Probabilistic Uni-processor Schedulability Analysis

N. Nissanke[†], L. David[‡] and F. Cottet[*]

[†] Institute for Computing Research
London South Bank University
103, Borough Road, London SE1 0AA, U.K.

[‡] Campus Universitaire de Beaulieu
Avenue du Gnral Leclerc
35042 Rennes Cedex, France

[*] LISI–ENSMA
1 Av. Clement Ader, Teleport 2
86961 Futuroscope, France

***Abstract*— The paper presents a probabilistic approach to schedulability analysis of uni-processor priority–driven (e.g. RM and EDF) preemptive periodic task systems with uncertain computation times. The approach is a general one but is targeted at Quality of Service (QoS) driven non–critical real–time applications. Execution patterns and termination times of tasks are derived as timed sequence of probabilities, allowing the calculation of a range of QoS characteristics such as jitter, latency and loss rate. An example and a stochastic simulation illustrate the analytical framework and its validity.**

I. Introduction

When dealing with QoS driven applications, such as modern multimedia applications, the worst case design approach suffers from a major limitation, namely, uneconomic resource utilisation. This is evident from the growing number of works devoted to alternative approaches; some of the key contributions being [1], [2], [3] and [5]. Statistical Rate Monotonic Scheduling (SRMS) [1] attempts a generalisation of the classical rate monotonic scheduling discipline by describing computation times probabilistically and by maintaining schedulability through admission control. Based on a series of stochastic processes called task graphs, Manolache et al. [5] present an efficient approach to performance analysis of periodic non–preemptable tasks capturing uncertainty in task execution times in terms of a continuous probability distribution function. Diaz et al. [2] propose a stochastic approach based on Markov processes for scheduling periodic tasks with uncertainties in their computation times using both static and dynamic scheduling algorithms. Stochastic Time Demand Analysis (STDA) [3] concerns the determination of a lower bound on execution rates in uni–processor fixed priority context with periodic tasks having both a guaranteed execution time and a guaranteed inter–release time. Real–Time Queueing Theory [4] is also a widely known approach, though queueing theoretic models are considered in [6] to be “either too simple to characterise the important properties ... or too complex for tractable analysis”.

Based on standard probabilistic concepts, this work proposes a novel approach to uni-processor preemptive scheduling of periodic task systems with uncertainties in computation times. Working in a discrete time model, it uses the concept of *timed sequence of probabilities* (TSP) – a crucially important representation for dealing with uncertainties in all time dependent events, e.g., uncertainties in task execution times and task termination probabilities at each and every time instant. The latter allows the calculation of a range of QoS performance indicators (e.g. jitter, response time latency; not given here for brevity) of tasks individually in a straightforward manner. How difficult such calculations are, or their feasibility, is not clear in the works cited earlier.

Section II introduces the probabilistic representation of uncertainties in execution time and the concept of TSP. Section III presents the proposed approach, while Section IV presents an illustrative example and a stochastic simulation as a verification. Section V concludes the paper with a summary of achievements.

II. Representation of Task Uncertainties

Given a set of n periodic tasks, τ_i, $i \in 1\,..\,n$, let their computation times and periods be, respectively, C_i and T_i in a discrete time domain. τ_i^j denotes τ_i's jth instance. Tasks being periodic, each T_i is a fixed quantity. Relative task deadlines are assumed to lie within the periods and, as a simplification here, each deadline to coincide with the next task request. Uncertainty in computation times is taken care of by letting each C_i to be a random variable characterised by a probability mass function (PMF) P_{C_i}. Each P_{C_i} is defined over $K_i + 1$ points, denoted by $c_{i,k}$, $k \in 0\,..\,K_i$. By definition, a) the values $c_{i,k}$ are assumed to be in the ascending order with increasing k, but with the restriction that $c_{i,K_i} \leqslant T_i$, b) $c_{i,0} = 0$ and $P_{C_i}(C_i = c_{i,0}) = 0$, and c) $P_{C_i}(C_i = c_{i,k}) \neq 0$ for $k \in 1\,..\,K_i$. Assumptions (b) and (c) are intended at producing a minimal set of mass values in the initial specification of uncertainties, while excluding zero as a possible computation time. A different way to represent a PMF is as a sequence of 'descending blocks', as opposed to a 'rectangular block' of computation time as used in deterministic scheduling, see Figure 1 and Table I.

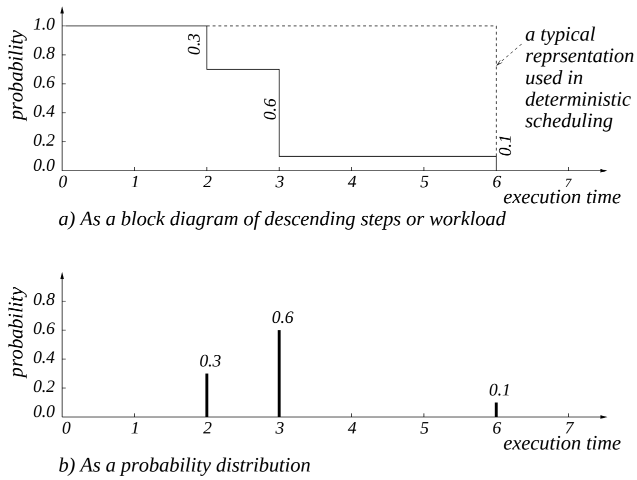

Fig. 1. Probabilistic representation of task execution requirements

Distribution functions				Workload as a TSP	
	Comp. time	Probability mass function	Survivor function $\mathcal{F}_{C_i} =$	Processor time	w_t^i – probability of τ_i requiring more than t time units
k	$c_{i,k}$	$P(C_i = c_{i,k})$	$P(C_i > c_{i,k})$	t	
0	0	0.0	1.0	0	1.0
				1	1.0
1	2	0.3	0.7	2	0.7
2	3	0.6	0.1	3	0.1
				4	0.1
				5	0.1
3	6	0.1	0.0	6	0.0

TABLE I
PROBABILISTIC REPRESENTATION OF UNCERTAINTIES IN TASK COMPUTATION TIMES

Table I also shows the representation of computation times used in the analysis, referred to as a *timed sequence of probabilities* (TSP) and having an obvious relationship with the survivor function $\mathcal{F}_{C_i}$; $\mathcal{F}_{C_i} = P(C_i > c_i)$ and c_i being a sample point of C_i. Each of its elements w_t^i is indexed by a time value t, with t ranging contiguously from $c_{i,0}$ to c_{i,K_i}. w_t^i denotes the probability of the task τ_i requiring more than t time units of the processor for its execution and the TSP is termed the '*workload*' placed by τ_i on the processor. For the example above, its TSP may also be shown as $\langle 1.0, 1.0, 0.7, 0.1, 0.1, 0.1, 0.0 \rangle$, assuming a starting time value 0 and implicitly indexing each of the probabilities in the sequence consecutively with a time value. TSPs are suitable for representing uncertainty in different kinds of transient data and not just workloads. In such cases, it is necessary to specify the starting time value of the TSP concerned so that its elements can be appropriately indexed.

III. Schedulability Analysis Algorithm

Uncertainties in task computation times give rise to uncertainties in the availability of the processor for the execution of particular tasks. In a preemptive priority driven execution process, at a given instant in time the processor is available for the execution of the highest priority task with certainty, i.e. with a probability of one. In a fixed priority context, such as RM scheduling, the highest priority task continually enjoys this privilege right through the execution history. This would not necessarily be the case in a dynamic scheduling context such as EDF because of the fluctuation of priorities over time. In the case of lower priority tasks, the probability of processor being available for their execution progressively decreases up to zero with decreasing priorities.

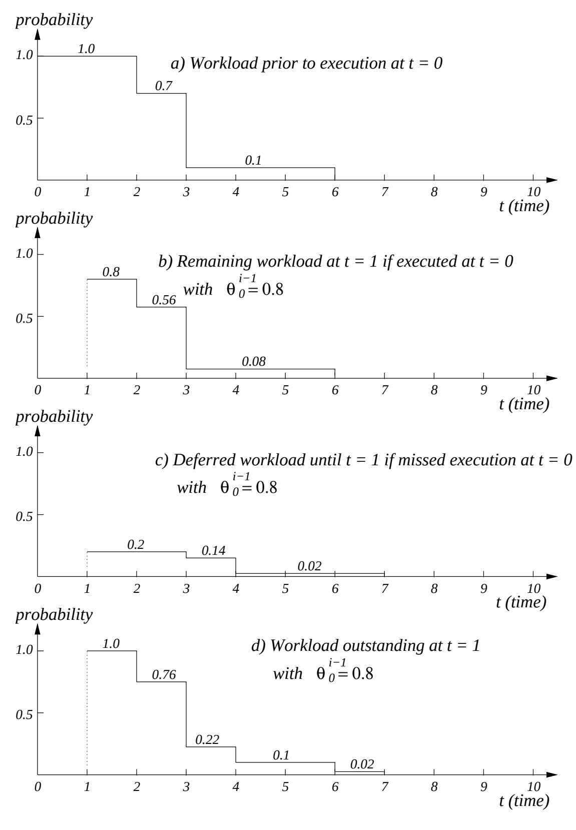

Fig. 2. Illustration of task execution (τ_i with $\theta_0^{i-1} = 0.8$ at $t = 0$)

For the length of time of scheduling, let us consider the Least Common Multiple (LCM) $\mathcal{L}$ of the task periods, though a longer length would be required if the model were to be extended to include deadlines greater than the periods. Let θ_t^j, $j \in 0\,..\,(n-1)$, denote the probability of the processor being available at time t for the execution of the task τ_{j+1} after execution of all its higher priority tasks, if any. The values θ_t^j form a TSP of the form $\Theta^j =$

$\langle \theta_t^j \mid t \in 0\,.\,.\mathcal{L} \rangle$ with a starting time value of 0. The TSP Θ^j is referred to here as *processor availability probabilities* (PAP). For $j > 0$, the actual value of each Θ^j is dependent on the actual workloads of the tasks $\tau_1, \tau_2, .\,., \tau_j$. When $j = 0$, however, Θ^0 has a simple expression, namely, a TSP with all 1s, i.e. $\Theta^0 = \langle 1.0, 1.0, .\,., 1.0 \rangle$. This is because at any time unit, if it requires, the highest priority task has access to the processor with certainty. Determination of other Θ^js is dealt with in (1) later.

In fixed priority scheduling, it is assumed that the task indices (i in τ_i) reflect the priority ordering. Thus, in RM scheduling tasks are ordered in the ascending order of task periods. Let us first consider fixed priority scheduling (e.g. RM). Dynamic scheduling (e.g. EDF) can then be dealt with a simple rearrangement of the order of the computations involved.

Turning to execution histories of individual tasks, let $e_t^{i,k}$ denote the probability of τ_i^k being under execution at time t and e_t^i the probability of τ_i as a whole being under execution at t. The corresponding TSPs e^i and $e^{i,k}$ are referred to as *task execution probabilities* (TEP). The derivation of $e^{i,k}$s may be explained as follows. Figure 2 illustrates the scheduling of an instance of a certain task τ_i with a workload shown in Figure 2(a), for convenience, at time 0. Should the processor be available at time 0 for executing τ_i, its workload will be reduced by one unit of time. However, this would depend on the availability of the processor. Assuming a value of $\theta_0^{i-1} = 0.8$ at $t = 0$, the probability of τ_is execution at time 0, that is, e_0^i, would be 0.8. The remaining workload is then given by the original workload less its first unit of time, but scaled down as a whole by a factor of 0.8; see Figure 2(b). This remaining workload will then be carried forward for execution from time 1 onward. However, there is also the possibility of τ_i missing execution at time 0, with a probability of 0.2. Should this happen, the original workload as a whole will be shifted by one unit time forward in time, but scaled down by a factor of 0.2; see Figure 2(c). However, τ_is execution at time 0 is contingent upon the processor availability and, hence, neither of the scenarios concerning the remaining workload is absolute. Therefore, the workload outstanding at time 1 is the point by point summation, shown in Figure 2(d), of the workloads, shown in Figures 2(b) and (c). This process can be continued successively for time values 1, 2, 3, etc., thus working out e_1^i, e_2^i, e_3^i, etc. and, eventually, e^i as a whole. When scheduling each τ_i, this process needs to be continued for all time values in $\mathcal{L}$, appropriately renewing instances of various tasks τ_l, $l \in 1\,.\,.\,i$ as they are requested.

The above process can be expressed in a straightforward manner as an algorithm, which is not given here for reasons of space. Another important quantity to be computed is the termination probability $f_t^{i,k}$ of τ_i^k at time t; i.e. the probability of τ_i^k being successfully completed exactly at time t and not executing afterwards. It can be defined as the probability of τ_i^k being assigned the processor at time $(t-1)$ and at that time there remaining a workload of exactly one time unit. Since the value of each term $f_t^{i,k}$ is dependent on the workload of τ_i^k at the time t, $f^{i,k}$ needs to be computed at each time instant alongside $e^{i,k}$. The values $f_t^{i,k}$ also form a TSP, denoted as $f^{i,k}$ and referred to here as a timed sequence of *task termination probabilities* (TTP).

With the knowledge of TEPs of τ_i and all its higher priority tasks, it is possible to compute the PAP Θ^i as

$$\Theta^i = \Theta^0 - \sum_{l=1}^{i} e^l \qquad \text{for} \quad i \geqslant 0 \tag{1}$$

with Θ^0 consisting of all 1s and using generalised operators $+$ and $-$ on TSPs. The above is a theorem and can be proved by mathematical induction. Θ^i so computed is required in the calculation of TEP of τ_{i+1} – the task immediately below τ_i in the priority ranking.

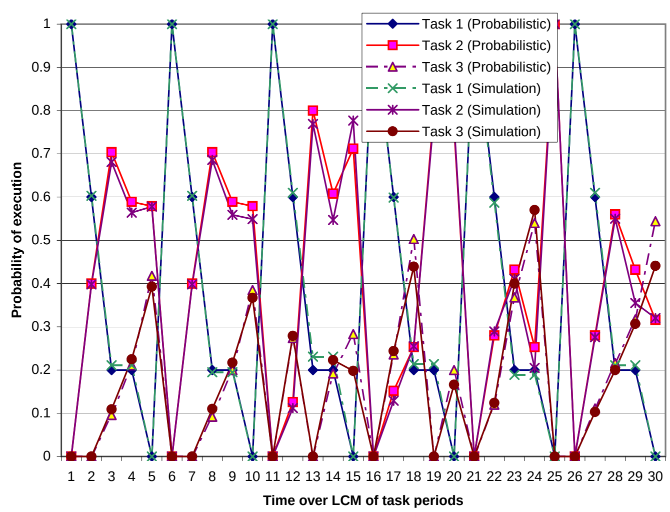

Fig. 3. Probability of task execution under RM scheduling over the simulation cycle: probabilistic analysis vs. simulation

The time complexity of the algorithm concerned for all tasks up to the nth priority task τ_n is given by $\frac{3}{2}\mathcal{L}\,\mathcal{C}\left(\sum_{i=1}^{n} T_i\right)$ and, therefore, is of the order $\mathcal{O}\left(\sum_{i=1}^{n} T_i\right)$ per unit time of $\mathcal{L}$, with $\mathcal{C}$ denoting an aggregate cost of the computational operations such as multiplication. Since task periods are usually greater than unity the sum

$\sum_{i=1}^{n} T_i$ grows faster than n. An insight into the computational complexity may be obtained by considering several special and extreme cases. For example, if the LCM of task periods is not affected by a new task τ_{n+1} then with its addition to the task set the complexity rises by T_{n+1}. If all tasks have the same period, resulting in $T_i = \mathcal{L}$, $i = 1 \,..\, n$, the complexity over the LCM is of $\mathcal{O}(n)$. On the other hand, if T_is conform to a geometric progression of the form $T_i = T\,q^{i-1}$, q being an integral constant, then the complexity is of $\mathcal{O}(q^{2n})$ over the LCM. These being extreme cases, the complexity in most practical situations lies somewhere in between.

$\tau_1(T_1 = 5)$		$\tau_2(T_2 = 6)$		$\tau_3(T_3 = 10)$	
Comp. time	PMF† (histogram‡)	Comp. time	PMF† (histogram‡)	Comp. time	PMF† (histogram‡)
1	0.4 (0.398)	1	0.3 (0.293)	2	0.3 (0.298)
2	0.4 (0.394)	3	0.5 (0.507)	4	0.6 (0.598)
4	0.2 (0.208)	5	0.2 (0.200)	6	0.1 (0.103)

† – used in in analysis ‡ – used in simulation (in parentheses)

TABLE II

PROBABILITIES OF TASK COMPUTATION TIMES (C_iS)

	τ_1^k		τ_2^k		τ_3^k	
k	RM	EDF	RM	EDF	RM	EDF
1	1 (1)	1 (1)	0.68 (0.69)	0.84 (0.78)	0.15 (0.19)	0.58 (0.55)
2	1 (1)	1 (1)	0.75 (0.73)	0.53 (0.51)	0.21 (0.24)	0.32 (0.35)
3	1 (1)	0.89 (0.86)	0.81 (0.79)	0.91 (0.91)	0.34 (0.41)	0.47 (0.54)
4	1 (1)	0.80 (0.80)	0.81 (0.76)	0.82 (0.83)		
5	1 (1)	0.78 (0.80)	0.84 (0.76)	1 (1)		
6	1 (1)	0.15 (0.22)				

analytical results: without parentheses; simulation: in parentheses

TABLE III

TERMINATION PROBABILITY OF TASK INSTANCES

In fixed priority scheduling, each task may be scheduled right through all time values of $\mathcal{L}$ and then schedule the next task in the priority order in the same manner. Turning to dynamic scheduling (e.g. EDF) all what is required is to consider schedulability of all tasks at each time value and then progress to the next time instant, obviously re-evaluating the priority order of the tasks at each time value. In the algorithm concerned, this corresponds to an alteration of the execution order of the computations (nesting order of loops) involved.

IV. ILLUSTRATIVE EXAMPLE

The example involves three tasks with characteristics given in Table II. Shown in parentheses are the histograms of the data used in a simulation over a period of 2,000 LCM cycles. As an illustration, the execution patterns of the tasks (TEPs) under RM is shown in Figure 3. The results (not all reproduced here for reasons of space) show that TEPs under both RM and EDF scheduling due to probabilistic analysis and simulation are sufficiently close. Similar results are obtained for task termination probabilities (TTPs) over the LCM under RM and EDF regimes; these are summarised in Table III giving the overall termination probabilities (success rate) of individual task instances within their periods.

V. CONCLUSIONS

This paper presents a probabilistic approach to analysing schedulability of periodic tasks with uncertain computation times in a uni–processor context under both RM and EDF scheduling regimes. Uncertainties in computation times are specified as PMFs but are represented as *timed sequence of probabilities* (TSP) – a general representation used for representing uncertainties in different events, including the execution pattern and termination of task instances at particular instants in time. The latter for each task are computed progressively by considering the probability of processor availability at each instant in time, which depends on the workloads and execution patterns of its higher priority tasks. Once both these TSPs are known, various QoS indicators may be computed, including the completion rate of tasks, their mean response time latency, failure rate, jitter, etc. (not given here for reasons of space). A major benefit of the proposed approach is that it allows addressing specific QoS indicators of tasks individually. Complexity analysis shows that the cost of computations involved grows additively with task periods and in proportion to the LCM of task periods. An example and the results of a stochastic simulation, conducted as a verification of the approach, demonstrate the approach and its validity.

Acknowledgement

The authors extend their gratitude to K. Gupta of the IIT, Guwahati, Assam, India, for his valuable comments.

REFERENCES

[1] A. Atlas and A. Bestavros. Statistical rate monotonic scheduling. In *19th IEEE Real-Time Systems Symposium*, Madrid, 1998.

[2] J. L. Diaz, D. F. Garcia, et al. Stochastic Analysis of Periodic Real-time Systems. In *23rd IEEE Real-Time Systems Symposium*, Austin, Texas, 2002.

[3] M. Gardner. Probabilistic Analysis and Scheduling of Critical Soft Real–Time Systems. Ph.D. Thesis. University of Illinois at Urbana–Champagn. 1999

[4] J. P. Lehoczky. Real–time Queueing Theory. Proceedings of the 17th *IEEE Real-Time Systems Symposium*. December 1996.

[5] S. Manolache, P. Eles and Z. Peng. Memory and Time-efficient Schedulability Analysis of Task Sets with Stochastic Execution Time. *13th Euromicro Conference on Real–Time Systems*. 2001.

[6] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks, (Invited paper). Proceedings of the IEEE. 83(10), 1995.