RTProb -Real Time Probabilsitic Tool for Probabilistic Schedulability Analysis using Markov Chain

Jasdeep Singh, Luca Santinelli, Guillaume Infantes, David Doose, Julien Brunel

RTProb - Real Time Probabilsitic Tool for Probabilistic Schedulability Analysis using Markov Chain

Jasdeep Singh, Luca Santinelli, Guillaume Infantes, David Doose and Julien Brunel
ONERA -DTIS Toulouse, name.surname@onera.fr

***Abstract*—This paper presents a probabilistic schedulability analysis tool for probabilistic Real-Time Systems (pRTS). By pRTS we intend a real-time system in which at least one of its parameter is defined using a probability distribution; in our case this parameter is the task Worst Case Execution Time (WCET) which is the probabilistic, probabilistic WCET (pWCET). The tool implements a formalism which is based on formal methods for modelling and analysis of pRTSs. It uses pWCETs to construct Continuous Time Markov Chain models, one per task job. For each job, the CTMC describes the job execution by taking into account all the interferences (probabilistic delays) that might exist. The tool also interface with model checkers for checking the models built. The results of the analysis as given by the tool are the probability of deadline miss and the response time curves for each task and for each job of the tasks.**

***Index Terms*—Probabilistic Real-Time System, pWCET, Markov Chain, Continuous Probability Distribution, Formal Methods**

I. Introduction

As the complexity of the real-time systems increases, accurate determination of the worst case execution (WCET) also becomes difficult. This is augmented by increasing usage of multi-core and commercial-off-the-shelf implementations [9]. The deterministic WCET contains large pessimism because actual execution times of the tasks may rarely be equal to the WCET. In order to quantify this pessimism, research is carried out on statistical methods to determine the execution times of the task [3], [1]. The result of such research is the notion of probabilistic worst case execution time (pWCET) that is a probability distribution which upper bounds all the possible execution times of a task [4]. The use of probabilities to model real-time tasks can potentially result to an efficient resource usage by reducing the pessimism involved in designing and guaranteeing a real-time system. A real-time system in which at least one parameter is represented by a probability distribution, and is not a deterministic value, is called a probabilistic real-time system (pRTS).

The schedulability analysis is performed on a given real-time system to ensure that all the timing constraints of jobs and tasks are met. As the jobs execute, they may delay the executions of other jobs, in turn missing a timing constraint. For deterministic schedulability analysis which uses the WCET, the value of these delays, also called backlogs, is relatively simpler to determine. However, since the execution of the jobs could be given as probability distributions, there exist probabilistic backlogs in the system. That means that there exists a probability that certain job imposes a backlog to other jobs. The probabilistic schedulability analysis takes into account such probabilistic backlogs to provide a probability for the system to meet its timing constraints.

The pWCET can be given as a continuous or discrete probability distribution. A continuous distribution gives the probability that the execution time takes a value within two limits. On the other, a discrete distribution gives the probability that the execution time takes certain discrete value. [3], [1] show that the result of measurement based probabilistic timing analysis (MBPTA) is a continuous distribution. Thus, the schedulability analysis which use these continuous distribution must be developed.

Some works are proposing probabilistic schedulability analyses and experience the complexity of combining probability distributions [8]. This is because there can be numerous probabilistic interactions to consider and offer guarantees. The complexity increases by a very large magnitude if the input distributions are continuous.

In this paper, formal methods are used to model pRTS where the pWCETs are described with continuous distributions. Formal methods have a mathematical foundation, and thus have a way to apply the underlying theorems for building the system model. This would help overcome the complexity with continuous distribution since mathematical constructions apply. Moreover, a model constructed using formal method can be subject to verification and model checking to obtain safe results a.k.a. pessimistic.

This paper presents an implementation with formal methods for schedulability analysis of pRTS using Continuous Time Markov Chain (CTMC). CTMC is a set of states and transitions labeled with parameters of continuous probability distribution. In particular, CTMC is labeled with rates of exponential distributions. CTMC possesses memorylessness property, i.e. to determine the future state, no knowledge of past is required and knowledge in the present state is enough. CTMC is able to model non-determinism (choice in the system) and probability (weight to the choices) and both of these aspects are necessary to model pRTS. The continuous pWCET distributions can be directly mapped onto CTMC state transitions, and the CTMC models can be formally checked.

The objective is to obtain the probability of deadline miss for the jobs in the system as well as their response time curves.

The implementation presented in this paper is named RTProb. It is based on the CTMC modelling of jobs in a pRTS from [11], [10]. The formalism uses the pWCET of the jobs and takes into account for the probabilistic delays that can exist between jobs in the system. RTProb builds CTMC models, interfaces with model checking, and computes the probability of deadline miss and response time for each job and each task.

Section II introduces the notations used in this paper and the assumptions for the probabilistic schedulability analysis proposed. Section III briefly explains the model behind the implementation. Section IV elaborates on the working of the tool. Section V concludes this paper with closing remarks and future work.

II. Notations and Assumptions

This section introduces the notations used and the assumptions made to apply CTMCs into the probabilistic schedulability analysis of pRTSs.

Given a continuous random variable $\mathcal{X}$ defined in $[0,+\infty)$

- **Probability Density Function (PDF):** $f_{\mathcal{X}}(x)$ of $\mathcal{X}$ gives the probability that a value extracted from $\mathcal{X}$ lies between a and b, $Pr(a \leq \mathcal{X} \leq b)$.
- **Cumulative Distribution Function (CDF):** $F_{\mathcal{X}}(x)$ of $\mathcal{X}$ gives the cumulative probability for $\mathcal{X} \leq x$.
- **Inverse Cumulative Distribution Function (ICDF):** $\overline{F}_{\mathcal{X}}(x)$ of $\mathcal{X}$ gives the exceeding threshold probability a x as the probability that $\mathcal{X} > x$.

The case in which the pWCET is represented by the exponential distribution with λ as the rate parameter, PDF: $f_{\mathcal{X}}(x) = \lambda e^{-\lambda x}$; CDF:$F_{\mathcal{X}}(x) = 1 - e^{-\lambda x}$; and ICDF $\overline{F}_{\mathcal{X}}(x) = e^{-\lambda x}$; all supported on the interval $(0,+\infty]$. The PDF is referred to as $EXP(\lambda)$.

Convolution: For two PDFs $f_{\mathcal{X}}(x)$ and $g_{\mathcal{Y}}(y)$, their convolution is denoted by $\otimes$, refers to the summation of the random variables $\mathcal{X}$ and $\mathcal{Y}$ generally given as: $f \otimes g(z)$ The convolution of more than two PDFs is represented as $\bigotimes_i \mathcal{C}_i$. Convolution operation is computationally costly. RTProb is conceived to reduce the use of convolution by observing a sequence that exists in the execution of jobs in the pRTS. This is elaborated in later sections. Convolution is still required for jobs arriving synchronously.

Task: A task τ_i is a tuple $\tau_i = \{\mathcal{C}_i, T_i, D_i\}$, $i = 1,2 \ldots m$ where

- $\mathcal{C}_i$ is the continuous PDF given by an exponential distribution with rate λ_i which represents the pWCET;
- T_i is the period;
- D_i is the task deadline such that $D_i \leq T_i$.

Job: These tasks execute periodically and the j-th periodic instance of a task is a job J_{ij}. A job is defined as $J_{ij} = \{\tau_i, d_{ij}, a_{ij}, p_{ij}\}$ where

- τ_i is the task to which the job belongs;
- $a_{ij} = (j-1)T_i$ is the job arrival time;
- $d_{ij} = jT_i$ is the job deadline;
- p_{ij} is the job priority, zero being the highest priority. The job priority gets assigned depending on the scheduling policy used.

Given is a task set of m tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_m\}$.
The hyperperiod $hp = lcm(T_i), \tau_i \in \Gamma$, $i = 1,2 \ldots m$, gives the scope of the schedulability analysis for EDF or FP.

Earliest Deadline First (EDF) or Fixed Priority (FP) [2] scheduling policy on a uniprocessor machine can be chosen. The policy is preemptive, i.e. arrival of a higher priority job can cause the already executing lower priority job to pause while the higher priority job finishes execution. The jobs are suspended if their execution reaches their respective deadline. This is to avoid theoretical problems given the pWCET is defined in $[0,\infty)$.

A. Assumptions

The CTMC formalisation requires some assumptions. In particular two, such that:

Assumption1: The pWCET distributions are continuous distributions. This is because, as already stated, the results of MBPTA approaches are continuous distributions. Moreover, we want to avoid converting from continuous to discrete, which could be complex and would require some knowledge on the system behavior: which are the discrete values to impose? [5] Continuous distributions are directly applied with CTMCs and operations between them benefit from CTMC mathematical background.

Assumption2: The pWCET distribution is assumed to be exponential $EXP(\lambda)$. This is because we want to interface CTMCs in which the transitions between the states of the CTMC are labelled with exponential rates. Not having an exponential distribution would increase the complexity because the model looses the memorylessness property. Moreover, imposing exponential distributions does not limit the applicability of the CTMC modelling to pRTSs, since it is always possible to find an exponential distribution that upper bounds a pWCET [10]. Formal and parametrized exponential upper bounding will be developed in future work. To note that measurement-based approaches [7], [3], [3], [1] estimate pWCET as distributions with exponential shapes. In those cases, the exponential distribution assumption does not even introduce further pessimism. Finally, we outline that it would always be possible to decompose a distribution into exponential elements and see this as job decomposition. Such decomposition will be dealt with in future studies.

III. Job Modeling

RTProb models and evaluates all the possible interferences that a job can receive and which can delay its execution, in turn increasing the probability of deadline miss. To each job J_{ij}, there are three ways in which its execution can be delayed:

Case1 - Preceding job. A job that precedes job J_{ij} in terms of priority is the set $\overline{J}^{prd}(J_{ij}) \stackrel{def}{=} \{J_{gh} : p_{gh} < p_{ij}, a_{gh} < a_{ij}\}$ and $p_{gh} - p_{ij}$ is minimum implying the previous job not released synchronously. The preceding job (the one before J_{ij}) is the only job giving backlog to the victim job. This is because the

process of analysis is sequential in the order of decreasing job-priority. Thus, the cardinality[1] of $\bar{J}^{prd}(J_{ij})$ is always one, $card(\bar{J}^{prd}(J_{ij})) = 1$; the set representation is for a general notation. The pWCET and arrival time of job $J_{gh} \in \bar{J}^{prd}(J_{ij})$ are represented as $C[\bar{J}^{prd}(J_{ij})] = \mathcal{C}_{gh}$ and $a[\bar{J}^{prd}(J_{ij})] = a_{gh}$, respectively.

Case2 - Synchronous job. A set of jobs arriving synchronously to J_{ij} is $\bar{J}^{syc}(J_{ij}) \stackrel{def}{=} \{J_{gh} : a_{gh} = a_{ij}, p_{gh} < p_{ij}\}$. The total push to the job J_{ij} by the jobs in $\bar{J}^{syc}(J_{ij})$ is given by the convolution of the pWCETs of all the jobs in the set $C^{syn}(J_{ij})$. The pWCET and arrival time of job $J_{gh} \in \bar{J}^{syc}(J_{ij})$ are represented as $C[\bar{J}^{syc}(J_{ij})] = \mathcal{C}_{gh}$ and $a[\bar{J}^{syc}(J_{ij})] = a_{gh}$, respectively.

Case3 - Preempting job. A set of preempting jobs is defined as $\bar{J}^{prm}(J_{ij}) \stackrel{def}{=} \{J_{gh} : a_{gh} > a_{ij}, p_{gh} < p_{ij}, a_{gh} < d_{ij}\}$. $\bar{J}^{prm}(J_{ij})$ is ordered in increasing arrival times of its constituent jobs. A $k-th$ job J_{gh} of $\bar{J}^{prm}(J_{ij})$ is represented as $J[k, \bar{J}^{prm}(J_{ij})] = J_{gh}$, with pWCET and arrival time as $C[k, \bar{J}^{prm}(J_{ij})] = \mathcal{C}_{gh}$ and $a[k, \bar{J}^{prm}(J_{ij})] = a_{gh}$ respectively. K_{ij} is the maximum number of preemptions J_{ij} can have and is given as $K_{ij} = card(\bar{J}^{prm}(J_{ij}))$.

These job classifications are depicted in Figure 1 for each job J_{ij}. The job executions are represented in the ICDF form to differentiate the case of pWCETs from deterministic WCETs. In here, the worst-case execution is described with a random variable, and the ICDF captures the distribution law as well as the probabilistic behaviour that jobs follow.

To each interference, we have develop upper bounds to represent them and interface with CTMCs. Some details are given in [10], more will come in future work.

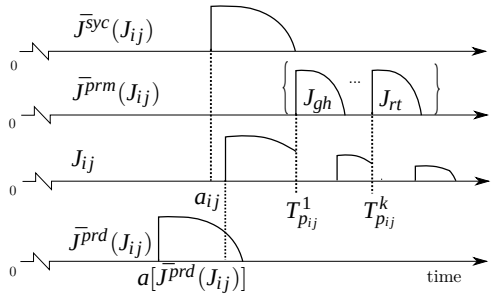


Fig. 1: Job J_{ij}, job set $\bar{J}^{prd}(J_{ij})$, job set $\bar{J}^{syc}(J_{ij})$, and job set $\bar{J}^{prm}(J_{ij})$ are represented with interactions between them.

A. CTMC

Continuous Time Markov Chain is a set of states and transitions between them with each transition labelled with an exponential rate λ. The CTMC possesses the memorylessness property. The set of transitions between the states is represented through a Q-matrix, which describes the transition for each couple of states as exponential distribution. The exponential rates represent the execution of a job before and after any preemption by considering all the possible interferences from higher priority jobs until the end of execution.

For a job J_{ij}, a set of states are defined as $X_{ij} = \{P_0, P_1, \ldots, P_{K_{ij}}, F\}$, in which state P_0 represents execution without preemption (after the eventual initial postponement), P_k represents execution after k-th preemption, and F represents the end of execution; J_{ij} suffers K_{ij} number of preemptions. The Q-matrix of size $[(K_{ij}+2),(K_{ij}+2)]$ is given as:

$$\begin{bmatrix} & P_0 & P_1 & \ldots & P_K & F \\ P_0 & -(\lambda_{p0}+\lambda_{f0}) & 0 & \ldots & 0 & \lambda_{f0} \\ P_1 & 0 & -(\lambda_{p1}+\lambda_{f1}) & \ldots & 0 & \lambda_{f1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P_K & 0 & 0 & \ldots & -\lambda_{fK} & \lambda_{fK} \\ F & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}^{[(K_{ij}+2),(K_{ij}+2)]},$$

where the subscript of λ denotes i) the final state, f if it goes to state F or k if it goes to P_k state, and ii) the k-th state from which it goes out. For example, λ_{f2} denotes the rate of transition from state P_2 to state F.

In the process to calculate the rates of Q-matrix, the backlog from $\bar{J}^{prd}(J_{ij}), \bar{J}^{syc}(J_{ij}), \bar{J}^{prm}(J_{ij})$ is used. Job J_{ij} CTMC model is $\{X_{ij}, Q_{ij}\}$, while the set of CTMC models of all the jobs in the hyperperiod $\{\{X_{ij}\}, \{Q_{ij}\}\}$ defines a pRTS.

Formal model checking is performed on the CTMC models to validate and build them. We have developed an iterative process adding a new preemption state and transitions per iteration, and the preemption property is validated via model checking. The preemption property is: 'maximum probability that the job is preempted at arrival t of the preempting job'. The response time distribution $\mathcal{RT}_{ij}$ for each job J_{ij} is computed by checking the CTMC against the property: 'maximum probability that the job ends execution by a time t', $0 \leq t \leq D_i$. The case where $t > D_i$, the schedulability analysis gives the probability of deadline miss DM_{ij} for a job J_{ij}. We remind that the response time distribution computed with CTMC is the probabilistic Worst-Case Response Time (pWCET) as the probability distribution that upper bounds any possible job response time.

Figure 2 presents the CTMC model for a generic job. The model has all the states that could occur: execution P_0, execution after a first preemption P_1, etc.. In total, it has K preemptions, and thus $K+2$ states; the state transitions are represented with exponential distributions and rates λ. Figure 3 joins all the CTMC job models in the hyperperiod ordered by priority: for each job there is a CTMC model associated. These elements are ordered in the sequence of decreasing priority of the jobs. The backlog from one model is propagated to the next.

The CTMC models are used to represent the jobs in the schedule. The pWCRT and deadline miss probability are computed from them. The whole process of building CTMC models and compute pWCRTs as well as deadline miss probabilities is what we call probabilistic schedulability analysis.

The overall approach is safe because (i) the pWCET is used for each job, which represents worst case execution; (ii) the analysis takes into account the worst cases (upper bounds) for backlogs which delay the execution of a job; (iii) formal model checking is performed at every step.

[1]Given a set S, the cardinality of S is represented as $card(S)$ which gives the number of elements in S

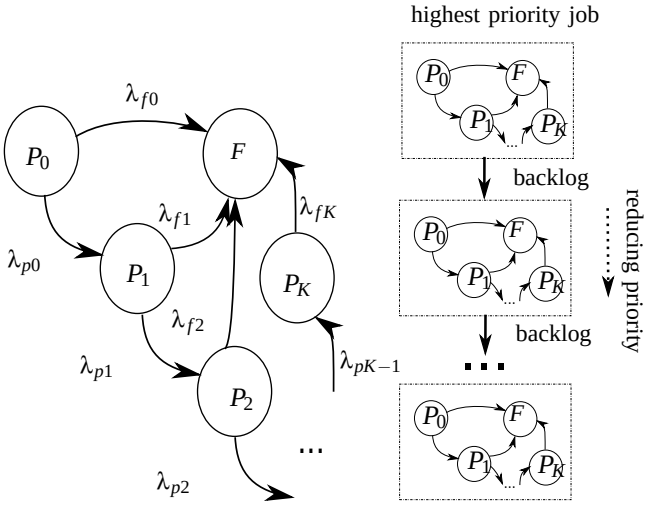


Fig. 2: Job CTMC model; the pre-emption effects are added and validated.

Fig. 3: Joining CTMC job models with backlog propagation.

IV. RTPROB TOOL

RTProb is the tool which implements the formalism briefly explained above. It is an implementation in python language and is available at https://forge.onera.fr/projects/probscheduling, the repository: https://git.onera.fr/probscheduling. The user must have Python 3.0 with packages `NumPy`, `os`, `time` and `pyplot`.

A task set is given as input to RTProb, with the rate of exponential which represents the pWCET and the deadline (equal to period). The task periods are used to determine the hyperperiod which is the scope of the analysis. The jobs are determined from the tasks, the number of jobs in the hyperperiod are calculated from the given tasks. Following this, the scheduling policy, EDF or FP, assigns priorities to the jobs, forming a list of jobs in the order of their decreasing priority.

The first step is to determine the backlog for each job. To do so, the sets $\bar{J}^{prd}(J_{ij})$, $\bar{J}^{syc}(J_{ij})$ and $\bar{J}^{prm}(J_{ij})$ are identified from the jobs list for each job J_{ij}. The job sets are the preceding job in terms of priority (Case1), the synchronously released jobs (Case2), and the preempting jobs (Case3), respectively,[10].

The Figure 4 shows the overall working of the RTProb tool. The block RTprob represents the implementation we have made. Because PRISM model checker is used, an important step consists of a PRISM script for CTMC model of each job. The constant interaction between the PRISM model checker and RTProb is made through a PRISM script of the CTMC model and a property to be checked. PRISM in returns provides the probability value for the property that is checked. To develop the script, certain information is required from the CTMC model of other jobs like backlog.

Once all the jobs are modelled, they are checked to obtain the probability of deadline miss in the block 'Analysis'. The block 'Response Time' performs this model checking in the PRISM model checker. The Algorithm 1 summarizes the schedulability analysis process.

procedure MODEL_ANALYSE_PRTS(*tasks, policy*)
  Order_Jobs(*Jobs,policy*) ▷ Order *Jobs* by their increasing priority
  for each job in *Jobs* **do**
    Define $\bar{J}^{pre}(J_{ij})$, $\bar{J}^{syc}(J_{ij})$, $\bar{J}^{prm}(J_{ij})$, $\bar{T}_p(J_{ij})$ ▷ The higher priority jobs sets
    Declare $X_{ij}=\{P_0,F\}; Q_{ij}=\{0,0;0,1\}$ ▷ Initial CTMC
    λ_{f0} =Backlog$(J_{ij},\bar{J}^{pre}(J_{ij}),\bar{J}^{syc}(J_{ij}))$ ▷ Backlog effects
    $Q_{ij}=\{-\lambda_{f0},\lambda_{f0};0,1\}$
    for each preemptive job $J[k,\bar{J}^{prm}(J_{ij})]$ in $\bar{J}^{pre}(J_{ij})$ **do** ▷ Preemption effects
      $Pr=P(J_{ij},P_k,t_p^k)$ ▷ k-th preemption
      Compute_rates $(\lambda_{fk},\lambda_{pk},Pr)$
      λ_{fk+1} = Delta_Pre$(J_{ij},J[k,\bar{J}^{prm}(J_{ij})])$
      Update(X_{ij},Q_{ij},k)
    $Pr(DM_{ij})=1-\mathcal{P}(J_{ij},F,t_{p_{ij}}^{K+1})$ ▷ Probability of deadline miss
    for time t **do**
      $F_{RT_{ij}}(t)$ =PRISM_Verify(J_{ij},F,t) ▷ Function of response time curve
  for each task in *tasks* **do**
    $Pr(DM_i)=max(Pr(DM_{ij}))$
    $F_{RT_i}(t)=max(F_{RT_{ij}}(t))$

Backlog$(J_{ij},\bar{J}^{pre}(J_{ij}),\bar{J}^{syc}(J_{ij}))$: determines the backlog to the job (J_{ij}) depending on the sets $\bar{J}^{syc}(J_{ij})$, $\bar{J}^{pre}(J_{ij})$, $\bar{J}^{prm}(J_{ij})$. Delta_Pre$(J_{ij},J[k,\bar{J}^{prm}(J_{ij})])$ calculates the exponential rate for execution after preemption. Compute_rates$(\lambda_{fk},\lambda_{pk},Pr)$ computes the transitions rates for the new transitions after the addition of a new state. Update$((X_{ij},Q_{ij},k))$ updates the CTMC matrix by adding the new state and the corresponding rates at the appropriate positions. $P(job,state,time)$ returns the probability of job being in state *state* at time instance or time interval *time*. These functions constantly use PRISM model checker to obtain the probabilities.

A. PRISM Model Checker

PRISM model checker [6] is used throughout the modelling process. It is a tool for formal model checking and analysis of systems that posses random or probabilistic behaviour. For the following, whenever a model is checked in PRISM, the required property is typed and saved as text file. A system command is executed using the `os` package in python to execute PRISM by giving the text files of the model and the property to check. The result is saved in a text file by PRISM. This text file is scanned and the value after `Result:` is read which is the probability demanded from model checking.

1) PRISM Scripting: The process of building CTMC for each job begins from the highest priority job. A text file is created which contains the script for the CTMC model in the language of PRISM model checker. The PRISM script begins with the name of the formal method `ctmc`. This is followed by the name of the module `module M` in the next line.

State and state transitions. The number of states of the CTMC are declared. The number of states required is equal to the number of preemptions that the job has, plus two. The state variable for the script is `x`. Moreover, an initial state is required to be declared. For our modelling, we declare that a state `0` is the finishing state F and the executing states $P_0,P_1,\ldots$ are declared $1,2,..$, respectively. Since there are K_{ij} preemptions to the job J_{ij}, the script becomes: `x:[0..`$K_{ij}+2$`] init 1;`.

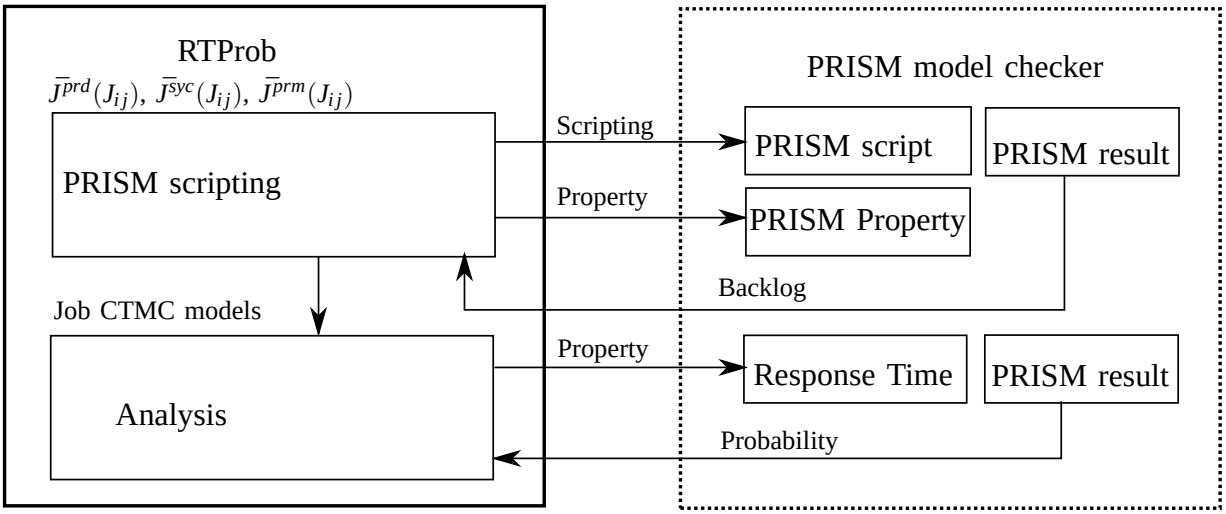


Fig. 4: The RTProb tool.

The transitions between the states are defined. The first rate of transition between the initial states P_0 and F (state 1 and 0) has to account for backlog from the previous high priority jobs. These are the higher priority jobs which have arrived earlier and have not yet finished execution and/or the jobs that have arrived synchronously. These are defined in the sets $\bar{J}^{prd}(J_{ij})$ and $\bar{J}^{syc}(J_{ij})$, respectively.

For the job which is arrived earlier (Case1), the CTMC model representing that job is checked to determine the probabilistic backlog. This is the probability that the earlier arrived job is executing when the job under observation has arrived. It is always possible to find this job because the CTMC modelling is performed in the sequence of decreasing priority. Moreover, for the synchronously arrived jobs (Case2), the convolution operation is performed. The probabilistic backlog and the convolution are combined to determine the safe exponential upper bound distribution. The rate of this exponential upper bound is the label of the transition between the states P_0 and F. Say this rate is λ^*_{f0}, the PRISM statement in the script is `[] x=1 ->` λ^*_{f0}`:(x'=0);`, which means that from the state 1 (P_0), the next state is state 0 (F) and the rate of the transition is λ^*_{f0}.

If there are preemptions to the job (Case3), a new state P_1 is added; it is `x=2`. This adds two new transitions to the existing CTMC. The transitions from P_0 to F and P_1 to F are calculated by splitting the rate λ^*_{f0} into λ_{p0} and λ_{f0}. This is done by checking the latest CTMC model (which is in construction) to obtain the probability that the job not finished and will move to the new state P_1. The rate of transition λ_{f1} from P_1 to F is calculated by checking the CTMC model of the preempting job. Here also, the CTMC model of the preempting job is available because it has higher priority. Thus the previous statement in the PRISM script changes to `[] x=1 ->` λ_{p0}`:(x'=2) +` λ_{f0}`:(x'=0) ;`. A new statement is added to model the state P_1 (x=2) as `[] x=2 ->` λ_{f1}`:(x'=0);`.

An example PRISM script for a job is:

```
ctmc
module M
x:[0..2] init 1;
[] x=1 -> 0.0318:(x'=2) + 5.039:(x'=0) ;
[] x=2 -> 10.133:(x'=0);
endmodule
```

The process formerly listed continues until the last state $P_{K_{ij}}$ is added where K_{ij} is the number of preemptions for the job J_{ij}. After the addition of the last states and transitions, the CTMC model for the job is complete. This process is repeating for all the jobs in the hyperperiod.

B. Analysis

Once the CTMC model is available for each job, the set of models is analysed to extract the value of probability of deadline miss and the response time distributionss.

The probability of deadline miss for a job is the probability that it does not finish execution by the time it reaches the deadline. In order to know the probability of deadline miss for the job, the CTMC model is checked using property 'the probability that the state F (finished) is not reached by the deadline', in the format of the PRISM model checker, `1-P=? [F=deadline x=0]` (state x=0 is F – finished execution).

The response time distribution of a job gives the probability that the job finishes execution by some time t. The same property as before can be checked for different times which gives the response time distribution for the job. That is, CTMC model for a job can be checked using a property that demands 'the probability that the state F is reached by time t'. In the format of the PRISM model checker, `P=? [F<=t x=0]` (state x=0 is F) and $0 \leq t \leq$ deadline.

An example task set has two tasks τ_1 and τ_2 with period 1 and 2 and pWCET with exponential rates 5 and 6 respectively. In the tool they are declared as `task.append(tasks.Task(5,1))`, `task.append(tasks.Task(6,2))` with `policy='EDF'`. There are two jobs of τ_1, J_{11} and J_{12}, and one job for τ_2, J_{21}. The PRISM script for J_{11} is:

```
ctmc
```

```
module M
x:[0..1] init 1;
[] x=1 -> 5.0:(x'=0) ;
endmodule
```

that of J_{12} is:

```
ctmc
module M
x:[0..1] init 1;
[] x=1 -> 4.933385407918634:(x'=0) ;
endmodule
```

and that of J_{21} is:

```
ctmc
module M
x:[0..1] init 1;
[] x=1 -> 3.954772344013123:(x'=0) ;
endmodule
```

The probability of deadline miss for each job is produced as:

```
Task: 1 Job: 1 = 0.0067379726200286205
Task: 2 Job: 1 = 0.00036725651068525433
Task: 1 Job: 2 = 0.007202100327258765
```

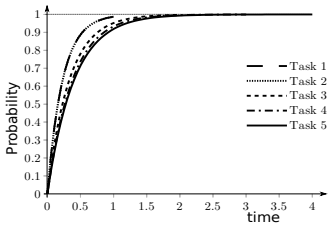


Fig. 5: EDF: response time for all the tasks.

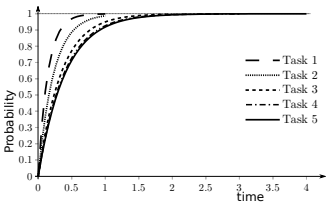


Fig. 6: FP: response time for all the tasks.

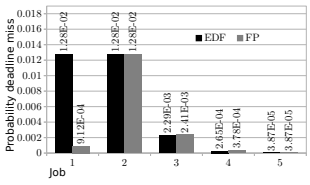


Fig. 7: EDF vs FP: all the tasks.

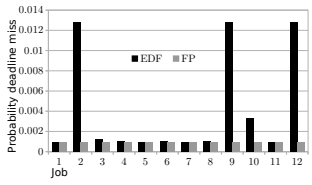


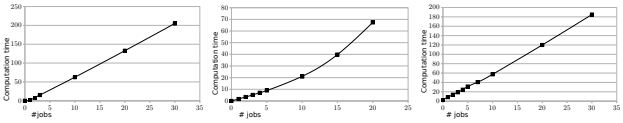
Fig. 8: Jobs of task τ_1

Figure 5 and Figure 6 show the response time distributions computed for a second example task set with 5 probabilistic tasks. There are compared EDF and FP scheduling. The times are in cycles. Figure 7 illustrates the deadline miss probability of all the 5 tasks in example. EDF and FP are compared in the same plot. Figure 8 details the deadline miss probability of task τ_1; all its 12 jobs deadline miss probability are presented.

C. Complexity

Given a task set Γ with m jobs in its hyperperiod, d being the discretisation unit of the distribution for numerical convolution, and D being the largest deadline a job, the asymptotic complexity of the CTMC modelling is $O((m(m+1)/2).(D/d)^2)$.

Figure 9 shows computational complexity as the time taken by the tool to compute probability of deadline miss for all the jobs as the number of jobs increase and the type of backlog changes. Figure 9(b) shows the time taken increases exponentially as the number of synchronously released jobs increase. This is because of the convolution operation required to compute the total execution of the synchronous jobs. Figures 9(a) and 9(c) show that time taken increases linearly as the number of non synchronous jobs increase and the number of preemptions increase.



(a) Computation time for the backlog from previous job

(b) Computation time for the backlog from synchronous jobs

(c) Computation time for the interference from preempting jobs

Fig. 9: Computational complexity for different interference scenarios; in the ordinate the computation time in seconds, in the abscisse the number of jobs that interfere.

V. Conclusion

RTProb has been here briefly described. It performs probabilistic schedulability analysis of pRTSs in which task execution is described with pWCET. RTProb implements a formalism based on formal method CMTC modelling of the jobs of each task. The working of the tool is presented which involves interactions with the probabilistic formal model checker PRISM. The tool is currently applied on a variety of projects.

Future works will be in the direction of removing some of the assumptions. In particular, the assumption that pWCET is an exponential distribution will be removed. A hybrid modelling method will be proposed which is flexible to the type of the input distribution (continuous or discrete) and other execution conditions.

References

[1] Jaume Abella, Eduardo Quiñones, Franck Wartel, Tullio Vardanega, and Francisco J. Cazorla. Heart of gold: Making the improbable happen to increase confidence in MBPTA. In *26th Euromicro Conference on Real-Time Systems, (ECRTS)*, 2014.

[2] Giorgio C. Buttazzo. Rate monotonic vs. edf: Judgment day. *Real-Time Systems*, 2005.

[3] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzeti, E. Quinones, and F. J. Cazorla. Measurement-Based Probabilistic Timing Analysis for Multi-path Programs. In *Euromicro Conf. on Real-Time Systems (ECRTS)*. IEEE, 2012.

[4] Robert I. Davis, Alan Burns, and David Griffin. On the meaning of pwcet distributions and their use in schedulability analysis. In *In Proceedings Real-Time Scheduling Open Problems Seminar at ECRTS*, 2017.

[5] David Griffin and Alan Burns. Realism in statistical analysis of worst case execution times. In *10th International Workshop on Worst-Case Execution Time Analysis (WCET)*, pages 44–53, 2010.

[6] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[7] George Lima, Dario Dias, and Edna Barros. Extreme value theory for estimating task execution time bounds: A careful look. In *28th Euromicro Conference on Real-Time Systems, (ECRTS)*, 2016.

[8] D. Maxim and L. Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *IEEE Real-Time Systems Symposium*, 2013.

[9] Vincent Nélis, Patrick Meumeu Yomsi, Luís Miguel Pinho, José Fonseca, Marko Bertogna, Eduardo Quiñones, Roberto Vargas, and Andrea Marongiu. The challenge of time-predictability in modern many-core architectures. In *14th International Workshop on Worst-Case Execution Time Analysis*, 2014.

[10] Jasdeep Singh, Luca Santinelli, and Guillaume Infantes. Backlog estimation for suspended probabilistic tasks released synchronously. In *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2017.

[11] Jasdeep Singh, Luca Santinelli, Guillaume Infantes, David Doose, and Julien Brunel. Markov chain modeling of probabilistic real-time systems. In *The 8th Real-Time Scheduling Open Problems Seminar at ECRTS*, 2017.