A Statistical Response-Time Analysis of Real-Time Embedded Systems

Yue Lu[1,3], Thomas Nolte[1]
[1]Mälardalen Real-Time Research Centre
Mälardalen University, Västerås, Sweden
{yue.lu, thomas.nolte}@mdh.se

Iain Bate[1,2]
[2]Department of Computer Science
University of York, YO10 5GH, York, UK
iain.bate@cs.york.ac.uk

Liliana Cucu-Grosjean[3]
[3]INRIA Nancy-Grand Est
Nancy, France
liliana.cucu@inria.fr

***Abstract*—Real-time embedded systems are becoming ever more complex. We are reaching the stage where even if static Response-Time Analysis (RTA) was feasible from a cost and technical perspective, the results of such an analysis are overly pessimistic. This makes them less useful to the practitioner. In addition, the temporal validation and verification of such systems in some applications, e.g., aeronautics, requires the probability of obtaining a worst-case response time larger than a given value in order to support dependable system functions. All these facts advocate moving toward statistical RTA, which instead of calculating absolute worst-case timing guarantees, computes a probabilistic worst-case response time estimate. The contribution of this paper is to present and evaluate such a statistical RTA technique which uses a black box view of the systems under analysis, by not requiring estimates of parameters such as worst-case execution times of tasks. Furthermore, our analysis is applicable to real systems that are complex, e.g., from a task dependencies perspective.**

***Keywords*-task execution and temporal dependencies; statistical response-time analysis; timing traces; black box view**

I. Introduction

When verifying non-functional properties related to system timing behavior, analysis techniques are traditionally dominated by *deterministic* Response-Time Analysis (RTA) [1], in which the adhering methods are based around some restrictive assumptions on how the system behaves and return an absolute Worst-Case Response Time (WCRT) guarantee. However, when it comes to RTA of real systems, it becomes complicated or even impossible to use deterministic RTA as these systems invalidate the simplistic assumptions that deterministic RTA relies on. The problems are exacerbated by the recent trends in system development. The trends include tasks not only in themselves exhibit complex control flow behavior but also there are explicit dependencies, e.g., complex transactions [2], and implicit dependencies, e.g., through caches or global shared state variables as illustrated in the case of the industrial robotic control system studied in [3]. Furthermore, it is difficult to obtain accurate task Worst-Case Execution Time (WCET) estimates as the input to RTA. This is due to the fact that the intricate temporal and execution dependencies between tasks [4] are too hard to handle [5]. Additionally, modern hardware that is designed to increase the overall system performance makes the runtime behavior of these systems inherently probabilistic in nature [6]. Therefore, it is considered beneficial to develop a RTA technique which does not rely on traditional system models used in deterministic RTA and WCET analysis methods.

Looking at the development of RTA for real-time embedded systems, as a step towards increasing its applicability in real systems, Bate [2] and Mäki-Turja [7], [8] have introduced more detailed RTA methods for dealing with *task execution precedence*, *task offsets* and task execution dependencies on *different modes*. These are still comparatively simple compared to some development practices and the analysis still relies on accurate WCET values. An option to deterministic RTA is Real-Time Queueing Theory (RTQT) [9] that provides a way to compute tasks' response time distributions using various real-time scheduling algorithms. However, preemption between customers (or tasks) is not permitted. In 1991, Atlas et al. [10] proposed a statistical rate monotonic scheduling for the *independent* task model, which is a generalization of the classical rate monotonic scheduling results for periodic tasks with high variable execution times and statistical Quality-of-Service (QoS) requirements. In this paper we develop a statistical black box view RTA technique that does not require any knowledge of accurate estimates of parameters of the system under analysis, such as the WCET of tasks. Our method also provides a solution to applications where users would like to know the probability of missing a specific deadline within some delay of the system functions, e.g., as required by the certificates in aeronautics [11], [12].

Our work is not the first paper proposing solutions for systems which cannot be properly analyzed by deterministic RTA [6], [13], [14]. It is interesting to stress that probabilistic WCET may also be obtained by using statistical methods [15], [16], [17], and the resulting distributions can be later used as inputs by the probabilistic response time analysis, such like [14], [18]. Unfortunately, calculating such distributions easily results in great complexity, and it is difficult to implement due to floating point problems [19]. Therefore in this work, we avoid such problems by presenting a statistical RTA based on traces of execution of target systems. A first result of such an analysis is introduced by RapidRT [20], which however has limitations in the following aspects:

1) RapidRT is lacking a rigorous statistical analysis framework needed to choose appropriate analysis parameters in order to achieve the desired properties of the results, e.g., the confidence in the value obtained and the likelihood in the response time being exceeded.

1052-8725/12 $26.00 © 2012 IEEE
DOI 10.1109/RTSS.2012.85

2) The parameters used in RapidRT (e.g., different sample sizes) are obtained according to empirical evidence given by few evaluation models, lacking the support given by expert statistical evidence.
3) The adopted search algorithm [20] was designed to determine the approximate best-fit algorithm parameters. In this paper an algorithm to determine the exact best-fit algorithm parameters is developed.
4) An automated tool chain and more thorough evaluations were lacking in the presentation of RapidRT.

Our goal in this paper is therefore to remove the above mentioned four limitations and improve prior work in obtaining a safe and tight upper bound on the WCRT estimate of the tasks under analysis, based upon a *calibrated* and *tight* predictive Probability Density Function (PDF) histogram of the task. To be specific, by calibrated we mean that the result given by our method can bound the true task WCRT, i.e., a *safe* task WCRT estimate. Tightness (which is related to *pessimism*) is about how close the safe result lies to the true task WCRT. Clearly, the tighter, the less pessimistic, the better. In particular, the technical contributions presented in this paper are two-fold:

1) We introduce the statistical RTA framework *RapidRT**, which can deal with a wide range of systems including complex industrial real-time embedded systems with intricate task execution and temporal dependencies as introduced in Chapter 3 in [4]. This is done by using a novel sampling method to achieve training data that will be corrected by a posterior process based on a set of statistical techniques, search algorithms and a certain task reliability requirement.
2) We present a new algorithm which significantly improves the accuracy of results (in terms of obtaining a tighter upper bound on the WCRT estimate of tasks) as well as reduces algorithm computing time, comparing prior work [20]. Specifically, such improvements have been demonstrated by an extensive set of simulation models containing representative features of real industrial control applications.

The remainder of the paper is organized as follows. Section II introduces the system model, the background theory, related work, and an overview of our analysis. Section III presents the RapidRT* framework in detail, followed by Section IV that describes the evaluation framework, our testbed and the developed tool chain, and the results of the evaluation. Finally, conclusions and future work are drawn in Section V.

II. Statistical Response-Time Analysis of Systems

This section describes the analysis used in this paper starting with the system model in Section II-A followed by the background on the Extreme Value Theory (EVT) used in Section II-B. Next the related work that has used EVT for analysis of real-time systems is reviewed in Section II-C. Finally an overview of our approach is given in Section II-D.

A. System Model

The systems considered in this work are comprised of a number of non-blocking tasks, and the knowledge of the parameters of these tasks is not required by our analysis. Furthermore, no specific scheduling algorithm is assumed in this work. Based on these properties of a black box approach, our analysis does not need any internal knowledge of tasks.

B. Background on the EVT Used

EVT is a theory of statistics extending the *Central Limit Theorem* [21] to the tail of a distribution. More precisely, for a given random variable $Y = max\{X_1, ..., X_n\}$ formed from the maxima of a set of n i.i.d. random variables X_i, EVT predicts that the distribution of Y will converge to the *Generalized Extreme Value* (GEV) distribution as n goes to infinity. Therefore EVT is used to model the likelihood of infrequent events. Typically, there are two common approaches [22] associated with EVT: *Peak Over Threshold* (POT) and *Block Maxima*. The first approach POT only studies the data which exceed over a threshold μ to a *Generalized Pareto Distribution*. One disadvantage of POT is that the choice of the threshold value μ is not evident. Furthermore, POT is known to not behave well when the data are not sufficient. Therefore, we use the block maxima approach in this paper.

Though the independent and identically distributed (i.i.d.) hypothesis is a common assumption made by statistics and probability theory in order to simplify the underlying mathematics, we will clarify a point about the difference between such a statistical independence required by statistics and the internal dependencies existing in the execution of tasks in the target system: EVT does not require the system under analysis to exclude any dependencies between non-blocking tasks, instead it analyzes the execution of the target which is independent of other executions. Moreover, in one system execution (i.e., X_i in our notation), we may capture task dependencies if they exist in the system, which are however not forbidden by EVT.

According to EVT, Y converges only to one of the three following forms: 1) *Gumbel* when the underlying distribution has a non-heavy upper tail (e.g., *Normal*) and, 2) *Fréchet* when the underlying distribution has a heavy upper tail (e.g., *Pareto*) and, 3) *Weibull* when the underlying distribution has a bounded upper tail (e.g., *Uniform*) and relates to minima (i.e., the smallest extreme value). Particularly, a probabilistic estimate can be computed by using the best-fit distribution parameters, according to a certain probability of being exceeded, as shown by Equation 1.

It is interesting to note that if we prove that the response time distribution of tasks converges to a Gumbel distribution, then according to EVT it cannot converge to Fréchet or Weibull. We now explain the reasoning allowing us to choose the correct distribution:

- We study the response time of probabilistic real-time systems.
- This work should provide results that are as close as possible to those given by any exact probabilistic RTA,

such as [23], [18], which take as input, probabilistic WCET estimates obtained by [16], [17]. Such probabilistic WCET estimates are proved to converge to the Gumbel Max distribution.
- Since the probabilistic worst-case response time is obtained by convolving the probabilistic WCET estimates, then their result is a distribution covering values with very low probability.
- Since Weibull is finite and Fréchet is not defined for all values (thus extremely low), then the result of a statistical RTA can only converge to the Gumbel Max distribution.

The conclusion that the Gumbel Max distribution is the correct one for the task RT sub-training data, has been backed up by the results of two Goodness-Of-Fit (GOF) hypothesis tests, i.e., the Exponential Tail (ET) test [24] and Chi-squared test [25] used in our work. More details can be found in Table I.

TABLE I
THE RESULTS OF TWO GOF HYPOTHESIS TESTS, I.E., THE ET TEST AND CHI-SQUARED TEST, HAVE SHOWN THAT THE I.I.D. TASK RT SUB-TRAINING DATA IN DIFFERENT EVALUATION MODELS, CONFORM TO THE GUMBEL MAX DISTRIBUTION.

Models	The ET test	The Chi-squared test	The confidence level
MV1-*	Pass	Pass	95%
MV2-*	Pass	Pass	95%
MV3-*	Pass	Pass	95%
MV4-*	Pass	Pass	95%

Equation 1 shows an example of using the percent-point function of the Gumbel Max distribution in EVT, to compute a probabilistic estimate, which is relevant to this work.

$$est = \mu - \beta \times log(-log((1 - P_{evt})^b)) \quad (1)$$

where μ and β are the two (best-fit) parameters of the Gumbel Max distribution, P_{evt} is the acceptance probability in EVT, and b is the block size.

C. Related Work for the Use of EVT in Real-Time Systems

EVT has been applied to the WCET analysis of a program in the recent years, and the featured work includes [26], [16], [17]. Specifically, in [26], Edgar presents the initial work on using EVT for WCET estimation, by firstly fitting the *raw* measured execution time sample to the Gumbel Max distribution based upon an unbiased estimator. A WCET estimate is then calculated using an excess distribution function. Hansen [16] improves the work by using a block maxima approach for the estimation of the probabilistic WCET, rather than fitting the raw execution time sample to the Gumbel Max distribution directly as in [26]. The limitations of [26], [16] when applying EVT are presented in [27] and these limitations are solved in [17]. Moreover, in [17] the authors have proved the convergence to the Gumbel Max distribution by using the ET test as the appropriate statistical test for this case. It is also important to note that as pointed out in [27], EVT makes the assumption required by statistics and probability theory, i.e., the sample elements in the sample have to be i.i.d..

D. Overview of Our Approach

The basic approach followed in this work is summarized by the following five stages:

1) The first stage of the analysis is to capture information from the available sources concerning the RT of the individual tasks, e.g., the existing system logs containing recorded RT data of the task under analysis. Specifically, no assumption is made as to whether such sources are collected either via simulation, or by monitoring the behavior of the actual target using *software probes* as introduced in Chapter 7 in [28].
2) Next, a task RT training data (consisting of a number of task RT sub-training data corresponding to the K statistical models to be introduced in Section III-A) is sampled from the set of available sources achieved in Step One. Specifically, each sub-training data is taken such that an i.i.d. assumption can be made, and such that there are sufficient samples allowing for appropriate tuning of the final analysis.
3) Then, a posterior statistical correction process is performed to allow the primary objective of the analysis to be achieved, i.e., to obtain a safe and tight WCRT estimate of tasks. This step decomposes the reliability target for the WCRT of tasks into a number of probabilities to be used in the statistical analysis. Such probabilities are associated with different analysis context, such as our sampling method, EVT and the GOF hypothesis test (of which more details are given in Section III-B).
4) Given an appropriate task RT sub-training data, the statistical analysis in the posterior statistical correction process is tuned such that the maximum of a probability distribution (in this case the Gumbel Max distribution) generated with a given set of parameters is a sufficiently close match, at the required confidence level, to the maximum of the actual distribution of the sub-training data. At this step, a calibrated and tight PDF histogram of the task WCRT consisting of the estimates of the maximum of each task RT sub-training data is obtained.
5) The final stage is to use the achieved task WCRT PDF to obtain the final WCRT of the task under analysis.

The first and fifth of the stages above are effectively the same as Edgar's approach, with the obvious difference here response times are analyzed instead of execution times. The second stage has been included to overcome the issue that the EVT's being used are intended for data without having strong dependencies and hence conforming to the i.i.d. assumption. The third stage decomposes the system's objectives into a set of probabilities used by the EVT concerning the Gumbel Max distribution, the GOF hypothesis test, different search algorithms, etc. Finally, the parameters of the i.i.d. task RT sub-training data are tuned to give better results using the percent-point function of the Gumbel Max distribution (as shown in Equation 1) in the fourth stage. Figure 1 shows the corresponding work flow. The first four stages of our method ensure the i.i.d. hypothesis required by statistics and

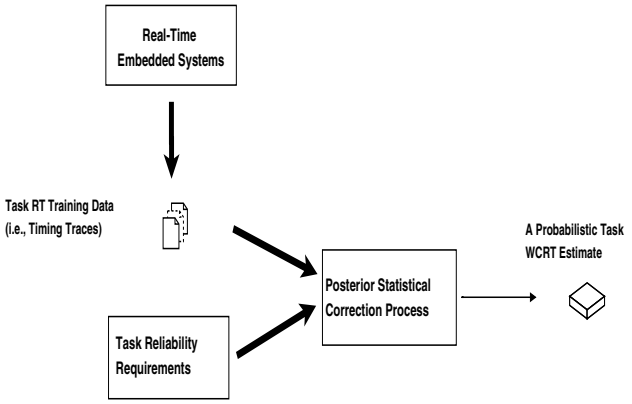

Fig. 1. The work flow about RapidRT.

probability theory as well as EVT. Firstly our sampling method (to be introduced in Section III-A) is applied on the original RT data to remove some strong dependencies, in order to provide the i.i.d. samples. Secondly, the adopted block maxima approach also gives a set of independent data [22] which are used for obtaining an estimate later.

III. $RapidRT^*$ Analysis Framework

Note that we use the notation RapidRT*, i.e., by adding the symbol * to RapidRT, in order to differentiate between our new framework and the original framework presented in prior work [20].

The core of the RapidRT* analysis framework presented in this section is the law of total probability [29], which tells us that a predictive PDF of tasks' WCRT can be given by

$$P(rt_{est}) = \sum_{k=1}^{K} p(rt_{est}|M_k)p(M_k|rt_k^T) \quad (2)$$

where K is the number of statistical models required by the analysis framework, each of which the form is a Gumbel Max distribution, $p(rt_{est}|M_k)$ is the task WCRT PDF based on the model M_k alone (which is corresponding to the task RT sub-training data rt_k^T), and $p(M_k|rt_k^T)$ is the posterior probability of the model M_k being correct given the task RT sub-training data rt_k^T. Here $p(M_k|rt_k^T)$ reflects how well the model M_k fits the sub-training data.

In the following, we introduce the RapidRT* analysis framework in detail by firstly presenting the description of the proposed sampling method in Section III-A, followed by the underpinning posterior statistical correction process in Section III-B. Also, in this section the overall algorithm is presented in pseudo-code format.

A. Sampling

As introduced in Section II-D, the first and second stage in our approach is the sampling method which specifically consists of three steps as follows:

Step One *The collection of representative task RT samples*: First, the sample which could be representative of the underlying task RT population has to be collected in a way without introducing any bias [30]. Furthermore, this step concerns with the issues of *sampling strategy* and *sample size*. The samples achieved in this step will be called *Simple Random Sampling (SRS)*. In this work, we employ the technique of SRS [21], which gives every possible observation of a given size (i.e., system inputs) the same chance to be chosen (based around the usage of a *uniform distribution*). The main benefit of SRS is that it guarantees that the chosen sample is representative of the underlying population [30], without introducing any bias to sampling.

Step Two *The collection of the task RT sub-training data rt_k^T satisfying the i.i.d. assumption*: Based around the SRS task RT samples achieved in the previous step, we next construct the sub-training data rt_k^T which satisfies the i.i.d. assumption (and hence can be used by the posterior statistical correction process) and contains the sample with higher response times, compared with the sampling method in prior work RapidRT.

Step Three *The collection of the training data rt^T by repeating Step One and Two for K times* and hence comprising K sub-training data rt_k^T, which results in $rt^T \leftarrow rt_1^T, ..., rt_k^T$, for the K statistical models corresponding to the K sub-training data.

The main problem in Step One is the determination of the appropriate sample size of an SRS task RT sample, which depends on a number of accuracy factors that must be considered. In the context of RTA, such factors are *desired precision of results*, *confidence level* and *the degree of variability*. We introduce their definitions (based on [31] which introduces how to determine the sample size in general) as follows:

1) *Desired precision of results*: The level of precision is the closeness with which the sample predicts where the true values in the population lie. The difference between the sample and the real population is called the *sampling error*. The commonly assumed sampling error is $\pm 3\%$ and $\pm 5\%$. In addition, *the higher level of precision required, the larger sample sizes and the higher cost to achieve those samples will be.*
2) *Confidence level*: The confidence level refers to the percentage of the collected sample that would have the true population value at the accepted level, at which the sample is within the range of precision. The 95% confidence level is more commonly used in practice.
3) *Degree of variability*: Variability is the degree to which the attributes or concepts being measured in the questions are distributed throughout the population.

The different sample sizes within our method are directly related to the utilization of EVT. For instance, the ET test works for poor data (size of 20), and it provides a Gumbel distribution that is known to be pessimistic. If more data is included, then the corresponding estimation is more accurate. The application of EVT to the problem of the estimation of probabilistic WCET indicates that the minimum sample size is

less than 1 000. In addition, an interested user may determine such minimum sample size for the estimation of probabilistic WCRT by adding a convergence step to our algorithm and then comparing consecutive results of EVT on those data. However, this is beyond the purpose of this paper, and it is kept as future work.

According to the statistical evidence [31], the determination of such an SRS task RT sample size can be done using the above definitions and the table in Appendix Two in [31]. In this work, the detailed explanation is: since the underlying task RT population is extremely large due to a very large system search space consisting of task arrival jitter, task execution time and environmental input stimulus [32], we therefore use the maximum of all the specified population sizes in the table, i.e., 100 000. Since the variability of task RT population is too difficult to estimate, it is therefore best to use the conservative figure of 50% as suggested by [31]. The confidence level is chosen as 95%, which is standard in applied practice. The sampling error is $\pm 3\%$. Note that the reason for choosing the sampling error to be $\pm 3\%$, other than $\pm 5\%$, is that the larger the sample size is, the higher level of precision is, which is especially important when we achieve the SRS task RT samples. Consequently, by using the information previously given, the sample size of each SRS task RT sample is determined to be 1 099, when the population size is 100 000, variability is 50% and sampling error is $\pm 3\%$.

In Step Two, there are no dependencies between the maximum of any two independent SRS task RT samples, which are collected by running two independent executions of the system being analyzed. We are interested in the prediction of the task WCRT. Therefore, maxima are taken to achieve the task RT sub-training data rt_k^T which satisfies the i.i.d. assumption, and hence can be used by our posterior statistical correction process. Additionally, when we fit the sample elements in the sub-training data rt_k^T to the Gumbel Max distribution (in EVT) for the statistical model M_k (i.e., the i.i.d. task RT sub-training data fitting process hereafter), the corresponding sample size is 191. This is the smallest sample size when the corresponding i.i.d. task RT sub-training data fitting process is successful by using the upper-part binary search [33], based on our evaluation (as shown in Table II).

In Step Three using the same table (Appendix One in [31]), the sample size of the task RT training data rt^T (which contains K task RT sub-training data rt_k^T) of 398 is chosen when the population size is 100 000, variability is 50% and sampling error is $\pm 5\%$. Note that 398 sub-training data samples (when the sampling error is $\pm 5\%$) is sufficient enough to construct the *task WCRT estimate sample* (in Section III-B) according to our evaluation. The cost of sampling in our algorithm will become expensive if i.e., 147 146 209 more sample elements than necessary are taken (i.e., $1\,099 \times 191 \times 1\,099 - 1\,099 \times 191 \times 398$). Having more samples would not yield significant improvements in the results.

In order to have a better understanding about the sample sizes in different context of our proposed method, we summarize the above information as follows:

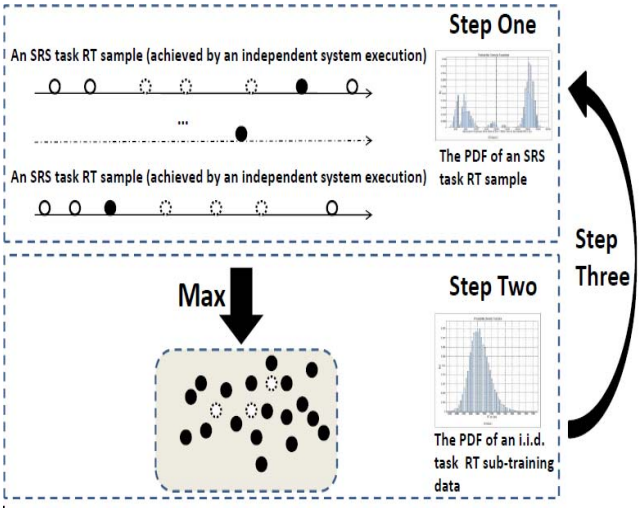

Fig. 2. Illustration of our proposed sampling method in RapidRT*. Since we are only interested in showing the shape of different response time samples of the same task achieved in different steps, the detailed ranges of task RT data are not interesting to show clearly in the picture.

1) *An SRS task RT sample* contains a set of measured and representative task RT data per each system execution by using SRS (without introducing any bias when we achieve those samples), and its sample size is 1 099.
2) *A task RT sub-training data rt_k^T for the model M_k* contains a set of maxima of SRS task RT samples, and its sample size is 191.
3) *A training data rt^T for K statistical models* (referred as the task WCRT estimate sample in the posterior statistical correction process) is comprised of a set of sub-training data rt_k^T, and its sample size is 398.

The proposed sampling method is illustrated in Figure 2, and its implementation is shown in lines from 5 to 8 in Algorithm 1 (to be introduced in Section III-B). The key aspect of this figure is that the multi-modal actual distribution of the task response time is transformed into a *positive-skewed* (i.e., right-skewed or right-tailed) distribution that can summarize the tail of the actual distribution. Then, following Step Three we repeat Step One and Two for 398 times.

TABLE II
IN THE TABLE, × INDICATES THAT THE I.I.D. TASK RT SUB-TRAINING DATA FITTING PROCESS FAILED, CORRESPONDING TO THE CHOSEN SAMPLE SIZE (WHICH IS DETERMINED BY THE UPPER-PART BINARY SEARCH [33]); √ REPRESENTS THE OPPOSITE CASE OTHERWISE.

	100	150	175	188	194	**191**	190
GOF test	×	×	×	×	√	√	×

B. Posterior Statistical Correction Process

The RapidRT* analysis framework takes five input parameters, and returns a probabilistic WCRT estimate of the task τ under analysis. To be specific, the five input parameters are: τ (the task under analysis), n (the sample size of the task RT

training data rt^T), m (the sample size of a task RT sub-training data rt_k^T), l (the sample size of an SRS task RT sample) and P_{rr} (the given task reliability requirement). Specifically, P_{rr} is the probability of the task τ having a response time violation, which is, e.g., no more than 10^{-9} per hour or 8.784×10^{-6} per year (i.e., $24 \times 366 \times 10^{-9}$) when P_{rr} is 10^{-9}.

In order to use the given task reliability requirement in the posterior statistical correction process in RapidRT*, we have to decompose the algorithm parameter P_{rr} into a set of probabilities according to different kinds of context as follows:

1) $P_{sampling}$ is the probability of having some sample elements which are not included in the task RT training data rt^T collected by our proposed sampling method. Since the confidence level of the training data rt^T given by our sampling method is 95%, the value of $P_{sampling}$ is 0.05 (i.e., $1 - 0.95$).
2) Since the confidence interval for the task WCRT estimate predictive PDF is chosen at the level of 95% and the significance level of the hypothesis tests in the posterior statistical correction process is 0.05, the value of the probabilities $P_{REF_{cl}}$ (the significance level of the task WCRT estimate predictive PDF) and P_{GOF} (the significance level of the GOF hypothesis test) is therefore 0.05. The value 0.05 is a typical value based on preliminary assessments that provided appropriate results [34])
3) P_{evt} is the acceptance probability in EVT which is used in the Gumbel percent-point function (as shown in Equation 1) to obtain a probabilistic WCRT estimate of the task.

In the RapidRT* analysis framework, all the steps outlined above have to be taken one by one, as a result, P_{rr} is a multiple of $P_{sampling}$, $P_{REF_{cl}}$, P_{GOF} and P_{evt} as expressed by Equation 3.

$$P_{rr} = P_{sampling} \times P_{REF_{cl}} \times P_{GOF} \times P_{evt} \tag{3}$$

where $P_{sampling}$, $P_{REF_{cl}}$ and P_{GOF} are 0.05, and P_{rr} is 10^{-9}.

Consequently, the value of P_{evt} can be calculated by the following equation:

$$P_{evt} = \frac{P_{rr}}{P_{sampling} \times P_{REF_{cl}} \times P_{GOF}} \tag{4}$$

where $P_{sampling}$, $P_{REF_{cl}}$ and P_{GOF} are 0.05, and P_{rr} is 10^{-9}, and hence P_{evt} is 8×10^{-6} (i.e., $\frac{10^{-9}}{0.05 \times 0.05 \times 0.05}$).

Next, RapidRT* will verify if the task WCRT estimate sample consisting of n probabilistic task WCRT estimates (corresponding to K task RT sub-training data, where $K = n$) conforms to a normal distribution or not, based around the result given by the non-parametric GOF hypothesis test Kolmogorov-Smirnov test [35] (the KS test hereafter). If it is, then the algorithm will calculate the confidence interval (i.e., the CI hereafter) of the task WCRT estimate sample at the confidence level 95%, and choose the upper bound on the CI as the final WCRT of the task. Otherwise, if the task WCRT estimate sample cannot be fitted to a normal distribution, the *resampling* statistical method *bootstrapping* will be adopted. Specifically, the way of using the most widely-used bootstrapping method i.e., *bias-corrected and accelerated (BCA) bootstrap* to obtain the confidence intervals of a population can be found in [36]. The outline of the posterior statistical correction process in the RapidRT* analysis framework is as follows:

1) Compute the task WCRT estimate est_k per each sub-training data rt_k^T corresponding to the statistical model M_k (which uses the Gumbel Max distribution in EVT and a given task reliability requirement). The detailed steps are as follows, i.e.,
   a) Set the initial block size b to 1, for each sub-training data rt_k^T.
   b) If the number of blocks $k = \left\lfloor \frac{m}{b} \right\rfloor$ is less than 30, the algorithm stops since there are not enough samples to generate an estimate [16].
   c) Segment m response times into blocks of size b, and for each of the $\left\lfloor \frac{m}{b} \right\rfloor$ blocks find its maximum.
   d) Estimate the best-fit Gumbel parameters μ and β to the block maxima by using the exhaustive search algorithm as shown in lines 16 to 23 in Algorithm 1.
   e) Calculate a probabilistic task WCRT estimate based on the best-fit Gumbel Max parameters obtained through Step d) (i.e., μ, β) and the acceptance probability in EVT (i.e., P_{evt}). In addition, *evtgumbelmax* in line 19 in Algorithm 1 represents the implementation of the Gumbel percent-point function as shown by Equation 1 (introduced in Section II-B).
2) After verifying if the task WCRT estimate sample (i.e., $EST \leftarrow est_1, ..., est_i, ..., est_n$) can successfully be fitted to a normal distribution by using the KS test, RapidRT* will return the result, i.e., $\overline{EST} + 2\sigma_{EST}$ (the sum of *mean* value and 2 *standard deviation* of EST at the confidence level 95%).

The precise description of the algorithm using pseudo-code is outlined in Algorithm 1, in which the lines 8 to 33 show the posterior statistical correction process.

Parameters:

τ: the task under analysis

n: integer value - the sample size of a task WCRT estimate sample

m: integer value - the sample size of a task RT sub-training data

l: integer value - the sample size of an SRS task RT sample

P_{rr}: double value - the given task reliability requirement in terms of a certain probability

Returns:

rt_{est}: double value - the WCRT estimate of the task τ

IV. Case study: an industrial robotic control system

This section is split into four parts: Section IV-A presents the setup of the evaluations performed in this case study.

Algorithm 1 $RapidRT^*(\tau, n, m, l, P_{rr})$

$P_{sampling}, P_{REF_{cl}}, P_{GOF} \leftarrow 0.05, P_{rr} \leftarrow 10^{-9}$
$P_{evt} \leftarrow 8 \times 10^{-6} \leftarrow \dfrac{P_{rr}}{P_{sampling} \times P_{REF_{cl}} \times P_{GOF}}$
$n \leftarrow 398, m \leftarrow 191, l \leftarrow 1\,099$
for all est_i such that $1 \leq i \leq n$ **do**
  for all $rt_{i,j}$ such that $1 \leq j \leq m$ **do**
    $rt_{i,j} \leftarrow MAX(SRS(\tau, l))$
  end for
  $X_i \leftarrow rt_{i,1}, ..., rt_{i,m}$
  $b \leftarrow 1$
  $k \leftarrow \left\lfloor \dfrac{m}{b} \right\rfloor$
  $\mu, \beta \leftarrow 0$
  $success \leftarrow false$
  while $k \geq 30$ **and** $success = false$ **do**
    $S_i \leftarrow s_{i,1}, ..., s_{i,k} \leftarrow segment(X_i, b)$
    $Y_i \leftarrow y_{i,1}, ..., y_{i,k} \leftarrow maxima(S_i)$
    if $passChiSquareTest(Y_i, GumbelMax, P_{GOF}) > 0$ **then**
      $success \leftarrow true$
      $\mu, \beta \leftarrow ChiSquareTest(Y_i, P_{GOF})$
      $est_i \leftarrow evtgumbelmax(\mu, \beta, P_{evt}, b)$
    else
      $b \leftarrow b + 1$
      $k \leftarrow \left\lfloor \dfrac{m}{b} \right\rfloor$
    end if
  end while
end for
$EST \leftarrow est_1, ..., est_i, ..., est_n$
if $passKS(EST, Normal, P_{GOF}, P_{REF_{cl}})$ **then**
  $\overline{EST} \leftarrow \dfrac{1}{n} \times \sum_{i=1}^{n} est_i$
  $\sigma_{EST} \leftarrow \sqrt{\dfrac{1}{n} \sum_{i=1}^{n} (est_i - \overline{EST})^2}$
  $rt_{est} \leftarrow \overline{EST} + 2\sigma_{EST}$
else
  $rt_{est} \leftarrow bootstraptest(EST)$
end if
return rt_{est}

Section IV-B outlines the evaluation models followed by our testbed and toolchain in Section IV-C. Finally, Section IV-D presents our results and their validity.

A. Evaluation Setup

In the evaluation of the proposed framework, we are interested in the tightness of analysis results given by RapidRT*, the corresponding improvements over RapidRT from the perspectives of sampling methods, the pessimism of analysis results as well as the computing time.

1) Tightness: The first property to investigate in this evaluation is whether the results are calibrated and tight, or not. The tightness of the results obtained by RapidRT* is of highest importance from an applicability and usefulness point of view. Specifically, we are not allowed to provide optimistic or too pessimistic results of the task WCRT estimate. If the results were optimistic, then they are invalid, i.e., they are not calibrated. If the results would be a little bit pessimistic, they are safe and good to use. If the results would be too pessimistic, then they are not tight and as a result less useful from a practical point of view.

Tightness of the analysis results is difficult to evaluate, unless we have access to the true WCRT of the task under analysis. Hence, in order to assess the tightness of the results obtained by RapidRT*, we need to compare such results of RapidRT* to some known best-practice response time values. A natural candidate for comparison would be to use the traditional RTA – a technique commonly used to compute the exact WCRT of tasks. However, we cannot compare our results against such traditional RTA techniques, not only because there are different analysis views adopted, but also because the system model assumptions in traditional RTA are invalid, due to the black box view with possible intricate task execution and temporal dependencies in our target systems.

On the other hand, as a reference for comparison, we can compare the performance of RapidRT* against HCRR [32] – a meta-heuristic search algorithm that guides the traditional *Monte Carlo Simulation* (MCS), in order to find good and representative response time values that cannot be obtained by the traditional MCS. However, HCRR requires an accurate model of the system under analysis in order to compute useful results. Under such circumstances, HCRR has shown to perform very well, hence, hinting on a good performance also on models that cannot be analyzed by using traditional RTA techniques, such as our evaluation models.

It should be noted that RapidRT* does not require an accurate model of the system under analysis, but merely the training data (in terms of task RT traces) to be used by the analysis. In our experiments, however, we do have accurate models of the evaluation systems such that the values given by HCRR could be used to indicate the level of pessimism of RapidRT* results, as the known best-practice WCRT of tasks.

2) Improvements: Since we have removed a number of limitations in RapidRT such that the performance of the new analysis RapidRT* should significantly increase, the second property to investigate is the improvement over our prior work. Interesting properties to investigate include improvements in our sampling method, the pessimism in the computed task response time estimates, and the performance with respect to computing time.

B. Evaluation Models

In order to assess the performance of RapidRT*, we decide to evaluate a large number of models of applications. To be specific, our evaluation models are based around four different *base models*, which are designed to include some behavioral mechanisms adopted by an industrial robotic control system. Moreover, we have applied system evolution scenarios to these

base models to create a total of 32 different models used for our evaluation.

Next, we outline the characteristics of our evaluation models, followed by some details of the base models, as well as the way of designing different variations of the base models pertaining to system evolution.

1) Model Characteristics: The characteristics of the behavioral mechanisms in our evaluation models include intricate task execution and temporal dependencies, e.g., asynchronous message-passing by sending and receiving messages from buffers (as shown in lines 1 to 4 in Figure 3), *execute* statements representing some computation time taken by the (sub-)task (as shown in Line 6 in Figure 3), Global Shared State Variables (GSSVs) used in selecting control branches in tasks, runtime changeability of task priorities and periods (as shown in lines 8 to 13 in Figure 3), and task offsets.

```
1   msg = recvMessage(MyMessageQueue);
2   while (msg != NO_MESSAGE){
3         process_msg(msg);
4         msg = recvMessage(MyMessageQueue);}
5
6   execute(for_some_time);
7
8   if (GSSV1 == 1){
9     var1 = 10;
10    tcb->period = 20000;}
11  else{
12    var2 = 5;
13    tcb->period = 10000;}
```

Fig. 3. Iteration-loop wrt. message passing and GSSVs, and runtime changeability of task priorities and periods in the tasks in the industrial robotic control system evaluated in our work.

2) Base Models: As we mentioned previously, our evaluation models are centering around four different base models, which are designed to have increasing complexity. The differences between these base models are mainly concerning the contained task execution and temporal dependencies as well as the number of sub-tasks, queues and GSSVs, which are increased from MV1-* to MV4-*, as shown in Table III, making them more complex. The system architecture of the most complicated base model MV4-*, is shown in Figure 4. More details about our base models, including their detailed pseudo code can be found in [37] (unfortunately we have to leave out such details in this paper due to space limitations).

3) Model Variations: In order to have a large number of evaluation models, we simply apply some typical system evolution scenarios to the four base models, each resulting in a set of new system models to analyze. In doing this, we either increase or decrease the execution time of sub-tasks in tasks in our base models, which reflects the scenarios of the change on system CPU speed. In practice, such scenarios can happen e.g., when the system is ported to a new hardware platform, which thereby is upgraded or downgraded according to new design requirements.

- For the increase in system CPU speed, we limit ourselves to use 2, 5 and 10 as relevant factors in the work.
- For the decrease in system CPU speed, we limit ourselves to use 0.9, 0.8, 0.7 and 0.6 as relevant factors in the work. It is worth noticing that a factor of 0.5 will result in the corresponding known best-practice task WCRT longer than the corresponding task period, i.e., if one would assume the deadline of a task to be equal to its corresponding period, applying a factor of 0.5 would violate schedulability of the system.

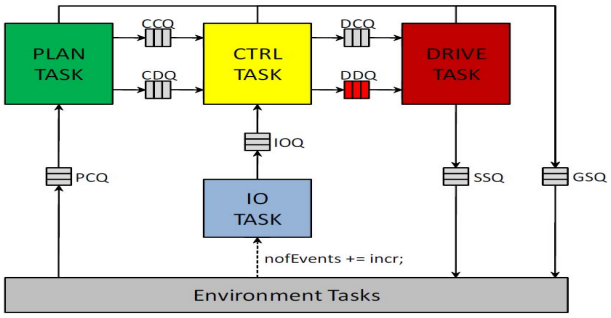

Fig. 4. An industrial robotic control system architecture.

Each of the factors highlighted above is marked as a postfix to the model name, i.e., MV1-2 represents that the CPU speed of the model variation MV1-2 is two times as fast as the CPU speed of the model MV1;.In other words, the corresponding execution time of sub-tasks in MV1-2 is 0.5 times as large as the corresponding ones in MV1.

To summarize, the task parameters used in these evaluation models are shown in Table IV, where the *CTRL* task is the task under analysis in Section IV-D, which is the task with the most complicated timing behavior.

In addition, the variability within the Execution Time (ET) of tasks is also taken into consideration in all our evaluation models. One example of such variability is shown by Figure 5, where the variability of task execution time is featured in terms of a multi-modal distribution having two peaks.

C. Implementation

RapidRT* is developed to work on samples of timing data, and these samples can be taken either from monitoring of a real system, or from sampling a simulation of a system. For our evaluation purpose, the timing data in the experiments presented in this section are taken from the latter, i.e., by simulating the system models.

The implementation of RapidRT* consists of two parts, and its first part, i.e., our sampling method, is implemented in a relatively straightforward manner, which chooses the maximum of the recorded SRS task RT sample corresponding to each of a set of independent simulation runs by using MCS. Typically, MCS is realized by providing some generated simulator input data (conforming to a uniform distribution) in RTSSim, which is a simulation framework based on C programming language. RTSSim includes the typical RTOS services to the task models, such as task scheduling (e.g., fixed-priority preemptive scheduling), IPC via message passing and synchronization (semaphore). RTSSim also measures

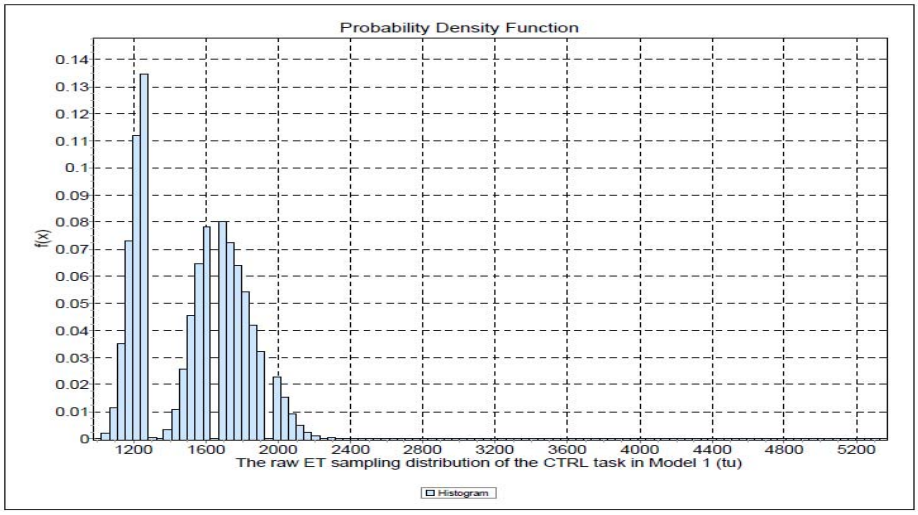

Fig. 5. One example of the variability of the ET distribution of the task under analysis in our evaluation models.

TABLE III
MODELS DESCRIPTION AND THE RELEVANT COMPLEXITY. THE LOWER NUMBERED COMPLEXITY IS LESS COMPLEX, I.E., 1 STANDS FOR THE SIMPLEST EVALUATION MODEL.

Models	Sub-tasks	Queues	GSSVs	Description	Complexity
MV1-*	40	7	8	IPC via the bounded number of messages and GSSVs, and task offsets.	1
MV2-*	42	7	8	IPC via the bounded number of messages and GSSVs, and run-time changeability of priorities of tasks, and task offsets.	2
MV3-*	42	7	8	IPC via the bounded number of messages and GSSVs, and run-time changeability of priorities and periods of tasks, and task offsets.	3
MV4-*	59	12	10	IPC via the *unbounded* number of messages and GSSVs, and runtime changeability of priorities and periods of tasks, and task offsets.	4

response time and execution time for each finished instance of a specified task, and reports the maximum value observed during the simulation. The interested reader can refer to [38] for a thorough description of RTSSim. The other part of RapidRT*, i.e., the posterior statistical correction process, is implemented as an executable program with a friendly GUI developed using Microsoft's C# programming language. More details can be found in [39].

TABLE IV
TASKS AND TASK PARAMETERS FOR EVALUATION MODELS. THE LOWER NUMBERED PRIORITY IS MORE SIGNIFICANT, I.E., 0 STANDS FOR THE HIGHEST PRIORITY. CTRL_H AND CTRL_L REPRESENT THE CTRL TASK WITH A HIGHER AND A LOWER PRIORITY RESPECTIVELY.

Task	*Period (μs)*	*Offset (μs)*	*Priority*	*Models*
DRIVE	2 000	12 000	2	MV1-*, MV2-*, MV3-*, MV4-*
CTRL_H	20 000	0	4	MV2-*, MV3-*, MV4-*
IO	5 000	500	5	MV1-*, MV2-*, MV3-*, MV4-*
CTRL_L	10 000	0	6	MV1-*, MV2-*, MV3-*, MV4-*
PLAN	80 000	0	8	MV1-*, MV2-*, MV3-*, MV4-*

D. Evaluation Results

In this section we discuss the evaluation results, by firstly presenting the calibration and tightness of the task WCRT computed by RapidRT*, which is followed by its improvement over our prior work RapidRT.

1) Tightness: Here we evaluate the performance of RapidRT* in terms of calibration and tightness of the results, comparing the WCRTs derived with an alternative best practice approach, HCRR. In Table V and VI, the values in Column *BP* are the values for the WCRTs derived with the alternative approach for all the models MV1-*, MV2-*, MV3-* and MV4-*. The inherent pessimism in the resulting WCRT estimates that are provided by RapidRT* is expressed as percentages which are calculated in the following way, i.e., $\frac{rt_{est}^{RapidRT^*} - rt_{BP}}{rt_{BP}} \times 100\%$. As shown in the tables, the results given by RapidRT* are at most 14.78% more pessimistic compared with the known best-practice task WCRTs.

Note that one may argue, after looking at the results, why not use the best practice approach as it gives accurate task response time values, i.e., hinting on less pessimism compared

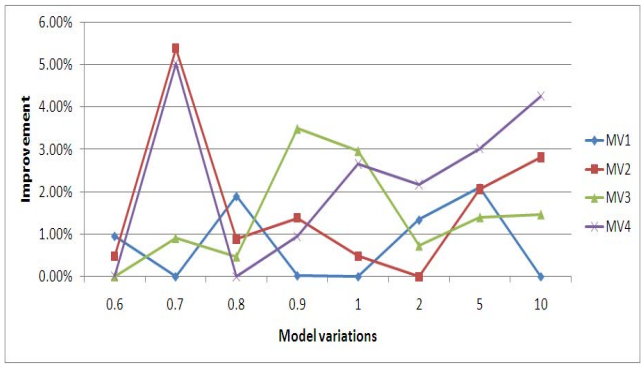

Fig. 6. Our new sampling method can find higher task response time data in all evaluation models, comparing the sampling method in prior work RapidRT.

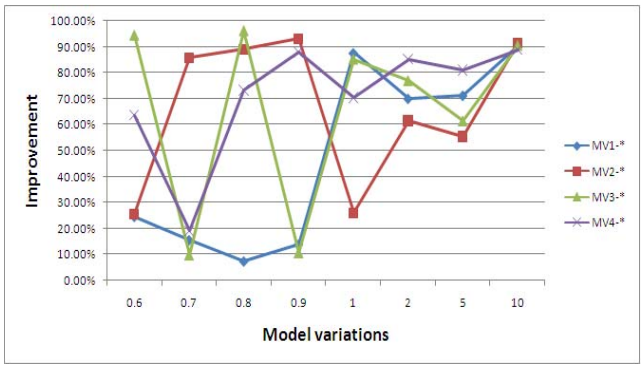

Fig. 7. The improvement of the computing time taken by RapidRT* compared with the prior work RapidRT, when they are used to analyze the same evaluation models.

to RapidRT*. Recall that the restriction of the best practice approach is that it needs an accurate model of the system under analysis, whereas RapidRT* does not have such a requirement. Instead we believe of a broader applicability of RapidRT* in complex industrial embedded software systems, while the evaluation setup in this section is merely designed to investigate the properties of calibration and tightness.

2) Improvements: Here we firstly evaluate the performance of the proposed sampling method, in terms of the improvements of results with regard to the collected higher task RT data, comparing the sampling method in our prior work RapidRT. As shown in Figure 6, our sampling method can find higher task RT data in all the evaluation models with the maximum improvement 5.37% comparing our prior work. More importantly, the samples achieved by our method satisfy the i.i.d. assumption which is required by any statistical correction process, while the samples achieved by the traditional sampling method Monte Carlo Experiment [40] are not i.i.d., due to the existence of dependencies between tasks.

Next we present the improvement of RapidRT* concerning the pessimism in the computed task response time estimates, comparing the previous work RapidRT. Typically, the percentages of such improvement (i.e., $(\frac{rt_{est}^{RapidRT}-rt_{BP}}{rt_{BP}} - \frac{rt_{est}^{RapidRT^*}-rt_{BP}}{rt_{BP}}) \times 100\%$) are shown in Column *Imprv.* in both Table V and VI. As shown in the tables, the significant improvements compared with the ones obtained by the prior work, are shown by a reduction of pessimism up to 36.26%.

Finally, looking at the time it takes to practically use RapidRT* on a model of a system, Figure 7 shows that the *computing time* taken by RapidRT* is dramatically shortened in the range from 7.45% and 96.14%, compared to the time consumed by RapidRT. Typically, for the most complicated models MV4-*, the relevant improvement is at least 19.32% and on average 71.22%. The reason here is that in this work, we effectively reduce different sample sizes used in our statistical analysis framework based upon some statistical evidence (as introduced in Section III-A). Additionally, such sample sizes also result in more accurate task WCRT computation, as shown by our evaluation.

TABLE V
RAPIDRT* CAN MORE TIGHTLY BOUND THE KNOWN BEST-PRACTICE WCRT OF TASKS IN ALL EVALUATION MODELS, WHEN CPU SPEED OF SYSTEMS IS INCREASED (OR THE ET OF ALL SUB-TASKS IN TASKS IS DECREASED), COMPARING PRIOR WORK RAPIDRT.

Models	BP	RapidRT	RapidRT*	Pessimism	Imprv.
MV1-1	4 432	5 196.68	**4 512.923**	1.83%	15.43%
MV1-2	1 842	2 019.36	**1 954.082**	6.09%	3.54%
MV1-5	740	809.42	**784.087**	5.96%	3.42%
MV1-10	346	387.67	**374.283**	8.17%	3.87%
MV2-1	5 332	6 994.932	**5 986.396**	12.27%	18.91%
MV2-2	1 892	2 073.429	**1 970.612**	4.16%	5.43%
MV2-5	760	831.996	**790.591**	4.03%	5.45%
MV2-10	356	397.434	**394.068**	10.69%	0.95%
MV3-1	4 432	5 921.68	**5 086.828**	14.78%	18.84%
MV3-2	1 842	2 032.48	**1 949.110**	5.82%	4.53%
MV3-5	740	813.03	**788.158**	6.51%	3.36%
MV3-10	346	387.52	**375.177**	8.43%	3.57%
M4-1	8 474	8 698.289	**8 595.674**	1.44%	1.21%
M4-2	3 804	4 108.012	**4 058.691**	6.70%	1.30%
M4-5	1 342	1 414.212	**1 387.387**	3.38%	2.00%
M4-10	668	709.705	**695.133**	4.06%	2.18%

TABLE VI
RAPIDRT* CAN MORE TIGHTLY BOUND THE KNOWN BEST-PRACTICE WCRT OF TASKS IN ALL EVALUATION MODELS, WHEN CPU SPEED OF SYSTEMS IS DECREASED (OR THE ET OF ALL SUB-TASKS IN TASKS IS INCREASED), COMPARING PRIOR WORK RAPIDRT.

Models	BP	RapidRT	RapidRT*	Pessimism	Imprv.
MV1-0.9	4 901	5 897.087	**5 200.832**	6.12%	14.21%
MV1-0.8	6 857	9 851.958	**7 606.950**	10.94%	32.74%
MV1-0.7	7 823	9 654.238	**8 452.365**	8.05%	15.36%
MV1-0.6	9 340	12 245.315	**10 383.397**	11.17%	19.93%
MV2-0.9	6 796	7 594.867	**7 010.435**	3.16%	8.60%
MV2-0.8	7 419	9 173.447	**8 406.046**	13.30%	10.34%
MV2-0.7	9 204	10 407.086	**9 864.532**	7.18%	5.89%
MV2-0.6	11 127	12 195.437	**11 718.066**	5.13%	4.29%
MV3-0.9	4 901	6 439.785	**5 229.438**	6.70%	24.70%
MV3-0.8	6 857	9 534.211	**7 497.974**	9.35%	29.70%
MV3-0.7	7 823	10 199.179	**8 891.231**	13.66%	16.72%
MV3-0.6	9 340	12 800.490	**9 414.106**	0.79%	36.26%
M4-0.9	9 312	10956.003	**10 153.057**	9.03%	8.62%
M4-0.8	11 089	12 138.439	**11 141.422**	0.47%	8.99%
M4-0.7	13 108	14 811.001	**13 639.713**	4.06%	8.94%
M4-0.6	15 167	21 536.190	**16 706.451**	10.15%	31.84%

Overall, these results give us confidence to state that RapidRT* can handle a real-life-scale task system in a few minutes, which consequently shows its practical feasibility.

3) Experiments summary: Summarizing the above observations, our evaluation results confirm the following points:

- Our proposed statistical RTA method (which does not require any knowledge about the task execution time), corresponding to different model complexities, can find calibrated and tight task response time estimates, by bounding the known best-practice task worst-case response times in all our evaluation models. To be specific, such estimates are in the close proximity to the known best-practice WCRT of tasks, i.e., less than 15% more pessimistic compared with our best practice approach.
- Comparing prior work, the pessimism of the computed response time estimates given by RapidRT* is reduced up to 36.26%, and such computed response time estimates obtained by the new analysis framework is always more accurate than the ones given by RapidRT.
- The computing time required by the new method is significantly shortened by on average 62.62% (for all the evaluation models), comparing its previous version. On average the trials took only a few minutes to compute. This is an important step toward handling real life-scale task systems exhibiting high degrees of variability.

V. Conclusions and Future Work

In this paper, we have developed a new statistical RTA technique to derive calibrated and tight upper bounds on response time estimates of tasks in real-time embedded systems containing intricate task execution and temporal dependencies. Specifically, the work presented in this paper includes three contributions:

1) To the best of our knowledge, this is the first approach to RTA which does not require any knowledge of the task execution times. Our method uses the measured task response time sample satisfying the independent and identically distributed assumption, as the qualified training data.
2) We have shown how to compute an upper bound on the task worst-case response time estimate for a given task reliability requirement. This has been achieved with a sampling method, and a posterior statistical correction process that uses a selection of search algorithms and statistical techniques.
3) The analysis results are calibrated and tight, and the corresponding computing time has been significantly improved against an alternative state of the art work. This has been shown by our intensive evaluations using information from real industrial applications.

For future work, we will extend the proposed technique to deal with software systems running on multi cores or multiprocessors. We will explore the possibility of using our proposed approach for WCET analysis. Currently, the correlation between the tightness of the analysis results given by RapidRT* and the selected factors of the change on system CPU speed in our evaluation is not so obvious to observe. We may conduct some relevant investigation of this using other types of evaluation models.

Acknowledgements

The authors express their gratitude to the anonymous reviewers of this paper that have provided useful comments and suggestions which helped in improving the overall presentation and quality of the paper.

This work is supported by the Swedish Foundation For Strategic Research, the Swedish Research Council, and Mälardalen University.

References

[1] S. H. Son, I. Lee, and J. Y.-T. Leung, *Handbook of Real-Time and Embedded Systems*. Chapman and Hall/CRC, 2007.
[2] I. Bate and A. Burns, "An Integrated Approach to Scheduling in Safety-Critical Embedded Control Systems," *Real-Time Systems*, vol. 25, no. 1, pp. 5–37, 2003.
[3] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A Metaheuristic Approach for Best Effort Timing Analysis Targeting Complex Legacy Real-Time Systems," in *14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'08)*, April 2008, pp. 258–269.
[4] Y. Lu, "Pragmatic approaches for timing analysis of real-time embedded systems," Ph.D. dissertation, Mälardalen University Press Dissertations, June 2012.
[5] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, pp. 1–53, April 2008.
[6] A. Burns, G. Bernat, and I. Broster, "A Probabilistic Framework for Schedulability Analysis," in *3rd International Conference on Embedded Software (EMSOFT'03)*, October 2003, pp. 1–15.
[7] J. Mäki-Turja and M. Nolin, "Efficient implementation of tight response-times for tasks with offsets," *Real-Time Systems*, vol. 40, no. 1, pp. 77–116, October 2008.
[8] J. Mäki-Turja and M. Sjödin, "Response-Time Analysis for Transactions with Execution-Time Dependencies," in *19th International Conference on Real-Time and Network Systems (RTNS'11)*, September 2011, pp. 139–145.
[9] R. O. Baldwin, N. J. Davis IV, J. E. Kobza, and S. F. Midkiff, "Real-time queueing theory: A tutorial presentation with an admission control application," *Queueing Systems: Theory and Applications*, vol. 35, no. 1-4, pp. 1–21, 2000.
[10] A. K. Atlas and A. Bestavros, "Statistical rate monotonic scheduling," in *19th IEEE Real-Time Systems Symposium (RTSS'98)*, December 1998, pp. 123–132.
[11] CENELEC, *IEC 61508 Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, 2nd ed., 2010.
[12] "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, ARP4761, 2001."
[13] L. Cucu-Grosjean, "Probabilistic real-time schedulability analysis: from uniprocessor to multiprocessor when the execution times are uncertain," RR-INRIA, Tech. Rep., May 2009.
[14] J. M. López, J. L. Díaz, J. Entrialgo, and D. García, "Stochastic analysis of real-time systems under preemptive priority-driven scheduling," *Real-Time Systems*, pp. 180–207, November 2008.
[15] T. Nolte, A. Möller, and M. Nolin, "Using components to facilitate stochastic schedulability analysis," in *Work-In-Progress (WIP) Session of the 24th IEEE Real-Time Systems Symposium (RTSS'03)*, December 2003, pp. 7–10.
[16] J. Hansen, S. Hissam, and G. Moreno, "Statistical-Based WCET Estimation and Validation," in *9th International Workshop on Worst-Case Execution Time Analysis (WCET'09)*, June 2009, pp. 123–133.
[17] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quinones, and F. J. Cazorla, "Measurement-based probabilistic timing analysis for multi-path programs," in *24th Euromicro Conference on Real-Time Systems (ECRTS'12)*, July 2012, pp. 91–101.

[18] G. A. Kaczynski, L. Lo Bello, and T. Nolte, "Deriving Exact Stochastic Response Times of Periodic Tasks in Hybrid Priority-driven Soft Real-time Systems," in *12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07)*, September 2007, pp. 101–110.

[19] K. Refaat and P.-E. Hladik, "Efficient stochastic analysis of real-time systems via random sampling," in *22nd Euromicro Conference on Real-Time Systems (ECRTS'10)*, July 2010, pp. 175–183.

[20] Y. Lu, T. Nolte, J. Kraft, and C. Norström, "A Statistical Approach to Response-Time Analysis of Complex Embedded Real-Time Systems," in *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'10)*, August 2010, pp. 153–160.

[21] D. S. Moore, G. P. McCabe, and B. A. Craig, *Introduction to the practice of statistics*, 6th ed. New York, NY 10010: W. H. Freeman and Company, 2009.

[22] P. Friederichs, "An Introduction to Extreme Value Theory," COPS Summer School, July/August, 2007.

[23] K. Kim, J. L. Diaz, L. Lo Bello, J. M. Lopez, C.-G. Lee, and S. L. Min, "An Exact Stochastic Analysis of Priority-Driven Periodic Real-Time Systems and Its Approximations," *IEEE Transactions on Computers*, vol. 54, no. 11, pp. 1460–1466, 2005.

[24] M. Garrido and J. Diebolt, "The ET test, a goodness-of-fit test for the distribution tail," in *2nd International Conference on Mathematical Methods in Reliability (MMR'00) – Methodology and Practice*, July 2000, pp. 427–430.

[25] "Chi-squared test, www.enviroliteracy.org/pdf/materials/1210.pdf, 2012."

[26] S. Edgar and A. Burns, "Statistical Analysis of WCET for Scheduling," in *22nd IEEE Real-Time Systems Symposium (RTSS'01)*, December 2001, pp. 215–224.

[27] D. Griffin and A. Burns, "Realism in Statistical Analysis of Worst Case Execution Times," in *10th International Workshop on Worst-Case Execution Time Analysis (WCET'10)*, July 2010, pp. 49–57.

[28] J. Kraft, "Enabling timing analysis of complex embedded software systems," Ph.D. dissertation, Mälardalen University Press, August 2010.

[29] S. Zwillinger and D. Kokoska, *CRC Standard Probability and Statistics Tables and Formulae*. CRC Press, 2000.

[30] "Website of sampling theory," http://stattrek.com/Lesson3/ SamplingTheory.aspx.

[31] "How to Determine a Sample Size," http://extension.psu.edu/ evaluation/pdf/TS60.pdf.

[32] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte, "Simulation-Based Timing Analysis of Complex Real-Time Systems," in *15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'09)*, August 2009, pp. 321–328.

[33] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, Sep. 2001.

[34] S. Stigler, "Fisher and the 5% Level," *Journal of CHANCE*, vol. 21, no. 4, p. 12, 2008.

[35] A. M. Law and D. M. Kelton, *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 1999.

[36] "Resampling Stats Add-in for Excel User's Guide, Version 4," Resampling Stats, 2012.

[37] Y. Lu, "An Evaluation Framework for Complex Industrial Real-Time Embedded Systems," Mälardalen University, Technical Report, February 2012.

[38] J. Kraft, "RTSSim - A Simulation Framework for Complex Embedded Systems," Mälardalen University, Technical Report, March 2009.

[39] Y. Lu, T. Nolte, L. Cucu-Grosjean, and I. Bate, "RapidRT: A Tool For Statistical Response-Time Analysis of Complex Industrial Real-Time Embedded Systems," in *Real-Time Systems@Work, the Open Demo Session of Real-Time Techniques and Technologies of the 32nd IEEE Real-Time Systems Symposium (RTSS'11)*, November 2011.

[40] "Metropolis, Monte Carlo, and the MANIAC," http://library.lanl.gov/cgi-bin/getfile?00326886.pdf.