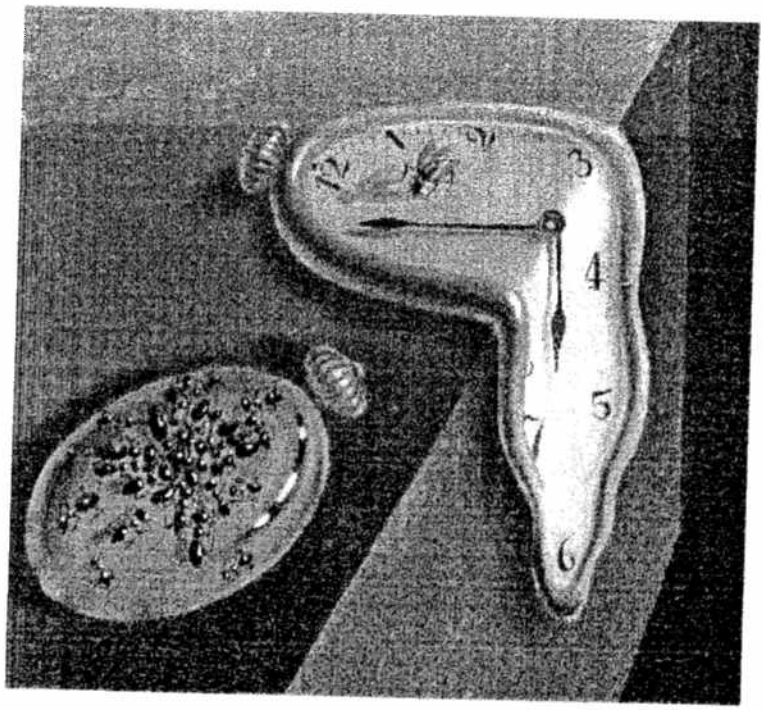
Institut für Automation
Abt. für Automatisierungssysteme

Technische
Universität
Wien

Projektbericht Nr. 183/1-15
Jänner 1991

Qualifying Dynamic Task Scheduling in Hard Real-Time Systems: A Novel Approach

J. Blieberger, U. Schmid



Ausschnitt aus: Salvador Dali, "Die Beständigkeit der Erinnerung"

QUALIFYING DYNAMIC TASK SCHEDULING IN HARD REAL-TIME SYSTEMS: A NOVEL APPROACH (EXTENDED ABSTRACT)

J. BLIEBERGER AND U. SCHMID

ABSTRACT. This paper surveys our recent research on qualifying algorithms for dynamic task scheduling with hard deadline constraints. Our abstract model of the real world relies on a system consisting of a task scheduler, a task list, and a single server. This system serves indeterministically arriving tasks, with arbitrary but independent task arrival and task execution time distributions. Our qualification of different scheduling techniques is based on the probability distribution of a random variable SRD(T), which is the length of time that passes before a task's service time violates a fixed deadline T. Our investigations quantify the (superior) performance of FCFS scheduling (which corresponds to the *earliest deadline first* discipline due to our fixed deadline assumption, of course) and the (poor) behaviour of systems using preemptive or nonpreemptive LCFS scheduling. The mathematical methods used for the derivation of the results are well-established in the analysis of algorithms and data structures, and form the basis of ongoing research by the authors in the area of real-time systems, too.

1. INTRODUCTION

A well-known design problem for hard real-time systems concerns methods for a suitable *task scheduling*. Scheduling goals for hard real-time systems are completely different from those fitting the needs of ordinary computer systems. The whole problem is sufficiently well-understood in the case of *deterministic* task arrivals, mainly *periodic* tasks. Requirements of this type may be scheduled in advance, i.e., *offline*; systems relying on this idea are usually called *static*. On the other hand, sufficient theoretical foundations for *indeterministic* task arrivals are lacking. Scheduling in such *dynamic* systems has to be performed during normal operation, i.e., *online*.

Some of our recent research addresses the problem of qualifying scheduling techniques for indeterministic task arrivals in hard real-time systems. Relying on a simple but general abstract model we succeeded in defining a (mathematically tractable) quality measure for scheduling algorithms. The complete derivation of our major results, which rely on some well-established combinatorial and asymptotic methods from the *analysis of algorithms and data structures*, is contained in a number of "mathematically-oriented" papers, cf. [BS92b], [SB92b], [BS92a], [SB92a].

Key words and phrases. real-time behaviour, FCFS scheduling, LCFS scheduling, probability generating functions, asymptotic methods.

This paper summarizes some of our results and presents insights gained from the work on the subject (obviously with an eye to presenting them to researchers working on the theory of real-time systems, but not being primarily mathematically inclined). The outline of the rest of the paper is as follows: Section 2 contains a description of the underlying model and the definition of the quantity of interest. Section 3 provides some notational conventions, Section 4 is devoted to our major results. At last, some conclusions are appended in Section 5.

2. The Model

Our investigations rely on a system consisting of a task scheduler, a task list of (potential) infinite capacity, and a single server. Arriving tasks are inserted into the task list by the scheduler, according to the scheduling discipline, of course. The server always executes the task at the head of the list, that is, scheduling is done by rearranging the task list. A dummy task will be generated by the scheduler if the list becomes empty. If the server executes a dummy task, the system is called *idle*, otherwise *busy*.

Rearranging the task list is assumed to occur at discrete points on the time axis only, without any scheduling overhead. The (constant) time interval between two such points is called a *cycle*. Due to this assumption we are able to model tasks formed by indivisible, i.e., atomic *actions* with duration of 1 cycle. The *task execution time* of a task is the number of cycles necessary for processing the task to completion if it might occupy the server exclusively. An ordinary task may have an arbitrary task execution time, a dummy task as mentioned above is supposed to consist of a single no-operation action (1 cycle). The *service time* of a task is the time (measured in cycles) from the beginning of the cycle in which the task arrives at the system to the end of the last cycle of that task.

Obviously, the time axis is covered by *idle periods* and *busy periods*, which are supposed to include the initial idle cycle, too. A sequence of nonviolating busy/idle periods followed by a busy period containing at least one deadline violation is called a *run*, the sequence without the last (violating) busy period is refered to by *successful run*. Supposing a *fixed service time deadline* of T cycles, we gain a quality criterion for a scheduling algorithm by introducing the random variable *successful run duration* $\mathrm{SRD}(T)$. This random variable denotes the time interval from the beginning of an initial idle cycle to the beginning of the (idle) cycle initiating the busy period containing the first violation of a task's deadline T. Thus, even the expected value of $\mathrm{SRD}(T)$ allows to estimate a particular scheduling algorithm, since it provides some insight into how long the system will operate from a time it is found idle (for example, initially turned on) to the first violation of a deadline.

The *probability generating function* (PGF) of the number of task arrivals during a cycle is denoted by

$$A(z) = \sum_{k \geq 0} a_k z^k, \qquad \text{where } a_k = \mathrm{prob}\{k \text{ tasks arrive during a cycle}\}$$

and should meet the constraint $a_0 = A(0) > 0$, i.e., the probability of no arrivals during a cycle should be greater than zero. This assures the existence of idle cycles. Note, that our definition implies that arrivals during two different cycles are independent.

The PGF of the task execution times (measured in cycles) is denoted by

$$L(z) = \sum_{k \geq 0} l_k z^k, \qquad \text{where } l_k = \text{prob}\{\text{task execution time is } k \text{ cycles}\}$$

with the additional assumption $L(0) = 0$, i.e., all task execution times should be greater than or equal to one cycle. Again, this definition implies that task execution times are independent from each other and from the arrival process.

It turns out that the overall execution time, i.e., the number of cycles necessary for processing all actions induced by task arrivals during one cycle, plays a central role. The appropriate PGF evaluates to $P(z) = A(L(z))$. For technical reasons we need some additional conditions on $A(z)$, $L(z)$, and $P(z)$, respectively. We omit a detailed discussion for the sake of simplicity since most of these conditions are usually easy to establish. Note however, that we explicitely exclude the trivial case $P(z) = p_0 + (1 - p_0)z$.

3. Notational Conventions

We use the following standard notations:

(1) $f(x) = \mathcal{O}(g(x))$ for $x \to x_0$, if there exists some real constant $C > 0$ independent of x which guarantees $|f(x)| \leq C|g(x)|$ for all x in a suitable neighbourhood of x_0.

(2) $f(x) \sim g(x)$ for $x \to x_0$, if $\lim_{x \to x_0} f(x)/g(x) = 1$.

Furthermore, we use an intuitively clear notation for comparing functions with different asymptotic growth ratios (cf [GKP89, p. 426ff]). We write

$$a_n \prec b_n \qquad \Longleftrightarrow \qquad \lim_{n \to \infty} \frac{a_n}{b_n} = 0$$

and

$$a_n \leq b_n \qquad \Longleftrightarrow \qquad \lim_{n \to \infty} \frac{a_n}{b_n} = C, \quad 0 \leq C \leq 1$$

in order to "compare" functions with the same asymptotic growth ratio.

For example, let $a_n = n^2$ and $b_n = 2n^2$. Then we have neither $n^2 \prec 2n^2$ nor $2n^2 \prec n^2$. But, since b_n grows only "a little bit" faster than a_n, we have the intuitively meaningful notation $a_n = n^2 \leq 2n^2 = b_n$.

4. Known Results

This section lists some of our major results. In fact, we summarize asymptotic expressions for the mean of SRD(T) as T gets large, for FCFS and both nonpreemptive and preemptive LCFS Scheduling.

We have to consider three different cases, namely

(1) *Normal Case*

This (most important) case is characterized by an average load of less than 100%, which may be expressed by $P'(1) < 1$ (since $P'(1)$ equals the average number of actions caused by task arrivals within a cycle). That is,

our system has to deal with task arrivals keeping it not totally busy on the average.

(2) *Balanced Case*
Here, our system is kept 100% busy on the average, i.e., $P'(1) = 1$.

(3) *Overloaded Case*
This case may be characterized by an average (offered) load which is higher than the maximum load the system is able to cope with, that is, $P'(1) > 1$.

First, we present the average length of a successful run (measured in cycles) for the normal case. It turns out that these values grow exponentially in T, for all scheduling techniques investigated. Note, that we derived asymptotic expressions for all higher moments, too. In fact, it should be possible to prove that the limiting distribution is an exponential one; details may be found in our original papers.

Theorem 4.1 (Theorem FCFS). *(FCFS scheduling in the normal case, cf.* [SB92b, Theorem 1]*). The mean of* SRD(T) *is given by*

$$\mu_{FCFS} \sim \frac{P'(\kappa) - 1}{(\kappa - 1)(1 - P'(1))^2} \kappa^T \qquad \text{for } T \to \infty,$$

where $\kappa > 1$ is the solution of $x = P(x)$, $x > 1$. □

Theorem 4.2 (Theorem npLCFS). *(nonpreemptive LCFS scheduling in the normal case, cf.* [SB92a, Theorem 5.1]*). The mean of* SRD(T) *is given by*

$$\mu_{npLCFS} \sim C\, T^{3/2} \rho^T \qquad \text{for } T \to \infty,$$

where

$$C = \frac{2\sqrt{\pi}(\rho - 1)(\tau - a_0)L(\tau)}{bL(\rho)(1 - P'(1))} \left(\frac{(1 - a_0)\,(L(\tau) - L(a_0))\, a_0(\rho - 1)}{L(a_0)(\tau - a_0)} - \frac{(\tau - 1)(\tau - a_0\rho)L'(\tau)}{L(\tau)} + \tau - a_0 \right)^{-1},$$

$\tau > 1$ is the solution of $P(x) = xP'(x)$, $\rho = \tau/P(\tau) > 1$ and $b = \sqrt{2P(\tau)/P''(\tau)}$.
□

Theorem 4.3 (Theorem pLCFS). *(preemptive LCFS scheduling in the normal case, cf.* [BS92a, THeorem 2]*). The mean of* SRD(T) *is given by*

$$\mu_{pLCFS} \sim \left(\frac{2\pi P''(\tau)}{P(\tau)} \right)^{1/2} \frac{\tau^2(\rho - 1)}{\rho^2(1 - P'(1))}\, T^{3/2} \rho^T \qquad \text{for } T \to \infty,$$

where $\tau > 1$ is the solution of $P(x) = xP'(x)$ and $\rho = \tau/P(\tau) > 1$. □

The following theorems list the results concerning the expected value of SRD(T) for the balanced case. Note that we derived expressions for the variance of SRD(T), too, but no results concerning the limiting distributions as yet.

Theorem 4.4 (Theorem FCFS$^{\rightarrow}$). *(FCFS scheduling in the balanced case, cf.* [BS92a, Theorem 2]*). The mean of* SRD(T) *is given by*

$$\mu_{FCFS^{\rightarrow}} \sim \frac{1}{\psi_i} \frac{i!}{(i - 1)(2i - 1)!} T^i \qquad \text{for } T \to \infty,$$

where $i \geq 2$ denotes the order of the zero of $P(x) - x$ at $x = 1$, i.e., the smallest integer value of i such that

$$P(x) - x = \psi_i (x-1)^i + \mathcal{O}\big((x-1)^{i+1}\big) \qquad \text{for } x \to 1$$

and $\psi_i \neq 0$. □

Theorem 4.5 (Theorem npLCFS$^{\rightarrow}$). *(nonpreemptive LCFS scheduling in the balanced case). We are able to show that the mean of* SRD(T) *is given by*

$$\mu_{npLCFS^{\rightarrow}} \sim T \qquad \text{for } T \to \infty. \quad \square$$

Theorem 4.6 (Theorem pLCFS$^{\rightarrow}$). *(preemptive LCFS scheduling in the balanced case). We are able to show that the mean of* SRD(T) *is given by*

$$\mu_{pLCFS^{\rightarrow}} \sim T \qquad \text{for } T \to \infty. \quad \square$$

At last, we summarize the results concerning the mean of SRD(T) in the overloaded case. As one might have expected, we obtain a very short average successful run duration, even for large T. Again, we derived expressions for the variance of SRD(T), too, but no results concerning the limiting distribution as yet.

Theorem 4.7 (Theorem FCFS^r). *(FCFS scheduling in the overloaded case, cf.* [BS92a, Theorem 1]*). The mean of* SRD(T) *is given by*

$$\mu_{FCFS^r} \sim \frac{\beta}{1-\beta} \frac{1}{1-P'(\beta)} \qquad \text{for } T \to \infty,$$

where $\beta < 1$ is the solution of $x = P(x)$, $x < 1$. □

Theorem 4.8 (Theorem npLCFS^r). *(nonpreemptive LCFS scheduling in the overloaded case). The mean of* SRD(T) *is given by*

$$\mu_{npLCFS^r} \sim \frac{\beta}{1-\beta} \frac{1}{1-P'(\beta)} \qquad \text{for } T \to \infty,$$

where $\beta < 1$ is the solution of $x = P(x)$, $x < 1$. □

Theorem 4.9 (Theorem pLCFS^r). *(preemptive LCFS scheduling in the overloaded case). The mean of* SRD(T) *is given by*

$$\mu_{pLCFS^r} \sim \frac{\beta}{1-\beta} \frac{1}{1-P'(\beta)} \qquad \text{for } T \to \infty,$$

where $\beta < 1$ is the solution of $x = P(x)$, $x < 1$. □

5. Conclusions

This section provides general discussions and ratings of the previous results. To save space, we have omitted details concerning the methods used for deriving our results, which may be found in our original papers. However, we should mention that we were successful in deriving our results using combinatorial and asymptotic methods from the *analysis of algorithms and data structures*, and not *queueing theory*. We are convinced that there are a lot of problems in the field of real-time systems, which may be attacked by similar methods; we will mention some possible directions of further research at the end of this section.

We investigated the ability of a scheduling algorithm to meet the deadlines of all tasks arriving at the system, namely, the random variable *successful run duration* SRD(T). The most important results concern exponentially growing means of SRD(T) (as T gets large), for FCFS and both nonpreemptive and preemptive LCFS scheduling, in the normal case of low average load. By virtue of such results one might expect that our system will operate properly a very long time, regardless of the scheduling policy. Numerical results concerning a particular example showed very impressive results, too. By the way, note that a simulation approach for reasonable values of T seems to be impossible; such an attempt would last too long, even on a CRAY computer! Thus, we have solved a problem by means of analytic modelling, which is not tractable by simulation, contradicting the widespread view of simulation being a panacea!

Though we expected a superior performance of FCFS scheduling because it is equivalent to the *earliest deadline first* scheduling discipline (due to our fixed deadline assumption), we never expected such a significant difference concerning the deadline meeting properties between FCFS and LCFS scheduling. In fact, since κ is larger[1] than ρ, both LCFS algorithms are found to be inferior to FCFS in the normal case, which is summarized in the following theorem:

Theorem 5.1 (Low Load Theorem). *If the system is concerned with low load only, we have*

$$\mu_{pLCFS} \leq \mu_{npLCFS} \prec \mu_{FCFS}. \quad \square$$

The balanced case shows that FCFS is still superior to both LCFS scheduling algorithms. Comparing our Theorems pLCFS$^{\rightarrow}$, npLCFS$^{\rightarrow}$, and FCFS$^{\rightarrow}$ yields the following rating:

Theorem 5.2 (Balanced System Theorem). *In the balanced case, we have*

$$\mu_{pLCFS^{\rightarrow}} \sim \mu_{npLCFS^{\rightarrow}} \prec \mu_{FCFS^{\rightarrow}}. \quad \square$$

At first sight, the results of the overloaded case are a little bit puzzling because they are asymptotically equivalent. Deadline missing in the case of overload, however, seems to be mainly caused by the general inability of the system to deal with the whole work, and not by the scheduling strategy. Thus, our results do not provide an example demonstrating that FCFS is not optimal in this case.

Though our results are limited due to our somewhat stationary probability model, they are still useful because of their non-equilibrium nature. For example,

[1] This fact may be proved in general and will be supported by providing numerical comparisons in the final paper.

using appropriate probability distributions for task arrivals and for task execution times, we may determine some limits regarding the tolerable length of successful run durations. That is, given a (tolerable) probability of deadline missing (say, 10^{-9}), we are able to compute the maximum duration of such periods (assuming that the limiting distribution of SRD(T) is exponential, of course).

Needless to say, the whole approach is only a modest start to analytic modelling of systems for real-time applications; there are a lot of more or less important problems left:

(1) Definition and investigation of other quantities which describe real-time behaviour better than our SRD(T) does.
(2) Application of our approach to other scheduling algorithms, such as priority based scheduling.
(3) Adding system overhead for scheduling and dispatching.
(4) Dropping the limitation of a single server.
(5) Considering the occurrence of deterministic and cyclically created tasks.
(6) Releasing the fixed deadline assumption.

Obviously, a crucial point is how to model the task arrival process to meet practical requirements. This problem, which is central to all attempts of analytic modelling a real application, is not solved sufficiently. In order to preserve the tractability of the computations, one is traditionally tempted to use the well-thumbed exponential or geometric distributions, or parameter variant normal distributions such as in diffusion approximation. Unfortunately, these approaches are justified for some traditional applications only (large timesharing systems, for example), but it seems questionable to be successful with them in real-time applications.

Hence, the developement of an approach which allows the extension of our stationary probability model to a more suitable dynamic one seems to be of central importance. In order to obtain an adequate model, it is necessary to investigate real applications with regard to the stimuli they are concerned with, i.e., there is a need of know-how in monitoring real-time systems; both how to do it and what quantities are to be monitored to obtain desired characteristics. On the other hand, refined techniques for tracting the theoretical part are necessary in order to make use of an adequate model; in fact a broad field of theoretical and practical research!

References

[BS92a] Johann Blieberger and Ulrich Schmid. FCFS scheduling in a hard real-time environment under rush-hour conditions. *BIT*, 32(3):370–383, 1992.

[BS92b] Johann Blieberger and Ulrich Schmid. Preemptive LCFS scheduling in hard real-time applications. *Performance Evaluation*, 15(3):203–215, 1992.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, MA, 1989.

[SB92a] Ulrich Schmid and Johann Blieberger. On non-preemptive LCFS scheduling with deadlines. (to appear), 1992.

[SB92b] Ulrich Schmid and Johann Blieberger. Some investigations on FCFS scheduling in hard real-time applications. *Journal of Computer and System Sciences*, 45(3):493–512, 1992.

Department of Automation (183/1) at the Technical University of Vienna, Treitlstrasse 3/4, A-1040 Vienna, AUSTRIA

E-mail: blieb@auto.tuwien.ac.at

E-mail: s@auto.tuwien.ac.at