transactions on Computers,* vol.41 no.1 pp. 36-51, 1992.

[13] J.M. Hammersley and D.C. Handscomb, *Monte Carlo Methods*, Meuthen, London, 1964.

[14] P. Heidelberger, "Fast Simulation of Rare Events in Queueing and Reliability Models," *ACM Transactions on Modeling and Computer Simulation* Vol. 5, No. 1, 1995.

[15] R. Jain, *FDDI Handbook,* Addison-Wesley, 1994.

[16] C. M. Krishna and K. G. Shin, "Performance Measures for Multiprocessor Controllers," *Performance '83*, May 1983.

[17] C.M. Krishna and K.G. Shin, *Real-Time Systems,* McGraw-Hill, 1997.

[18] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM,* Volume 21, 7, 1978.

[19] E.E. Lewis and F. Bohm, "Monte Carlo simulation of Markov unreliability models," *Nuclear Engineering and Design*, Vol. 77, 1984.

[20] M. Nakayama, "A Characterization of the simple failure biasing method for simulations of highly reliable Markovian Systems," *ACM Trans. Model. Comput. Simul.* vol. 4, no. 1, pp 52-88, 1994.

[21] M.L. Puterman, *Markov Decision Processes,* John Wiley & Sons Inc., 1994.

[22] S.M. Ross, *Applied Probability Models with Optimization Applications,* San Fransisco: Holden-Day, 1970.

[23] S.M. Ross, *Simulation,* Academic Press, 1997.

[24] P. Shahabuddin, V. Nicola, P. Heidelberger, A. Goyal and P. Glynn, "Variance Reduction in Mean Time to Failure Simulations," *1988 Winter Simulation Conference Proceedings,* IEEE Press, 1988.

[25] P. Shahabuddin, "Simulation and Analysis of Highly Reliable Systems," Ph.D. Thesis, Department of Operations Research, Stanford University, Palo Alto, California.

[26] P. Shahabuddin and M. Nakayama "Estimation of reliability and its derivatives for large time horizons in Markovian systems", *1993 Winter Simulation Conference Proceedings,* IEEE Press, pp 491-499.

[27] P. Shahabuddin, "Simulation of Highly Reliable Markovian Systems," *Management Science,* vol. 40, pp 333-352, 1994.

[28] W. Stallings, *Handbook of Computer-Communications Standards,* Howard W. Sams & Co., 1988.

[29] J.S. Steinman, "Breathing Time Warp," *Proceedings of the 1993 Workshop on Parallel and Distributed Simulation,* 1993.

[30] K.K. Toutireddy, "A Testbed for Fault Tolerant Real-Time Systems," *M.S. Thesis,* Univ. of Mass. Amherst, 1996.

[31] K. Yu, "RAMP and the Dynamic Recovery and Reconfiguration of a Distributed Real-Time System," *Ph.D. Thesis,* Univ. of Mass. Amherst, 1996.

[32] K. Yu and I. Koren, "Reliability Enhancement of Real-Time Multiprocessor Systems through Dynamic Reconfiguration," *Fault-Tolerant Parallel and Distributed Systems,* D. Pradhan and D. Avresky (Editors), pp. 161-168, IEEE Computer Society Press, Los Alamitos, CA, 1995.