Performance Analysis of a Probability Model for Scheduling Hard Real-Time Tasks*

Wei Zhao

Computer and Information Science Department
University of Massachusetts

COINS Technical Report 86-17

Abstract

In this paper, we report the performance analysis for a probability model for scheduling hard real-time tasks. In the model, tasks arrive in system as a Poisson process with constant computation time and one of n alternative laxities which are less than the computation time. Based on this model, we derive the formulas of the guarantee ratio and CPU utilization.

*This material is based upon work supported in part by the National Science Foundation under grants DCR-8318776, and DCR-8500332, and by the Office of Naval Research under grant 048-716/3-22-85.

Performance Analysis of a Probability Model for Scheduling Hard Real-Time Tasks

Wei Zhao

1. Introduction

In this paper, we study the performance of a probability model for scheduling hard real-time tasks. In the model, tasks arrive in system as a Poisson process with constant computation time and one of n alternative laxities which are less than the computation time. The performance metrics we concern in this model are *guarantee ratio* and *CPU utilization*. The former is defined as the fraction of tasks which finish before their deadlines. In this model, we derive the explicit formulas of the guarantee ratio and CPU utilization.

In the remiander of this section, we define the probability model of hard real-time computation system for this study. Then we outline the analytic results of performance metrics such as CPU utilization and task guarantee ratio from the model. Section 2 proves the results formally, and Section 3 discusses the observations from the results and proposes the further work.

We propose that the hard real-time computation system is carried on a CPU and there is no other resource competition. There are c classes of tasks. Tasks in class k have laxity $= L_k$

and arrive at system as a Poission process with arrival rate = λ_k. Tasks are independent. Each task uses CPU C time units if it can be guaranteed. The scheduling time is not taken into account.

For the conveinece of discussion, we define

$$\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n, \tag{1}$$

and

$$L\lambda = L_1\lambda_1 + L_2\lambda_2 + \cdots + L_n\lambda_n. \tag{2}$$

Note that we take $L\lambda$ as one notation, not L times λ.

With this model, we are interested in two performance metrics: First, the CPU utilization, U, is given by

$$U = \lim_{t\to\infty} \frac{\text{CPU time being used by } t}{t}; \tag{3}$$

Secondly, the ratio of tasks being guaranteed, G, is

$$G = \lim_{t\to\infty} \frac{\text{Number of Tasks being Guaranteed by } t}{\text{Number of Tasks Arrived by } t}. \tag{4}$$

We can prove then

$$U = \frac{\lambda C}{\lambda C + e^{-L\lambda}} \tag{5}$$

and

$$G = \frac{1}{\lambda C + e^{-L\lambda}} \tag{6}$$

In the next section, we will formally prove (5) and (6).

2. The Proofs

2.1 Background

Here, we introduce background information from renewal theory. This information is necessary for the proofs in the latter subsections.

A system may have two states *on* and *off*. For example, in our system, we may define that in STATE *on*, the CPU is being used, and in state *off*, the CPU is idle. The system transfers from one state to another cyclically. That is, the system states are ..., *on, off, on,* .... Suppose that in a cycle i. e. two consequent *on* and *off* states, the duration in which the system is *on* (*off*) is a random variable On (Off) with identical independent distribution. Let P_{on} (P_{off}) be the proportion of time that the system is *on* (*off*), and let CL be the random variable for the whole length of a cycle, that is,

$$\text{CL} = \text{On} + \text{Off}, \tag{7}$$

then, from the renewal theory, we know

$$\begin{aligned} P_{\text{on}} &= \frac{E[\text{On}]}{E[\text{On} + \text{Off}]} \\ &= \frac{E[\text{On}]}{E[\text{CL}]}. \end{aligned} \tag{8}$$

and

$$P_{\text{off}} = \frac{E[\text{Off}]}{E[\text{On} + \text{Off}]}$$

$$= \frac{E[Off]}{E[CL]}. \tag{9}$$

2.2 CPU Utilization, U

In this subsection, we derive Formula (5) for the CPU utilization, U.

We observe that in our hard real-time computation system, the usage of CPU has the cyclic behavior, and has two states, *being used* and *being idle*. Let Used (Idle) be the random variable for the time of CPU being used (idle) in a cycle. When the tasks arrive as a Poission process, it is reasonable to assume that Used (Idle) has identical independent distribution. Then from (3) and (8), we obtain

$$U = \frac{E[\text{Used}]}{E[\text{Used} + \text{Idle}]} \tag{10}$$

Because tasks arrive as Poission process,

$$E[\text{Idle}] = E[\text{The time length in which there is no task arrival}]$$

$$= 1/\lambda \tag{11}$$

Let N be the random variable such that there are N tasks being executed in a cycle, then

$$\text{Used} = NC \tag{12}$$

where C is the computation time of tasks. Therefore,

$$\begin{aligned} E[\text{Used}] &= E[NC] \\ &= CE[N]. \end{aligned} \tag{13}$$

To compute $E[N]$, we need derive

$$P(N = n) = P\{\text{there are n tasks being executed in a cycle}\}. \tag{14}$$

Supoose that in a cycle totally n tasks run. Let t_0 be the time the first task begins run; t_i be the time the i-th task finishes and the $i+1$-th task starts run, $i = 1, \ldots, n-1$; and t_n be the time the n-th task finishes. The n tasks consequently. That is, as soon as one task finishes, another task starts run until the last (guaranteed) task finishes. Due to condition $L_k < C, k = 1, \ldots, c$, this implies that at least one task from any class must arrive between the time at which the i-th task finishes, t_i, and the time at which the new task will not lose its deadline. The latter depends on which class this new task come from: If the new task is from class k, then the time the new task will not lose its deadline is $t_i - L_k$. From the characteristic of Possion process, we know

$$\begin{aligned} P\{ \;\; & \text{at} \;\; \text{least one task from any class arrives between the time} \\ & \text{at} \;\; \text{which the i-th task finishes}, t_i, \text{and the time at which} \\ & \text{it} \;\; \text{will not lose its deadline}\} \end{aligned}$$

$$\begin{aligned}
&= 1 - P\{\text{No such a task arives}\} \\
&= 1 - \prod_{k=1}^{c} P\{\text{No such a class k task arrives}\} \\
&= 1 - \prod_{k=1}^{c} e^{-L_k \lambda_k} \\
&= 1 - e^{-L\lambda}. \qquad (15)
\end{aligned}$$

There must be no such a task during the time when the last task is running so that when the last task finishes, the system state changes from *on* to *off*. Simplly

$$\begin{aligned}
P\{ \; &\text{no} \;\; \text{task arrives during the time when the last task} \\
&\text{is} \;\; \text{running so that the new task will not lose its deadline}\} \\
&= e^{-L\lambda}. \qquad (16)
\end{aligned}$$

Now, we have

$$\begin{aligned}
P(n) &= P(N = n) \\
&= (1 - e^{-L\lambda})^{n-1} e^{-L\lambda}. \qquad (17)
\end{aligned}$$

Then,

$$\begin{aligned}
E[N] &= \sum_{n=1}^{\infty} nP(n) \\
&= \sum_{n=1}^{\infty} n(1 - e^{-L\lambda})^{n-1} e^{-L\lambda}
\end{aligned}$$

$$\begin{aligned} &= e^{-L\lambda} \sum_{n=1}^{\infty} n(1 - e^{-L\lambda})^{n-1} \\ &= e^{-L\lambda} \sum_{n=1}^{\infty} \frac{d}{dq} q^n \end{aligned} \tag{18}$$

where $q = (1 - e^{-L\lambda})$. Because $0 \leq q < 1$, the series is converged. Hence,

$$\begin{aligned} E[N] &= e^{-L\lambda} \frac{d}{dq} \sum_{n=1}^{\infty} q^n = e^{-L\lambda} \frac{d}{dq} \frac{1}{1-q} \\ &= e^{-L\lambda} \frac{1}{(1-q)^2} = e^{-L\lambda} \frac{1}{(e^{-L\lambda})^2} \\ &= e^{L\lambda}. \end{aligned} \tag{19}$$

Then, from (13), (11) and (10), we have

$$\begin{aligned} E[\text{Used}] &= CE[N] \\ &= Ce^{L\lambda} \end{aligned} \tag{20}$$

and

$$\begin{aligned} E[\text{Used} + \text{Idle}] &= E[\text{Used}] + E[\text{Idle}] \\ &= Ce^{L\lambda} + 1/\lambda. \end{aligned} \tag{21}$$

Finally, by substituting (20) and (21) into (10), we obtain (5)

$$U = \frac{\lambda C}{\lambda C + e^{-L\lambda}} \tag{22}$$

2.3 Task Guarantee Rate, G

In this subsection, we derive formula (6) for the task guarantee rate, G.

By the conservation law of queueing theory, we should have

$$U = \lambda G C \tag{23}$$

Solve G, we obtain

$$\begin{aligned} G &= \frac{U}{\lambda C} \\ &= \frac{1}{\lambda C + e^{-L\lambda}}. \end{aligned} \tag{24}$$

This is exactly (6).

We can directly obtain (6) from renew theory. Recall that in the last subsection, we define that the system has two states: *on (used)* and *off (idle)*. We now still keep this state definition, but change the unit of measurements for the random variables Used and Idle. In deriving the formula for the CPU utilization, On is measured by the time CPU being used and Off by the time CPU not being used. And CL is the time length of whole cycle. To derive the formula for guarantee ratio, we let On be measured by the number of tasks which are guaranteed in the cycle, that is, N; and Off be the number of tasks which are not guaranteed, and Cl be the total number of tasks which are generated in cycle, and we call it *Total*. That is, Total is a random variable for the total number of tasks generated in a cycle. By (9), we have

$$G = \frac{E[N]}{E[\text{Total}]}. \tag{25}$$

Let T = Used + Idle be random variable for the time length of a cycle. We calculate $E[\text{Total}]$ by conditioning on T:

$$\begin{aligned} E[\text{Total}] &= E[E[\text{Total}|T = t]] \\ &= E[\lambda T] \\ &= \lambda E[T] \\ &= \lambda E[\text{Used} + \text{Idle}] \\ &= \lambda C e^{L\lambda} + 1. \end{aligned} \tag{26}$$

Substitute (19) and (26) into (25), we prove (6) again:

$$G = \frac{1}{\lambda C + e^{-L\lambda}} \tag{27}$$

REFERENCES

[1] Ross, Sheldon M., Introduction to Probability Models, Academic Press, Inc., 185.

[2] Kurose, James, and Singh, Suresh, "A study of Quasi-Dynamic Load Balancing in Soft Real-Time Distributed Computer Systems", *submitted to a conference*, 1986.

[3] Ramamritham, Krithivasan, and Stankovic, John A., "Dynamic Task Scheduling in Distributed Hard Real-Time Systems", *IEEE Software*, Vol. 1, No. 3, July 1984.

[4] Wang, Y., and Morris, R., "Load Sharing in Distributed Systems", *IEEE Transactions on Computers*, Vol. C-34, No. 3., March 1985.

[5] Zhao, W., Ramamritham, K., and Stankovic, J.A., "Scheduling Tasks with Resource Requirements in Hard Real-Time Systems", to be published on *IEEE Transactions on Software Engineering*, Submitted at April 1985.

[6] Zhao. W., "A Heuristic Approach to Scheduling Hard Real-Time Tasks with Resource Requirements in Distributed Systems", *Ph.D. Dissertation, University of Massachusetts at Amherst, February 1986,*