22nd Euromicro Conference on Real-Time Systems

Efficient Stochastic Analysis of Real-Time Systems via Random Sampling

Khaled S. Refaat, Pierre-Emmanuel Hladik
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
Toulouse, France
{*krefaat, pehladik*}*@laas.fr*

***Abstract*—This paper provides a stochastic approach to the analysis of real-time systems under preemptive priority-driven scheduling. The main idea is to simplify the execution time distributions via random sampling to decrease complexity. This beneficial effect is counterbalanced by an increase in pessimism. However, the proposed analysis is significantly less pessimistic than the classical worst-case deterministic analysis. In addition, it could be tuned according to the memory and time availability. Thus, the proposed method provides, for the first time, a relation between pessimism and computational resources. The testing results show the effectiveness of the sampling approach in terms of practicality and optimism.**

***Keywords*-stochastic analysis; efficient; random sampling; real time systems; pessimism**

I. Introduction

A real-time system is a system that should satisfy some temporal constraints in order to work correctly. In soft real-time systems, the completion of an operation after its deadline decreases the quality of service, e.g. dropping frames when displaying a video. Therefore, a soft real-time system should be designed to finish its operation as fast as possible in order to provide an acceptable quality of service.

On the other hand, in a hard real-time system, the completion of an operation after its deadline is considered unacceptable and might lead to a catastrophic failure of the complete system. Furthermore, in some hard real-time systems, failing to meet a temporal constraint could result in life threatening situations. For example, an antilock braking system must detect and respond to loss of traction within a few milliseconds; a delay of one or two seconds would be intolerable and potentially deadly [1]. Therefore, these critical systems should be designed in a way that provides some guarantees.

Prediction of whether a task will satisfy a temporal constraint or not is necessary to design a useful real-time system. To conduct schedulability analysis of real-time systems, some assumptions are usually made. For example, classical approaches consider the worst-case execution time to characterize the execution time of a task. This assumption has been made in utilization factor analysis [2], [3] and response time analysis [4]. Although the computational cost under this assumption is small, the analysis is very pessimistic. Such worst-case execution time value is not frequently encountered during real execution and therefore basing the analysis on it leads to extremely pessimistic results giving rise to oversized real-time systems [5].

In order to tackle the problem of pessimism, it is possible to model the execution time of a task as a discrete random variable whose distribution could be obtained by hybrid techniques [6] or by measurements.

The problem is that the exact stochastic analysis of most real-time systems under preemptive priority driven scheduling is unaffordable in current practice [5]. To overcome this problem, some approaches perform stochastic analysis with a special scheduling model that isolates tasks so that each one could be analyzed independently [7], [8].

Moreover, some approaches introduce worst-case assumptions to simplify the stochastic analysis. Monolache *et al* proposed restricting preemption [9], whereas others introduced the critical instant assumption[10], [11], [12].

In [13], Díaz *et al* described a stochastic analysis method for general periodic real-time systems. The proposed method accurately computes the response time distribution of each task in the system, thus making it possible to determine the probability of missing the deadline for individual tasks. This method is not dedicated for a specific scheduling method, and can be used with both static and dynamic priority scheduling. However, this approach is intractable for large systems [13].

In such exact analysis, each task has an execution time distribution. Such distributions usually have a large number of execution time values. During the analysis, iterative procedures of convolution and other stochastic functions produce a huge number of values in computed backlog and response time distributions which makes the analysis impractical in terms of memory demand [5].

In [14], [5], Díaz *et al* extended their previous study by formalizing the concept of pessimism in the stochastic analysis. They showed how to compare between several distributions to produce a pessimistic analysis. Such introduction opened the door for safe approximations.

However, it is worth mentioning that any approximate analysis opens a new source of troublesome complexity. First, any approximate analysis should not be more optimistic than the exact analysis. In other words, the probability of missing the deadline computed from the approximate analysis should not be smaller than that computed from the exact one. Moreover, the pessimism resulting from the approximation should not be excessive so that the analysis could be useful. Furthermore, the time and memory demand of such approximate analysis should fit the embedded system or the used machine.

This paper uses the formal definition of pessimism. We propose a method for efficient stochastic analysis by simplifying the exact distributions through random sampling. Subsequently, the stochastic analysis is completed using the new pessimistic distributions.

During the sampling process, each distribution is replaced with another more pessimistic one through random sampling. The pessimism of the new distributions and therefore the safety are ensured by assigning the unsampled probability mass to the worst-case execution time. We show that the proposed analysis is safe according to the definition proposed in [5].

In addition to this, we show that the most efficient way to perform the sampling is to finish it before the analysis as opposed to the interleaving process of sampling and analysis.

Moreover, we propose a modification to our novel sampling analysis that has even decreased the pessimism of the approach with the same memory demand. However, such modification degrades the time efficiency and therefore should only be used if such degradation could be tolerated.

Simulations show how the proposed method provides, for the first time, a relation between pessimism and efficiency. Thus, it provides system designers with a capability to design their system according to their available resources.

While the previous methods suffer from either extreme pessimism or impracticability, our approach provides a relation between pessimism and computational resources. Actually, the amount of time and memory needed to complete the analysis are directly proportional to the number of samples used. In addition, the pessimism is inversely proportional to the number of samples. In fact, when all the distributions' values are sampled, the analysis becomes exact. The main gain of the sampling approach is that the number of samples could be increased according to the time and memory available for the analysis.

The paper is organized as follows. In section II, the system model is assumed and the notations used throughout the paper are given. Section III describes the stochastic analysis method, whereas in section IV, we propose the new efficient analysis via random sampling. We discuss the reason for sampling before analyzing in section IV-E. In Section V, the experimental results of our proposed analysis method are shown. Finally, in Section VI, we conclude.

II. System model and notations

The system consists of n independent periodic tasks $\{\tau_1, ..., \tau_i, ..., \tau_n\}$. A task τ_i is characterized by a set of parameters $\langle T_i, \Phi_i, e_i, D_i, M_i \rangle$. These parameters are the period, T_i, the initial phase, Φ_i, the execution time, e_i, the deadline or the temporal constraint, D_i, and the maximum allowed probability of missing the deadline, M_i.

The execution time, e_i, is a discrete random variable of a known distribution. The probability function (PF) is denoted by $f_{e_i}(\cdot)$, where $f_{e_i}(e) = P\{e_i = e\}$.

Each periodic task results in an infinite number of jobs. $\Gamma_{i,j}$ denotes the jth job of the task τ_i. Each job $\Gamma_{i,j}$ is released at a deterministic time, $\lambda_{i,j}$. The job release time, $\lambda_{i,j}$, is computed via the formula [5]:

$$\lambda_{i,j} = \phi_i + (j-1)T_i \tag{1}$$

The response time of a job $\Gamma_{i,j}$ is a discrete random variable denoted by $R_{i,j}$, whereas the response time of a task, denoted by R_i, is computed as the average of the response times of its jobs:

$$f_{R_i}(r) \overset{\text{def [13]}}{=} \frac{1}{m_i} \sum_{j=1}^{m_i} f_{R_{i,j}}(r) \tag{2}$$

where $m_i = T/T_i$, which is the number of jobs from τ_i released in a hyper-period of length T.

A task τ_i is said to be schedulable if $P\{R_i > D_i\} \leq M_i$.

III. Summary of previous stochastic analysis

Our approach is based on the stochastic analysis proposed by [13], [14], [5]. This Section summarizes this method. In order to have a thorough presentation, the reader can refer to [5].

For the sake of simplicity, the task to which a job belongs is not tracked, thus a job has a single index, for instance, Γ_j. The index of a job refers to its order in the infinite sequence of jobs, i.e. Γ_k is released before Γ_{k+1}, that is $\forall k, \lambda_k \leq \lambda_{k+1}$. The response time of a job Γ_j is computed as follows:

$$R_j = W(\lambda_j) + e_j + J_j \tag{3}$$

where:

- R_j denotes the response time distribution of an arbitrary job Γ_j,
- $W(\lambda_j)$ denotes the *backlog* at time λ_j, *i.e.* the sum of the remaining execution times of all the jobs that did not finish up to time λ_j while having higher priorities than the job under analysis.
- J_j denotes the interference of all higher priority jobs released after job Γ_j.

The backlog at the release time of any job Γ_j, denoted by $W(\lambda_j)$, can be computed using the following iterative procedure [5]:

$$W(\lambda_{k_0}) = 0 \tag{4}$$

$$W(\lambda_k) = shrink(W(\lambda_{k-1}) + e_{k-1}, \lambda_k - \lambda_{k-1}) \qquad (5)$$

Where λ_{k_0} denotes the release time of the first job released before Γ_j and has a higher priority. The shrink function is given by:

$$f_{shrink(W,\Delta)}(x) = \begin{cases} 0 & \text{if } x < 0, \\ \sum_{z=-\infty}^{0} f_W(z+\Delta) & \text{if } x = 0, \\ f_W(x+\Delta) & \text{if } x > 0. \end{cases} \qquad (6)$$

Iterations start with a zero backlog as in equation (4) and iterates on all higher priority jobs released before Γ_j.

After computing the backlog at the release time of Γ_j, the backlog distribution is convolved with the execution time distribution. Such convolution results in a partial response time which is valid only if no interference with subsequent higher priority jobs takes place. In case of the existence of higher priority jobs released after λ_j, this partial response time will be valid only from λ_j to λ_{j+1}. A validity range is indicated as a super index for the response time: $R^{[0,\lambda_{j+1}-\lambda_j]}$ which is computed as follows:

$$R^{[0,\lambda_{j+1}-\lambda_j]} = W(\lambda_j) + e_j \qquad (7)$$

In order to increase the range of validity, the following equation is used [5]:

$$R^{[0,\lambda_{k+1}-\lambda_j]} = AF(R^{[0,\lambda_k-\lambda_j]}, \lambda_k - \lambda_j, e_k), k > j \qquad (8)$$

where the job Γ_k has a higher priority than Γ_j, and AF is a stochastic function given by:

$$f_{AF(R,\Delta,e)}(x) = \begin{cases} f_R(x) & \text{if } x \le \Delta \\ \sum_{i=\Delta+1}^{\infty} f_R(i).f_e(x-i) & \text{if } x > \Delta \end{cases} \qquad (9)$$

Each iteration using equation (8) increases the interval of validity of the partial response time. The iterating is stopped when the deadline is included in the validity range. Consequently, the probability of missing the deadline for a certain job could be computed by summing the probabilities of the response time values lying before the deadline and subtracting this sum from one:

$$P(R_j > D_j) = 1 - \sum_{k=0}^{k=D_j} P(R_j^{[0,\Delta]} = k) \qquad (10)$$

After completing the analysis, the probability of missing the deadline of a certain task is computed by averaging the probabilities of missing the deadlines of all its jobs, see Eq. (2).

IV. Sampling analysis

A. Motivation

The exact stochastic analysis [13], [14], [5] has overcome the problem of pessimism that was introduced in the worst-case analysis. However, the computational requirements have rendered such exact approaches impractical except for simple scenarios [5].

In such exact stochastic analysis, during each iteration of the backlog computation, two random variables are convolved in equation (5) which results in a new random variable. Assuming that the numbers of values of the convolved distributions are n and m respectively, the number of values of the resultant distribution consists of more values than any of the convolved distributions. Moreover, it could reach nm in the worst-case. This could easily lead to a memory burst after several iterations.

To overcome this problem, we propose simplifying the execution time distributions before starting the analysis. We claim that such simplification will necessarily render the analysis practical while being far away from the extreme pessimism of the worst-case analysis. However, such approximation should be done carefully to ensure the safety of the analysis.

B. Introduction to sampling

In statistics, sampling is the process of selecting individual units or samples intended to yield some knowledge about a population of concern [15].

Probability sampling is the process of sampling while taking in consideration the probability of occurrence of each unit. In other words, a unit that has a high probability of occurrence will have a higher chance to be sampled than a unit with a lower probability of occurrence. Probability sampling is usually used when the probability of occurrence can be accurately determined [15].

As a special case, simple random sampling is a sampling design in which a distinct units are selected from the A units in the population in such a way that every possible combination of a units is equally likely to be the sample selected. In simple random sampling, the probability that the i^{th} unit of the population is included in the sample is $\pi_i = a/A$, so that the inclusion probability is the same for each unit. Other sampling methods include systematic sampling and stratified sampling [15].

To illustrate the benefit of sampling, consider the following real world example. In order to estimate the average weight of hens in the country, a sample hen can be taken from each farm to measure its weight. To do such sampling, a random number generated from a uniform distribution between 0 and 1 can be allocated to each hen, and the hen with the highest number in each farm is selected. Subsequently, each representative hen weight is multiplied by the number of hens in its farm. Finally, all such products

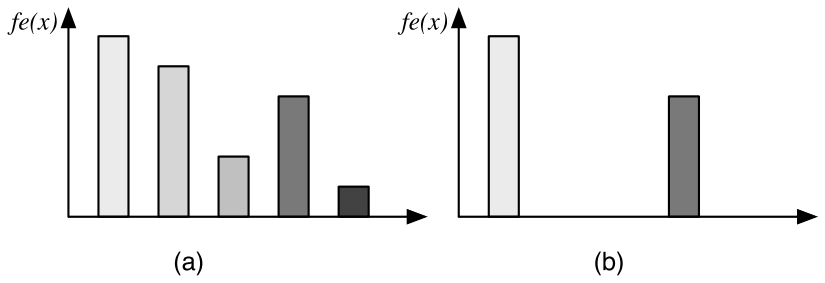

Figure 1. An exact execution time distribution and two samples from it

are summed and such sum is divided by the total number of hens in the country to get the estimated average. In this case, the sampled hen was used to represent all the hen population in its farm and that has essentially saved the time for weighing each hen in the nation which would be problematic.

C. The proposed sampling approach

In the exact stochastic analysis approach, each execution time of a task is represented by a probability distribution. We propose simplifying each execution time distribution by selecting k samples from the N values of the distribution using the probability sampling method. These new distributions are used to perform the stochastic analysis as it was shown in section III.

However, the new analysis must not be more optimistic than the exact one. An optimistic analysis might lead to an inconsistent verification process. An unschedulable task could be erroneously declared schedulable by an optimistic analysis. Therefore, we have to ensure that our new analysis is not more optimistic than the exact one.

Figure 1 shows an execution time distribution (Fig. 1(a)) and its simplified version (Fig. 1(b)) with $k = 2$, generated by random sampling, respectively. It is clear that the samples shown in figure 1(b) do not form a complete distribution that sums to one.

The distribution resulting from sampling is completed by assigning the rest of the probability mass to the worst-case execution time as shown in figure 2. It is important to prove that the analysis using the new distribution will be safe.

Each simplified distribution e'_j extracted from the original distribution of e_j takes the following mathematical form:

$$f_{e'_j}(x) = \sum_{i=1}^{k} f_{e_j}(x_i)\delta(x-x_i) + (1-\sum_{i=1}^{k} f_{e_j}(x_i))\delta(x-x_w) \tag{11}$$

where x_i is the execution time value of the i^{th} sample, x_w is the worst-case execution time value in the original distribution, f_{e_j} is the original execution time probability

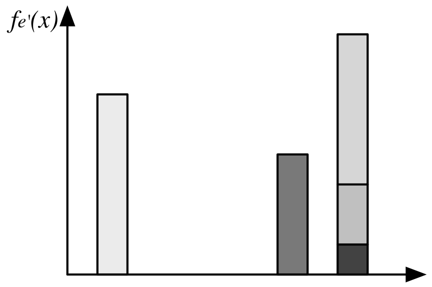

Figure 2. The final simplified distribution after compensating the probability deficit

distribution function, and δ is a function defined by:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \tag{12}$$

The first term in equation (11) represents the contribution of the k samples, whereas the second term is added to ensure pessimism by assigning the rest of the probability mass to the worst-case execution time.

1) Proof of safety: Diaz *et al* [5] have defined the "greater than" relation between two distributions (see Definition 1 in [5]). The definition is hereby summarized as follows: for two random variables X and X' if $P(X' \leq D) \leq P(X \leq D)$ for any D, then X' is said to be greater or more pessimistic than X which is denoted by $X' \succ X$. They showed that using more pessimistic execution time distributions during analysis will result in a pessimistic response time distribution and, therefore, will be a safe approximation.

Theorem 1. *The new distribution after sampling in accordance with equation* (11) *is greater than the original distribution.*

Proof: In our proposed simplification, since a portion of the probability mass was moved to a larger execution time value, the new random variable X' will have a cumulative distribution function that is always less than or equal to that of the original random variable. Therefore, the new distribution will be greater than ($\succ$) the original one and, therefore, our subsequent analysis will be safe. ■

2) Parameter tuning: The number of samples, k, is a parameter that should be tuned according to the time and memory available. k ranges from 1 to the total number of units available, N. When $k = N$, the analysis becomes exact. On the other hand, when $k = 1$ the analysis approaches the worst-case analysis which suffers from exaggerated pessimism. Therefore, k has to be as large as possible given that the memory and time available are sufficient to complete the analysis using that k.

In this paper, we show how tuning the number of samples could be used to provide different combinations of resource requirements and analysis pessimism.

In order to determine the suitable k, a system designer is supposed to collect a large number of different scenarios and determine the highest k that does not lead to any memory or time deficiency using the following simple algorithm:

1) Prepare a batch of different scenarios, s.
2) Initialize k to 1.
3) Analyze s using sampling analysis with k.
4) If analysis was successful $k = k + 1$ and go to step 3 else return $k - 1$

In step 4, a successful analysis is an analysis that is completed without requiring more memory than that available on the hardware that will be used. The system designer could also add, to such definition of success, a time efficiency requirement. In that case, an analysis whose execution time exceeds the maximum allowed time without causing any memory problems will not be considered successful.

D. Implementation

In order to generate a sample from a random variable X with the following probability mass function:

$$P\{X = x_j\} = p_j \tag{13}$$

where $j = 0, 1, ..., n$, and $\sum_j p_j = 1$, a random variable U, that is uniformly distributed over $(0, 1)$, is generated. After that, X is sampled as follows:

$$X = \begin{cases} x_0 & \text{if } U < p_0 \\ x_1 & \text{if } p_0 \leq U < p_0 + p_1 \\ \vdots & \\ x_j & \text{if } \sum_{i=0}^{j-1} p_i \leq U < \sum_{i=0}^{j} p_i \\ \vdots & \end{cases} \tag{14}$$

This sampling procedure is repeated k times for each execution time distribution.

We note here that repeated samples should not exist because this might lead to erroneous optimistic response time distributions. To see what could go wrong if a repeated sample was taken, consider the following execution time distribution: $\{(1, 0.2); (2, 0.2); (3, 0.5); (4, 0.1)\}$, where for each pair, the first value is the execution time of the job and the second is its probability. If repeated samples are allowed, a simplified distribution where $k = 2$ might be: $\{(1, 0.4); (4, 0.6)\}$ because $(1, 0.2)$ pair was sampled twice. The resultant simplified distribution is no longer greater than ($\succ$) the original distribution and, therefore, the analysis will be unsafe. Therefore, it is of utmost importance to make sure that the samples are all different. In this approach, sampling is repeated till k distinct samples are produced while neglecting any repetition.

E. Discussion

There are two options of sampling:

1) Sampling only one value-probability pair from each execution time distribution, completing the analysis, and then repeating the whole process for k times.
2) Sampling k value-probability pairs from each execution time distribution and then completing the analysis at once.

The latter option is more efficient since the former one passes through the whole analysis process each time and it might be repeating previous computations.

To illustrate such defect, consider a simple task set giving rise to three jobs that have the same priority and the same execution time distribution: $\{(1, 0.5); (2, 0.4); (3, 0.1)\}$, while being started at times 1, 2, and 3 respectively.

Using the former approach the analysis could have been completed, in the first step, using the following samples: $(1, 0.5)$ for the first job, $(1, 0.5)$ for the second job, and $(2, 0.4)$ for the third job. When the process is repeated using new samples in the second step, the samples could be $(1, 0.5)$, $(1, 0.5)$ and $(1, 0.5)$. This means that, to compute the backlog for the last job, adding the first two pairs will be repeated twice and it is inefficient not to have saved the result during the first step.

For such complexity, sampling k times in advance is more efficient due to the inherited dynamic programming method in the reuse of convolved distributions for outputting new convolved distributions.

Using the same example while sampling in advance and setting k to 1, the new simplified distributions after sampling (ordered by job's position) could be $\{(1, 0.5); (3, 0.5)\}$, $\{(1, 0.5); (3, 0.5)\}$, and $\{(2, 0.4); (3, 0.6)\}$. In this case, to compute the backlog for the last job, the first two distributions are convolved to get the following distribution: $\{(2, 0.25); (4, 0.5); (6, 0.25)\}$. This new distribution will be used to get convolved with the last job's simplified execution time distribution. We note here that no repeated computations occurred. Therefore, sampling k times in advance is more efficient.

To better understand the difference between the two options, figure 3 shows an example of a process of iteratively computing the backlog, where a node represents a value-probability pair and an edge $e\{n_i, n_j\}$ represents a single

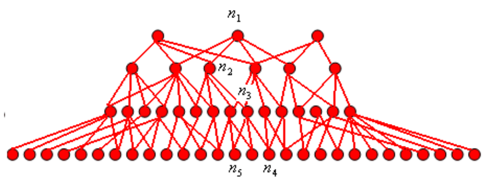

Figure 3. Graph showing the computations and involved value-probability pairs while iteratively computing the backlog

computation where n_i is used to compute n_j. The first level represents the initial backlog distribution, whereas the second level represents the new backlog distribution in the second iteration, and so on. Each edge from one node i at a certain level to another j in the lower level shows that node i was used in a computation to contribute to the value of node j.

When the sample is done in advance, each edge will be visited once because after reaching a certain level during the backlog computation iterations, the process never goes back. Instead, the current backlog computed (in the level) is used to be convolved with a new execution time distribution as it was shown in section III.

However, while repeating the analysis k times using single pairs, it is possible that $e\{n_1, n_2\}, e\{n_2, n_3\}$, and $e\{n_3, n_4\}$ are visited during the first step. After that, $e\{n_1, n_2\}, e\{n_2, n_3\}$, and $e\{n_3, n_5\}$ can be visited in the second step. Thus, $e\{n_1, n_2\}, e\{n_2, n_3\}$ are visited twice. This shows that sampling k times in advance inherently avoids repeated calculations as opposed to repeating the analysis k times with single values.

F. Modified sampling

The high probable low execution time values are the most effective for a good approximation. The philosophy behind favoring low valued execution times is that such values will probably contribute to the probability mass lying prior to the deadline in the final computed response time because they result in low response time values.

It is clear from eq. (10) that as the probability mass lying before the deadline increases, the probability of missing the deadline decreases. Decreasing the probability of missing the deadline essentially decreases the pessimism of the approximation.

Fortunately, sampling randomly from the exact distributions inherently favors high probable values. However, it does not differentiate between low and high valued execution times. We modify this approach to sampling from a generated distribution that puts some emphasis on low valued execution times.

In order to decrease the pessimism, we have to work on increasing the values lying before the deadline. Thus, low execution time values are vital to decreasing the probability of missing the deadline and therefore some emphasis is put on them in this modified sampling procedure. The sampling stage is modified by sampling from a modified distribution that favors lower values. Each probability is modified by the following equation:

$$p = p \div v^{\Psi} \tag{15}$$

where v denotes the value possessing this probability, p. Ψ denotes a positive favoring constant that is tuned to determine the degree of favoring lower values. The bigger the value of Ψ, the more low execution time values are favored. In fact, when Ψ approaches zero, the modified sampling turns out to be the premier sampling approach. The modified probabilities are normalized in order to form a valid probability distribution.

After modifying the distribution, the same method of sampling based on equation (14) is used. We note here that it is of utmost importance to replace the sampled probability with the corresponding original one before starting the analysis in order to be analyzing the original system.

We noticed that in some distributions there are two or three samples constituting almost the whole probability mass. Therefore, insisting on getting k distinct samples might lead to a significant delay because these high probable values will be always sampled. In the modified sampling approach, we decided to sample k times and discard any repetition. This avoids the re-sampling that might cause an unfortunate delay.

V. Simulation

A. Task set generation

In order to conduct experimentation, task sets are generated. For each task, a uniformly distributed random variable from 0 to 4 is generated and another one from 0 to 1 to set Φ_i and M_i respectively. Moreover, another uniformly distributed random variable from 1 to i is generated, to set the priority of the task, where i is the number of the generated task for the current generated task set. The worst-case execution time, E_{max}, is generated uniformly to be from 0 to 5, whereas the period is set to be exceeding the worst-case execution time by $\aleph$, where $\aleph$ is uniformly distributed from 0 to 7. The deadline for the task is set using the following equation: $D_i = T_i + \epsilon$, where ϵ is computed by: $\epsilon = 10 \times U - 5$, where U is a uniform random variable from 0 to 1.

In order to generate the execution time distribution of the task, a uniformly distributed random variable from 0 to 10 is generated and $\Re$ is set to it, where $\Re$ is the number of significant units in the distribution. Each value-probability pair for the $\Re$ units is computed by generating two uniformly distributed random variables from 0 to E_{max} and from 0 to 1 for the execution time value and its associated probability respectively.

In order to avoid probabilities being culminated in one or two pairs, we normalize the generated probabilities by $\Re$. Subsequently, new value-probability pairs are added using the same methodology but by normalizing probabilities using an arbitrary large number to simulate the execution time values that rarely occurs. New value-probability pairs are generated till the summation of all probabilities reaches unity.

The same method is used to add new tasks as long as the added task does not cause the utilization to be greater than one. If the new generated task will cause such increase, the task set generation process is stopped and the generated task set is saved. The whole process is repeated to generate several task sets to be used either for simulation or parameter tuning.

B. *Quantity of pessimism*

In order to compare the pessimism of all models, we average the probability of missing the deadline of all the tasks in each task set for each model. We name this quantity the pessimism of the approach (ρ):

$$\rho = \sum_{i=1}^{N} \frac{P\{R_i > D_i\}}{N}$$

where N denotes the number of tasks in the task set.

C. *Results*

We have generated 100 task sets randomly. The machine used for simulation has a 1.60GHz processor and 224 MB of RAM. Sixty nine of the generated cases were simple enough not to cause a memory burst for the exact stochastic analysis. Therefore such diminished set was used to compare between the worst-case analysis, the stochastic analysis, and the proposed sampling analysis whether while favoring low execution time values or not.

The average hyper-period for the diminished set was about 77.36, whereas the maximum hyper-period was 400. For the number of values in an execution time distribution, the average was about 7.34, and the maximum was 10. The maximum number of tasks, and jobs were 2, and 17 respectively, whereas the average number of jobs was about 5.5. The deadline was generated randomly but restricted to be greater or smaller than the task period by no more than 5 time units. The utilization of all task sets was ensured to be smaller than one.

Figure 4 shows sorted pessimism curves for all approaches. The x axis denotes the task set number, whereas the y axis denotes the pessimism quantity.

One can see that the worst-case analysis is the most pessimistic. On the other hand, the complete stochastic analysis exact curve provides a lower bound. Any analysis that is more optimistic than the exact curve is not safe. Thus, we are looking forward to getting curves nearer to the exact curve without being more optimistic than it.

For the sampling analysis, 4 samples were used and the favoring constant, Ψ, was, by trial and error, set to 0.4. The favoring constant was determined by trying several real numbers not exceeding one, and selecting the value resulting in less pessimistic results. If Ψ is set to a large value, low execution time values dominate, and high probable ones might not be normally favored. Therefore, searching for a suitable value for Ψ was limited to values not exceeding one. In our future work, we are planning to consider a more efficient strategy for determining Ψ.

As shown in figure 4, the sampling analysis curves are by far more optimistic than the worst-case analysis while being practically feasible in terms of memory as presented later. In addition, favoring low execution time values achieves a mild enhancement since it decreased pessimism.

For the time comparison, figure 5 shows the sorted time curves for all approaches. The x axis denotes the task set number, whereas the y axis denotes the time taken to finish the analysis in milliseconds (t).

In simple cases, all curves are comparable. However, in more complex cases, the worst-case execution time is the most efficient. Moreover, the complete stochastic analysis is the least efficient.

Both sampling methods were significantly more efficient than the exact analysis. In addition, the modified sampling approach was less efficient than the sampling approach. However, in some case, the modified sampling was more efficient.

The reason for this is that, in the modified sampling, the sampling method is not repeated if a repeated value was sampled. However, in the normal sampling method, sampling is repeated till k different values are sampled. This has caused the modified sampling to be more efficient in cases where only two or three values constitute a large portion of the probability mass.

In order to study the effect of changing the number of samples on pessimism, figures 6 and 7 show the sorted pessimism curves while using different number of samples using the sampling and the modified sampling methods respectively. One can see here that as the number of samples increases, the pessimism curve becomes less pessimistic.

However, sometimes the pessimism quantity, in figures 6 and 7, appears to be non-monotonic with respect to the number of samples. This is because of the randomness of the sampling process. Consider for example the following distribution: $\{(1,0.4);(2,0.2);(3,0.2);(4,0.2)\}$. A simplified distribution with $k=1$, in a case where the first pair was sampled, could be $\{(1,0.4);(4,0.6)\}$. Another simplified one with $k=2$, in a case where the second and the third pairs were sampled, could be $\{(2,0.2),(3,0.2),(4,0.6)\}$. One can see here that we were occasionally lucky in the case of $k=1$ and it happened that the resultant distribution was not more pessimistic ($\succ$) than that created with $k=2$. Additionally, in some execution time distributions, a single

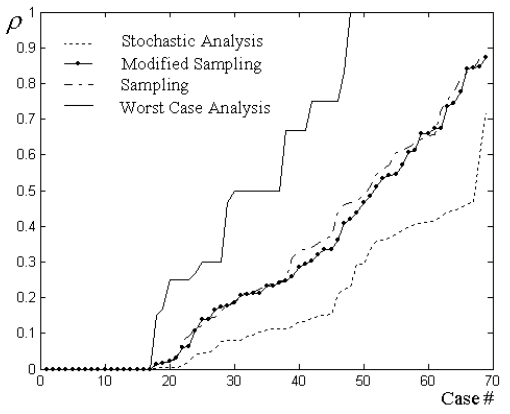

Figure 4. Pessimism curves

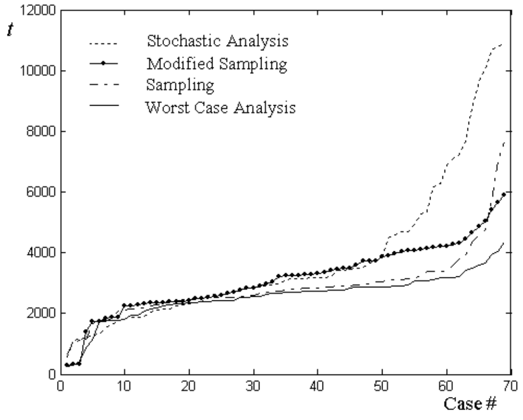

Figure 5. Time curves

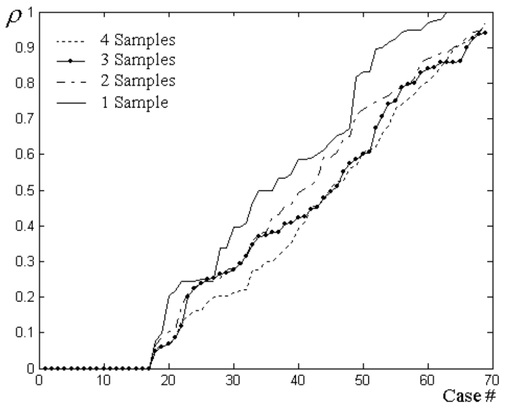

Figure 6. Pessimism curves using the sampling method for different number of samples

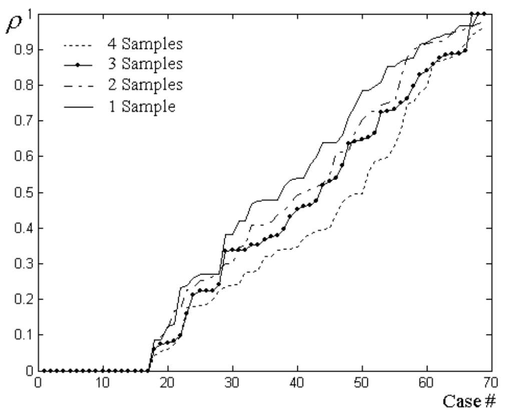

Figure 7. Pessimism curves using the modified sampling method for different number of samples

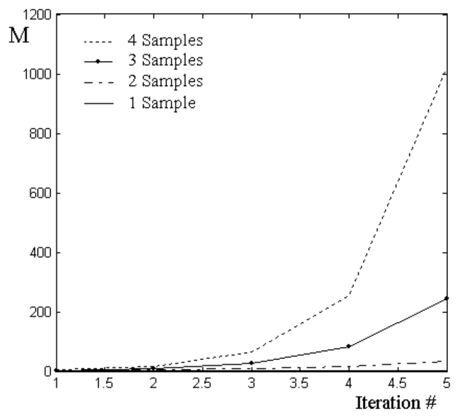

Figure 8. Worst-case backlog computation memory demand for different number of samples

execution time value possesses almost the whole probability mass. Therefore, insisting on getting k distinct samples might cause an unfortunate delay that increases with k.

In addition, Figure 8 shows the worst-case memory demand, M, while computing the backlog iteratively using different number of samples. The x axis denotes the iteration number, whereas the y axis denotes the memory demand in 8 bytes (assuming that each value is stored in 8 bytes). For example, when the number of samples is 4, the first convolution will result in 16 values in the worst-case. The second convolution will result in, at most, 16×4 values and so on. The curve showing the memory demand for the one-sample case is almost approaching zero and therefore not clear from the figure. It is clear that the memory demand rate of increase decreases as the number of samples decreases. The analysis times with different number of samples were comparable and therefore not shown.

Due to the fact that the number of samples affect the memory demand and the degree of pessimism, it provides a relation between computational resources and pessimism.

VI. Conclusion

In this paper, we have introduced a novel analysis technique that provides a relation between pessimism and computational resources. By changing the number of samples, one could increase or decrease the level of pessimism; and, accordingly, the time, and memory needed to complete the analysis. The proposed method is, by far, more optimistic than the worst-case analysis approach.

In addition to this, it is practical in terms of memory, and time since one could adjust it according to the memory, and time availability. Moreover, we have introduced another variant of our sampling approach that favors low execution time values. Favoring such values have caused a mild decrease in pessimism while fixing the number of samples used, and, accordingly, the memory demand. In the modified sampling method, we discarded any repeated values as apposed to insisting on getting k distinct samples by re-sampling. This has avoided an occasional increase in the analysis time.

References

[1] B. Lewis, *Fundamentals of Embedded Software*. Prentice Hall, 2004, ch. 1.

[2] L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal ACM*, vol. 20, no. 1, 1972.

[3] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *11th IEEE real-time systems symposium*, 1990.

[4] K. Tindell, A. Burns, and A. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, vol. 6, 1994.

[5] J. López, J. Díaz, J. Entrialgo, and D. García, "Stochastic analysis of real-time systems under preemptive priority-driven scheduling," *Real-Time Systems*, vol. 40, no. 2, 2008.

[6] G. Bernat, A. Colin, and S. Petters, "Wcet analysis of probabilistic hard real-time systems," in *the 23rd IEEE Real-Time Systems Symposium*, 2002.

[7] A. Atlas and A. Bestavros, "Statistical rate monotonic scheduling," in *19th IEEE real-time systems symposium*, 1998.

[8] L. Abeni and G. Buttazzo, "Stochastic analysis of a reservation based system," in *9th int workshop on parallel and distributed real-time systems*, 2001.

[9] S. Manolache, P. Eles, and Z. Peng, *Real-time applications with stochastic task execution times analysis and optimisation*. Springer, 2007.

[10] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. W, and J.-S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *the real-time technology and applications symposium*, 1995.

[11] M. Gardner, "Probabilistic analysis and scheduling of critical soft real-time systems," Ph.D. dissertation, University of Illinois, Urbana-Champaign, 1999.

[12] M. Gardner and J. Liu, "Analyzing stochastic fixed-priority real-time systems," in *the 5th international conference on tools and algorithms for the construction and analysis of systems*, 1999.

[13] J. Díaz, D. García, K. Kim, C.-G. Lee, L. L. Bello, J. M. López, S. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *the 23rd IEEE Real-Time Systems Symposium*, 2002.

[14] J. Díaz, J. M. López, M. García, A. M. Campos, K. Kim, and L. L. Bello, "Pessimism in the stochastic analysis of real-time systems: Concept and applications," in *the IEEE Real-Time Systems Symposium*, 2004.

[15] W. A. Fuller, *Sampling Statistics*. Wiley, 2009, ch. 1.