UNIVERSIDAD OF OVIEDO

Computer Science Department

Stochastic Analysis of the Steady-State Backlog in Periodic Real-Time Systems

Jose Luis Díaz, Jose María López, Daniel F. García

Technical Report
May, 2003

Abstract

Classical analysis of real-time systems focuses in the study of the "worst-case" scenario, by assuming that all task will require their worst-case execution time, and a critical instant in which the response time will be the longest. Recently, some authors modelled the execution time as a random variable, and developed several techniques in order to obtain the distribution of the response time, from which the probability of deadline misses can be derived.

We introduced in Díaz *et al.* [2002] one of these techniques, which can be applied when the maximum system utilization is not greater than 1. In this report, we show that these techniques can also be applied to systems with maximum utilization greater than 1, whenever the *average* system utilization is not greater than 1. For this case, we prove the existence of a *steady-state* regime, in which the response time of all jobs follow a stationary probability function. The response time distributions for that steady-state regime can be derived from the steady-state backlog distribution, using the techniques presented in Díaz *et al.* [2002], so in this report we concentrate solely in the obtention of this steady-state backlog distribution. We provide a methodology for the computation of the exact distribution, and investigate some approximations which have less computational cost than the exact solution.

Contents

List of Figures

List of Tables

Notation

Model and analysis notation

$\mathbf{A}$	Matrix derived from $\mathbf{P}$, which express the recurrence relation existent among the components of $\boldsymbol{\pi}$. (See pag. 33)
$b_i(j)$	Element of the Markov matrix $\mathbf{P}$, located at column i, row j. (See pag. 21)
$\mathcal{C}$	Execution time (random variable), equal to the time required by the task if no other tasks were active in the system. $\mathcal{C}_i$ denotes the execution time of the i-th task in a periodic tasks system, while $\mathcal{C}_j$ denotes the execution time of the j-th job activation from the time origin. (See pag. 8)
$C^{\mathbf{max}}$	Maximum execution time of a task (*WCET*). (See pag. 9)
$\bar{C}$	Average execution time of a task. (See pag. 13)
$C^{\mathbf{min}}$	Minimum execution time of a task. (See pag. 9)
D_i	Relative deadline of i-th task in the periodic tasks system. (See pag. 8)
$f_{\mathcal{X}}(\cdot)$	Probability function (PF) of the random variable $\mathcal{X}$. Also called "probability mass function" (PMF). (See pag. 9)
Γ	Job. Task instance. Sometimes it carries two subindexes (e.g. $\Gamma_{i,j}$), to indicate the j-th activation of the i-th task. Other times it carries a single subindex (e.g. Γ_j), when the task to which it belongs is not important, to denote the j-th job in the sequence of jobs activations from the time origin. (See pag. 9)
i	Index commonly used to refer to the parameters of a task, like in $\mathcal{C}_i$, D_i, etc. (See pag. 10)
j	Index commonly used to refer to the parameters of the job which makes the j-th place in the job sequence, when the task to which it belongs does not matter. (See pag. 10)
$\lambda_{i,j}$	Activation instant of the j-th instance of task τ_i. (See pag. 10)
λ_j	Activation instant of the j-th job from the time origin. (See pag. 10)
λi	Eigenvalues of matrix $\mathbf{A}$. (See pag. 33)
m_r	Row index of the last non-null element in the r-th column of the Markov matrix $\mathbf{P}$. (See pag. 21)
N	Total number of tasks in the periodic tasks system. (See pag. 8)
$\mathbf{P}$	Markov matrix governing the backlog process $\{\mathcal{W}(k)\}$. (See pag. 21)

$\boldsymbol{\pi}$	Column vector, solution of equation $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$. If its sum is 1, it is the probability function of the steady state backlog. (See pag. 22)
Φ_i	Phase or offset of i-th task in the periodic tasks system. (See pag. 8)
P_i	Priority of task τ_i, for task-level priority assignment policies such as RM or DM. (See pag. 8)
P_j	Priority of the j-th job's activation, from the time origin, for job-level priority assignment policies, such as EDF. (See pag. 8)
$\mathbb{P}\{x\}$	Probability of event x. (See pag. 9)
$\mathcal{R}_i$	Response time of task τ_i. It is a random variable. (See pag. 9)
$\mathcal{R}_{i,j}$	Response time of the j-th instance of task τ_i. (See pag. 10)
r	Maximum spare time. It is the time in which the system is idle during the first complete hyperperiod (defined in 5), for the case in which all the tasks require their minimum execution time. Markov matrix $\mathbf{P}$ shows a regular structure after its r-th column. (See pag. 21)
S	System, set of periodic tasks. (See pag. 8)
s	Minimum spare time. It is the time in which the system is idle during the first complete hyperperiod (defined in 5), for the case in which all the tasks require their maximum execution time. (See pag. 17)
τ_i	i-th task in the periodic tasks system. (See pag. 8)
T_i	Period of i-th task in the periodic tasks system. (See pag. 8)
$U^{\mathbf{min}}$	Minimum system utilization factor. (See pag. 13)
$\bar{U}$	Average system utilization factor. (See pag. 13)
$U^{\mathbf{max}}$	Maximum system utilization factor. It coincides with the total system utilization used in the clasical "worst-case" analysis. (See pag. 13)
$\mathbf{v}_i$	Eigenvectors of matrix $\mathbf{A}$. (See pag. 33)
$\mathcal{W}_k$	Backlog observed at the beginning of the k-th hyperperiod, i.e. at instant kT. It is a random variable. The sequence $\{\mathcal{W}_k\}$ is an stochastic process, called the "backlog process". (See pag. 17)
$W^{\mathbf{max}}$	Backlog at the end of the first complete hyperperiod (defined in 5), when all the tasks require their maximum execution time. (See pag. 17)
$W^{\mathbf{min}}$	Backlog at the end of the first complete hyperperiod (defined in 5), when all the tasks require their minimum execution time. (See pag. 21)

1. Introduction

Classical analysis of real-time systems [Liu and Layland, 1973; Lehoczky, 1990; Tindell *et al.*, 1994] focuses on guaranteeing the schedulability of the system when all jobs use their worst computation time.

This approach is very restrictive, as the worst computation time assumed for each task may be too pessimistic, and the situation under which each task would suffer the maximum interference from other tasks may be very unlikely, especially if we allow for variable computation times in the tasks. Both these factors lead to very high calculated response times, which *could* occur in theory, but with little *probability* in practice. There are many soft real-time applications whose tasks have highly variable computation requirements and deadlines are not hard. For these applications, the probability of missing a deadline could serve as a measurement of its Quality of Service (QoS). Moreover, knowledge of these probabilities can still be useful for the design of hard real-time systems, as long as the application allows for a given failure rate (for example, the probability of missing a deadline could be as small as the probability of hardware failure).

In [Díaz *et al.*, 2002] we have introduced a model in which the execution time of each job is considered an stochastic variable of known probability function. Thus, we consider not only the worst-case computation time, but all the possible computation times and their probabilities. We have presented an algorithm to calculate the statistical distribution of the response time for each job. This approach was already proposed by other authors [Tia *et al.*, 1995; Gardner, 1999; Abeni and Buttazzo, 1999; Atlas and Bestavros, 1998], but in all these works some aditional assumptions were made in order to simplify the problem (as for example, to restrict the deadlines of the tasks to be less than their periods, to analyze only the response time in a critical instant, or to require a special scheduler in order to guarantee isolation among tasks. The reader is refered to [Díaz *et al.*, 2002] to find more details about the motivation of our work, and their differences with the previous related work.

We will not repeat in this report the methods already explained in [Díaz *et al.*, 2002], in order to save space. Using these methods it is possible to obtain the probability function of the execution time of any job, given the probatility function of the backlog at the release instant of the first job in the sequence, and the release instants and execution time distributions of these jobs. This framework allows, at least in theory, for obtaining the probability function of the response time for any job. However, the total number of jobs is infinite, so it is not possible to numerically find the probability function response time for all of them. It is necessary to complete the method in order to avoid this "infinite" problem.

It is concevaible that, since the release instants of the job follow a repetitive pattern, their response times also follow a repetitive pattern. For example, perhaps the first job of each hyperperiod always presents the same response time distribution. If this were the case, the complete sequence of infinite jobs could be analyzed from a finite number of them.

In this report we will focus on the obtention of the backlog probability function at any instant. It will be shown that, in general, this probability function depends on time, and that under certain circunstances, it "converges" towards a steady-state distribution. We

will prove that the condition $\bar{U} < 1$ ensures this convergence, and provide methods for obtaninig the steady-state distribution. This way we can compute the probability function of the backlog at the beginning of a hypothetical "steady-state hyperperiod". Using this backlog as "initial backlog" and by applying the techniques in [Díaz *et al.*, 2002], it would be possible to obtain the response time distributions of all jobs in this hyperperiod. These represents the "steady-state" response time distributions, and thus the long-term behaviour of the system. From these distributions the probability of deadline misses for each task can be accurately computed.

In section 2 we present the system model, state the problem we want to solve, and define some important concepts. In section 3 we attack the problem of the convergence of the backlog distribution, with mathematical proofs of the conditions which ensure this convergence and the description of a methodology for numerically computing the exact distribution. In section 4 we show some ways of obtaining an approximation of that distribution, which are less computationally expensive.

2. System model and definitions

The system is composed of a single processor, wo which several **jobs** arrive at deterministic instants, known beforehand. The execution time of the jobs are random variables, and its probability function is also known. Each job has a priority associated to it, P_j (this can be a fixed-priority, set at design time, or it can be dinamically set when the job arrives to the system). The priority assignement policy does not concern us. A scheduler is running in the processor, which guarantees that the job with maximum priority among all ready jobs is the one which receives the processor. The scheduler is preemptive, that is, if a new job arrives with priority greater than the one of the job currently executing, this one will be suspended, in order to let the new one to take the processor. A suspended job is resumed as soon as it becomes the job with greater priority among all ready jobs.

Each job is a **task** instance. All jobs corresponding to the same task have the same execution time probability function, and the same deadline, (and in the case of fixed priorities, the same priority P_i). The only difference among the jobs corresponding to the same task is their release instant (and its priority for the case of dynamic priorities). The release instants of the jobs belonging to a task are deterministic, and they are separated in time by a constant interval. We assume that jobs do not access to shared resources, and they do not have critical sections which may cause blocking to them.

2.1. System parameters

The system can be modelled as a set of independent periodic tasks $S = \{\tau_i\}$, with $i = 1, \ldots, N$, being each task defined by the following parameters $\tau_i = \{T_i, \Phi_i, D_i, \mathcal{C}_i\}$:

- **Period** T_i. Is the time elapsed between two successive releases of the task.

- **Offset** Φ_i. Is the instant of the first release of the task (the release instant of its first job).
- **Deadline**. It is the deadline, relative to the release instant, in which the job must finalize.
- **Execution time** $\mathcal{C}_i$. It is the amount of "processor resource" which is required by the job. It can be seen as the time that the job would require to complete, if its execution would start inmediately, and no preemption could occur[1].

The first three parameters are deteriminstic, but the last parameter (the execution time $\mathcal{C}_i$) is a random variable. In order to clearly differenciate random variables from deterministic variables, we will use a callygraphic typeface for the first ones.

Then, an aditional system parameter is the statistical distribution of the execution time $\mathcal{C}_i$ of each task. We do not impose any particular distribution (such as gaussian, exponential or others). So, in order to complete the system specification, it is necessary to give the *probability function* of the execution times. Assuming some *resolution* in the time measure, the execution times can only take integer values, and then the probability function will be a discrete one. This probability function assigns one probability to each possible execution time.

We will denote by $f_{\mathcal{C}_i}(\cdot)$ the probability function of the random variable $\mathcal{C}_i$. Denoting by $\mathbb{P}\{x\}$ the probability of event x, we have, by definition:

$$f_{\mathcal{C}_i}(c) \triangleq \mathbb{P}\{\mathcal{C}_i = c\} \tag{1}$$

Although the probability function is theoretically infinite, in practice the execution time of a task τ_i is comprised in a finite interval $[C_i^{\min}, C_i^{\max}]$, so the probability function $f_{\mathcal{C}_i}(\cdot)$ can be completely described by a finite number of points. An obvious data structure for storing this kind of functions is the one-dimensional array. Each possible execution time will be an index to the array, and the stored value at that index will be its probability.

We will call the "**response time**" of task τ_i, to the time elapsed between the instant in which the task is released until the instant in which it finishes its execution. We will denote it by $\mathcal{R}_i$. This quantity is a random variable. If the task is running alone in the system (or if it is the highest priority task), its response time will coincide with its execution time. That is, the PF of $\mathcal{R}_i$ will be equal to the PF of $\mathcal{C}_i$. However, in the general case, other tasks executing in the system will cause interference on it, and this will produce a different PF for $\mathcal{R}_i$.

The task concept is an abstraction. What are running in the system are **jobs**, which are task instances. Each task τ_i gives rise to an infinite sequence of jobs $\Gamma_{i,j}$, all of them with the same execution time PF, and the same relative deadline, but with different

[1] Not to be confused with the response time, which is the elapsed time from the release of the job until it finishes, taking into account that the remaining jobs in the system can delay its execution, or preempt it.

release instants. We will denote by $\lambda_{i,j}$, the release instant of job $\Gamma_{i,j}$. This instant is deterministic, and is given by:

$$\lambda_{i,j} = \Phi_i + jT_i \tag{2}$$

Frequently, we will need to refer to any generic job, without specifying the task to which it belongs. In these cases we will use the notation Γ_j, with a single subindex. This simply means the j-th job released since the system starting. We will mainly use the subindex j for this purpose, reserving the subindex i for the tasks.

2.2. Problem statement

Given a system S, defined as explained above, obtain the response time probability function for all its tasks.

From these probability functions, it is possible to derive the probability of metting any given deadline, and the probability of missing it. It is also posible to derive other useful information, such as the most probable response time, the average response time, the standard deviation, etc.

So, the final objective of the analysis is to provide the probability functions of the response time of each task. However, we must recall that the task concept is an abstraction, because what really gets executed in the system are *jobs*. The response time of each job can have a different probability function. The response time of a task can be obtained by averaging the probability functions of the jobs belonging to it. However, the number of jobs belonging to a task is infinite, so the problem is to obtain the response time distribution for a task without needing to compute the distributions for the infinite number of jobs. We will see that this is possible, thanks to the periodicity in the release instants of the jobs.

The response time of a job depends on three factors:

1. The backlog existent at the instant in which the job is released, due to previous jobs with greater or equal priority which are still active.
2. The execution time of the job under consideration
3. The execution time of all future jobs which could preempt the job under consideration

In [Díaz *et al.*, 2002] these three factors are analyzed, and different algorithms are presented in order to take into account each factor an compute the final response time distribution. In particular, given the probability function of the backlog at the release instant of the job, it suffices to perform a discrete convolution between this probability function and that of the execution time of the job, in order to obtain a first approximation of the response time distribution of the job. In fact, this approximation will be the exact distribution if no preemption were allowed. In order to take into account the effect of the possible preemptions, the algorithm "split, convolve and merge" is presented in [Díaz *et al.*, 2002]. The result of this algorithm is the exact probability function of the response time of the job.

It remains the problem of computing the probability function of the backlog at the release instant of the job. In [Díaz *et al.*, 2002] an algorithm called "convolve and shrink" is presented, which allows to obtain this probability function, assuming that the probability function of the backlog at a previous instant is known. Since the initial backlog of the system (when the system starts) is zero, we always can to apply the "convolve and shrink" method from that initial backlog, until arriving to any desired instant.

However a more efficient approach could be possible. Let us suppose that all hyperperiods have the same initial backlog (that is, the probability function is the same). In this case, in order to compute the probability function at any instant, it is not neccessary to rewind to the system start, but only to the beginning of the hyperperiod. The influence of the whole past of the system is summarized in the probability function of the backlog at the beginning of the hyperperiod.

We will formalize these ideas, and show under which circunstances the backlog distribution at the beginning of all hyperperiods remains the same.

2.3. Other concepts and definitions

Once all tasks are released at least once (that is, at any instant after the length of the longest offset), the jobs activations will follow a repetitive pattern.

This is a fundamental idea, because if a repetitive pattern exists in the release instants of the jobs, it is reasonable to expect some kind of repetitive behaviour in the system. Even if the execution times are random, it can be expected that the stochastic behaviour of the system becomes periodic, i.e. follows a repetitive pattern of statistical distributions.

Definition 1. *A **hyperperiod** is a time interval which encompass a sequence of tasks releases, such that the relative sequence of releases will repeat identically in the future.*

The hyperperiod length, which we will denote by T, is the least common multiple of the periods of the tasks.

$$T = \operatorname*{mcm}_i T_i \tag{3}$$

Figure 1 shows an example with three tasks, with offsets 4, 7 and 11, and periods 6, 8 and 12. For each task a time axis is represented in horizontal. Vertical arrows denote release instants. According with the above equation, the hyperperiod length is 24 for this example. In fact, this can be seen in the figure, because from instant $t = 11$ onwards, there exists a release pattern with length 24, which repeats time after time.

The concept of hyperperiod provides a starting point for the stochastic analysis of the system, which avoids the computation of the response time for an infinite number of jobs. Although the jobs sequence is infinite, the number of jobs within one hyperperiod is finite. If all hyperperiods had the same stochastic behaviour, in the sense that the response time distributions are the same than those in the previous hyperperiod, it would suffice to obtain these distributions for a single hyperperiod.

However, it is not sufficient to have a repetitive pattern in the release instants of the jobs in order to guarantee a repetitive stochastic behaviour. In addition, the initial

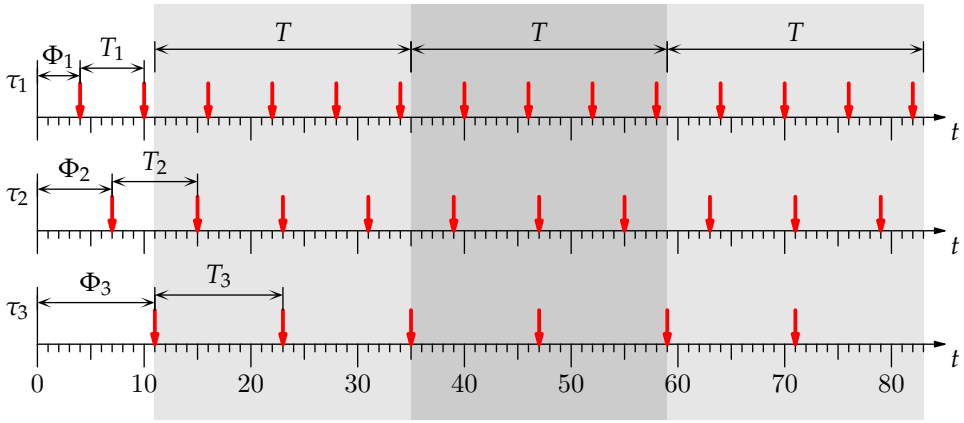


Figure 1: Illustration of the concept of hyperperiod

backlog in each hyperperiod has to be the same. On the contrary, the initial conditions will be different in each hyperperiod, and this would affect the response time of the jobs released in it.

In our problem, the backlog is a random variable, so we cannot ask its value to be the same at the beginning of each hyperperiod. Instead, we will ask its statistical distribution to be the same. That is, if the probability of the backlog being equal to i at the beginning of a given hyperperiod is p_i, then we will ask the same probability for the backlog at the beginning of the next hyperperiods. This way, all hyperperiods will have the same "initial conditions" (in a stochastic sense), and thus the jobs will have the same response time distributions.

We will show that the above exigence is met for some systems, in which the backlog distribution repeats identically at the beginning of all hyperperiods, while in other systems the exigence is not met and the initial backlog distribution is different for each hyperperiod. Among this second class, there are systems in which, although the backlog distribution is not the same from one hyperperiod to other, the difference among them decreases, approaching to zero as time goes on, so it is possible to talk about a "steady-state", which is reached after a sufficient number of hyperperiods. Finally, there exist systems in which the difference among the backlog distributions increases as time goes on.

We will show that the above classification depends on a simple parameter, which is the "utilization factor". In the classical analysis, the utilization factor is defined as the fraction of time in which the system is busy, assuming that all tasks are demanding their "worst case" execution time. This factor is easily obtained by summing up the quotient among the worst execution time and the period of each task.

In the stochastic case, however, each task is allowed to demand a random amount of execution time, so it is obvius that each hyperperiod can have a different utilization factor. In fact, the utilization is also a random variable, so we can define its maximum, minimum and average values, as follows:

Definition 2. *The **maximum utilization**, denoted by U^{max} is defined as:*

$$U^{max} = \sum_{i=1}^{N} \frac{C_i^{max}}{T_i} \tag{4}$$

being C_i^{max} the worst execution time of task τ_i. The maximum utilization represents the fraction of the hyperperiod in which the system is busy, for the worst case scenario.

Definition 3. *The **minimum utilization**, denoted by U^{min} is defined as:*

$$U^{min} = \sum_{i=1}^{N} \frac{C_i^{min}}{T_i} \tag{5}$$

being C_i^{min} the best execution time of task τ_i. The minimum utilization represents the fraction of the hyperperiodo in which the system is busy, for the best case scenario.

Definition 4. *The **average utilization**, denotaded by $\bar{U}$ is defined as:*

$$\bar{U} = \sum_{i=1}^{N} \frac{\bar{C}_i}{T_i} \tag{6}$$

being $\bar{C}_i$ the average execution time of task τ_i. The average utilization represents the fraction of the hyperperiod in which the system will be busy, in the average case.

Note that our definition of *maximum utilization* coincides with the classical concept of *total utilization factor*.

We will say that the system has reached a "steady-state", when the statistical distribution of the backlog is the same at the beginning of two consecutive hyperperiods (and thus, of all following hyperperiods). It is easy to tell if a system will reach a steady state or not, by only computing the values of $U^{\max}$ and $\bar{U}$. We will mathematically prove this assert, but it can be interesting to advance the result.

A system can be classified in three different classes, according to the values of the parameters $U^{\max}$ and $\bar{U}$ as follows:

1. If $U^{\max} \leq 1$, the steady state is reached "almost inmediately". To be precise, at the end of the first hyperperiod in which all task were released at least once.

   This kind of systems are very interesting from a practical point of view. In fact, all systems which can be analyzed with classical methods are of this kind.

2. If $U^{\max} > 1$ and $\bar{U} \leq 1$, the difference among the distribution of the backlog in two successive hyperperiods goes to zero as time goes on. The system "converges" towards a steady state, which is reached after an infinite number of hyperperiods. However, in practice, the difference among distributions can be negligible in a few hyperperiods.

   This kind of system is also interesting from a practical point of view, because it is usual that the maximum utilization (worst case utilization) is higher than 1,

System	Task	Parameters Φ_i	T_i	D_i	$\mathcal{C}_i$	Utilization $U^{\min}$	$\bar{U}$	$U^{\max}$
S_1	τ_1	4	6	6	U[1,2]	0.3750	0.6042	0.8333
	τ_2	7	8	8	U[1,2]			
	τ_3	11	12	12	U[1,3]			
S_2	τ_1	4	6	6	U[2,3]	0.7500	0.9792	1.2083
	τ_2	7	8	8	U[2,3]			
	τ_3	11	12	12	U[2,4]			
S_3	τ_1	4	6	6	U[2,4]	0.7500	1.125	1.5000
	τ_2	7	8	8	U[2,4]			
	τ_3	11	12	12	U[2,4]			

Table 1: Three example systems, with identical release patterns but different utilization factors

which would be make impossible to analyze the system with classical methods, and however the average utilization is less than 1, which could produce low probabilities of deadline misses.

3. If $\bar{U} > 1$, the steady state cannot be reached, and thus this kind of system cannot be analyzed.

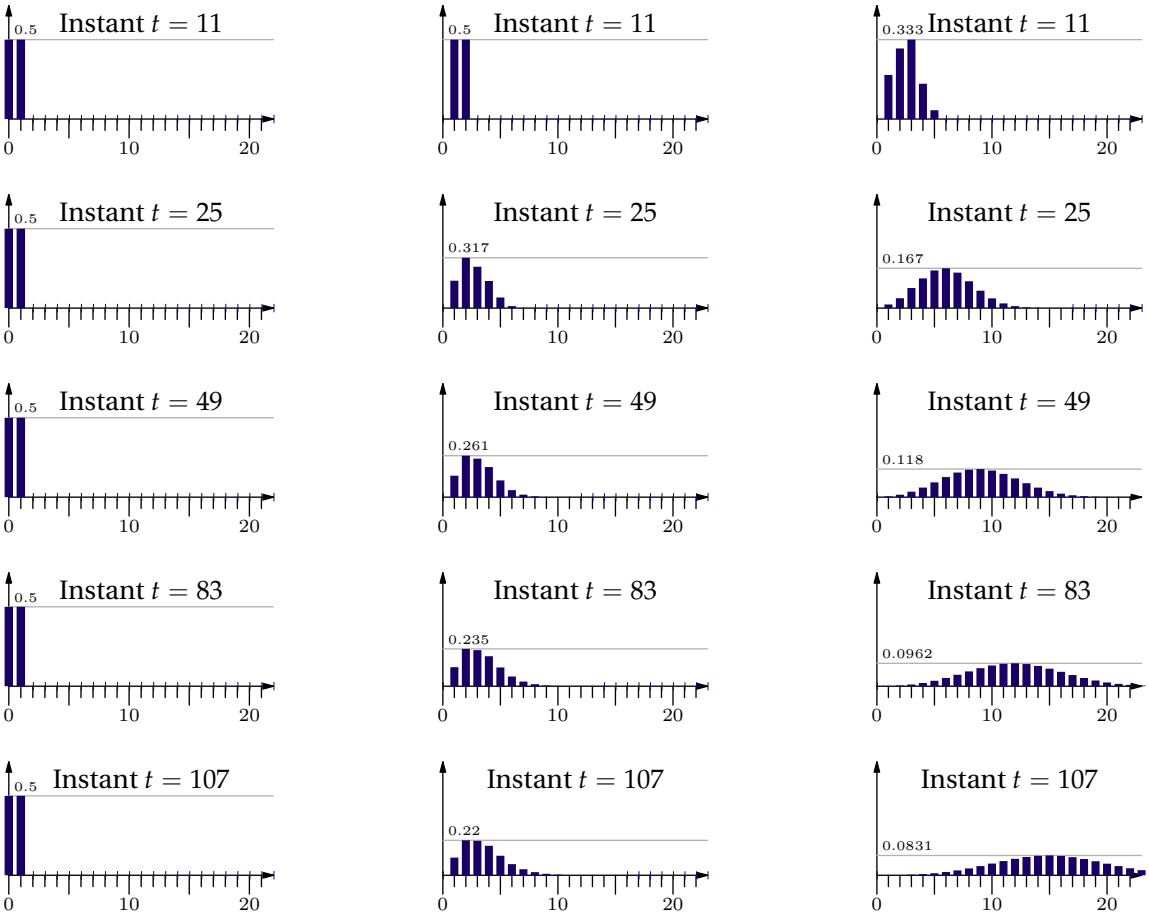
   Note, however, that this kind of system is unrealistic, because, in average, they are demanding to the system more time than the phisically available.

Let us see an example of a system of each type. Table 1 shows the parameters of three different systems, all of them composed of three tasks with the same periods, deadlines and offsets. The only difference among the examples are the execution time of the tasks. Note that the three systems have the same release pattern (coincident with the one shown in figure1).

The tasks execution times are random variables which follow, in this example, a uniform discrete distribution. For example, U[3,6] denotes that this random variable can take the values 3, 4, 5 or 6, all with the same probability (1/4 in this case). In the table, the values of $U^{\max}$, $U^{\min}$ y $\bar{U}$ are shown. They were computed according to equation (4), (5) and (6), respectively.

In system S_1, $U^{\max}$ is less than 1, so the system will reach its steady-state right at the end of the first hyperperiod. Figure 2(a) shows the backlog probability function at different instants, separated by 24 time units (the hyperperiod lenght). The first instant shown in the figure is $t = 11$, which is the instant in which all tasks were released at least once, c.f. Fig. 1). It can be observed that the probability function is the same at the beginning of each new hyperperiod.

In system S_2, however, the value of $U^{\max}$ is greater than 1, but since the average utilization $\bar{U}$ is less than 1, this system has to have a steady state, which is reached after a sufficient number of hyperperiods. In Fig. 2(b) it can be seen that the backlog



(a) System S_1 (maximum utilization less than 1)

(b) System S_2 (average utilization less than 1)

(c) System S_3 (average utilization greater than 1)

Figure 2: Evolution of the backlog distribution in the example systems

probability function is not exactly the same in different hyperperiods, but they are more and more closer. We will prove the existence of al limiting probability function, and we will give a method for obtaining it.

Finally, in system S_3 the value of $\bar{U}$ is greater than 1, so this system will never reach a steady-state. In Fig 2(c) it can be seen that in this case the probability function "stretches", "flattens" and "drifts" far from the origin. This means that the backlog accumulated at the end of each hyperperiod increases without limit. Any given value of the backlog is exceeded with high probability, after a sufficient number of hyperperiods has elapsed.

3. Steady-state backlog computation. Markovian analysis

3.1. Preliminary concepts

Let T be the length of the hyperperiod, defined as seen in eq. (3). We can divide the time-line in "zones", of length T, starting at $t = 0$; this way the instants kT, with $k = 0, 1, 2, \ldots$ are marked as hyperperiod beginnings (see fig. 3). However, as it can be seen in the figure, since each task has a different offset, in the first or firsts hyperperiods the activations pattern could be different to the one which appears in subsequent hyperperiods.

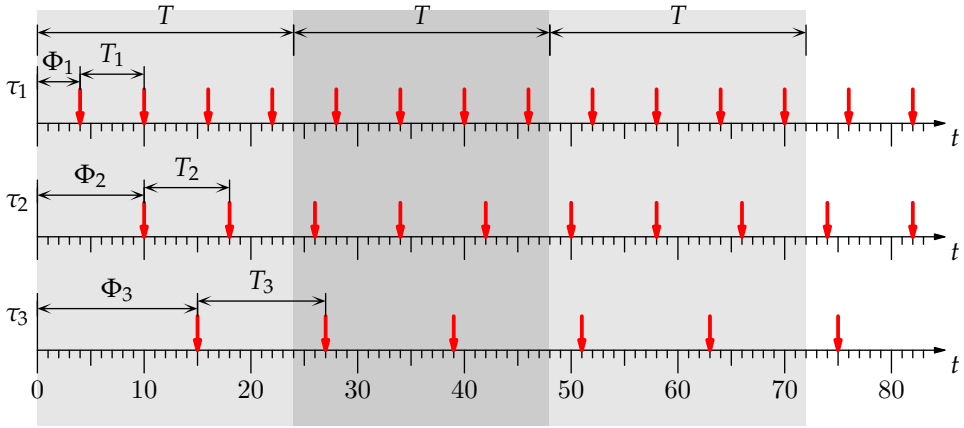


Figure 3: Example of irregularity in the first hyperperiods

Due to this situation, it results convenient the following definition:

Definition 5. *We call **complete hyperperiod** to a time interval of the form $[kT, (k+1)T)$ such that, inside that interval, each task τ_i is activated T/T_i times, being T the hyperperiod length and T_i the task period.*

*We call **first complete hyperperiod** to the complete hyperperiod with minimum k. We will denote this minimum k by k_0. This way, the first complete hyperperiod is the time interval $[k_0T, (k_0+1)T]$.*

In the example of fig. 3, the first complete hyperperiod begins at $t = 24$ and thus, in this case, $k_0 = 1$.

It is clear that, before the first complete hyperperiod we will have a kind of "transient regime", in which the activation pattern is not periodical. From the first complete hyperperiod on, the system enters into a "steady-state of activations", in which all subsequent hyperperiod will be complete, and the activation pattern will be the same for all of them.

However, even if the activation pattern is the same, this does not guarantee the same stochastic behavior (i.e: the same response time distributions), because this depends in general on the whole past of the system. This past can be summarized in the backlog distribution, and this one varies with time. Only if this distribution also presents a repetitive pattern the stochastic behavior of the system would be also repetitive.

So we are interested in the observation of the random variable $\mathcal{W}(t)$ at instants $t = kT, k = 0, 1, 2, \ldots$. In order to simplify the notation, we will introduce a new definition.

Definition 6. *We denote the backlog existent at instant kT by $\mathcal{W}_k$ and we call it "initial backlog of the k-th hyperperiod". I.e: $\mathcal{W}_k = \mathcal{W}(kT)$.*

3.2. Trivial case: $U^{\max} \leq 1$

In the following theorem, we will prove that, if the maximum utilization is less than 1, the backlog distribution will repeat identically at the beginning of each hyperperiod. In this case, thus, the analysis can be restricted to a single hyperperiod (as was done in the example in [Díaz *et al.*, 2002]).

Theorem 1. *If the initial backlog is zero at $t = 0$, and the system fulfills $U^{max} \leq 1$, then the backlog distribution at the end of the first complete hyperperiod ($t = (k_0 + 1)T$), will repeat at the end of all subsequent hyperperiods*

Proof. Let us imagine the scenario in which all jobs require their maximum execution time. Let $W^{\max}$ be the backlog observed at the end of the first complete hyperperiod under these circumstances, and s the number of time units in which the system is idle during this first complete hyperperiod (we will refer to s as the "minimum spare time"). It is clear that this spare time, plus the workload generated in this hyperperiod is equal to the hyperperiod length plus the final backlog, as shown in fig. 4. In addition, the workload generated in the hyperperiod for this worst-case scenario is equal to $TU^{\max}$, so we have:

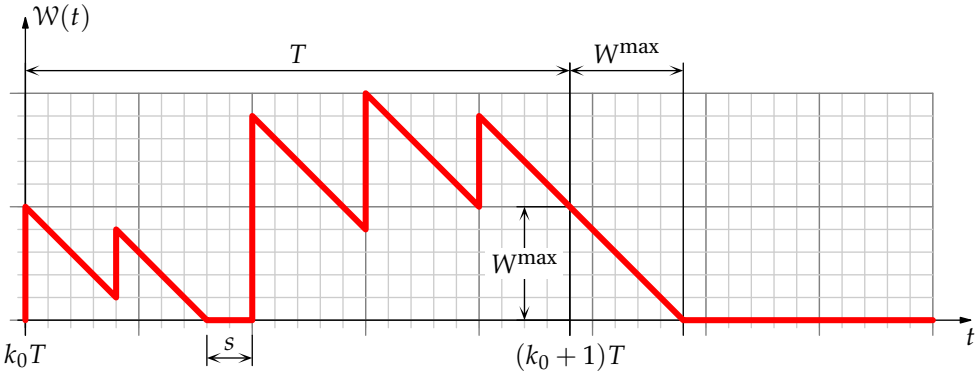
$$s + TU^{\max} = T + W^{\max}$$

By hypothesis $U^{\max} \leq 1$, so $TU^{\max} \leq T$. Using this fact in the previous equation we obtain $W^{\max} \leq s$.

If $W^{\max}$ were zero, the proof will be trivial, since in this case the next hyperperiod will start with zero backlog and all the generated workload, even for the worst case, would be less than T, so all subsequent hyperperiods will start with zero backlog too.

Let us assume that $W^{\max}$ is positive. This only can happen if all the spare time s occurred before the last busy period [2], as is easily understood looking at fig. 3. So, the next hyperperiod will "absorb" the backlog $W^{\max}$ before arriving to the last busy

[2] A busy period is a contiguous interval of time in which the backlog is positive



For any instant t, the sum of the workload until that instant (vertical strokes) plus the sum of the idle time until that instant (horizontal strokes), is equal to t plus the backlog at that instant. For simplicity, the figure do not show the jobs arriving in the next hyperperiod.

Figure 4: Graphical proof for theorem 1

period (because $W^{\max} \leq s$, as seen). This way, the backlog distribution at the end of the hyperperiod is not affected by the backlog distribution at the beginning, so it will be the same than in the previous hyperperiod, and it will repeat identically at the end of all the subsequent hyperperiods. □

The above theorem gives us the justification for restricting the analysis to a single hyperperiod. Since the initial backlog of a hyperperiod is equal to the final backlog of the preceding one, all hyperperiods from the second complete hyperperiod on present the same statistical distribution of the initial backlog. The stochastic analysis of the response time will produce the same results in all these hyperperiods. The first non-complete hyperperiods are non representative with respect of the steady state, so it is not necessary to analyze them.

Then, the general algorithm for analyzing this kind of systems would be:

1. Compute the instant in which the first complete hyperperiod begins, i.e., the value of k_0. This value exclusively depends on the offsets of the tasks. Note that if the offset is zero for all the tasks, the first complete hyperperiod begins at $t = 0$, so $k_0 = 0$ in this case.

2. Compute the backlog distribution at instant $(k_0 + 1)T$, using the method "convolve and shrink". Note that for the case in which all offsets are zero, the result will be zero, so this step can be skipped.

3. Perform the stochastic analysis of the response time over a single hyperperiod, using as initial workload the distribution obtained in the previous step

3.3. General case

For the general case, we will assume $U^{\max} > 1$, since if $U^{\max} \leq 1$ the solution can be obtained in a simpler way, by using the method explained in section 3.2.

When $U^{\max} > 1$, there exists a non null probability of the workload generated in a hyperperiod being greater than the hyperperiod length. This means that there exists a non null probability of the next hyperperiod starting with an initial backlog greater than the initial backlog of the previous hyperperiod. As a consequence, the backlog distribution is different among hyperperiods.

We will prove in this section that the backlog observed at intervals of length T, i.e., the sequence of random variables $\{\mathcal{W}_i\}$, is a *Markov chain*. We will find that, in order to this chain being stable, the average utilization $\bar{U}$ has to be less than one. If this condition is met, the distribution of each $\mathcal{W}_i$ converges towards a steady-state distribution as i increases. We will find the general form of this distribution and provide methods for numerically computing it.

3.3.1. The initial backlog is a Markov chain

A Markov chain is defined as a sequence of random variables, such that the PF of each one only depends on the PF of the immediately previous one, and not on the PF of the others. This property is often called the "memoryless" property of the Markov chain.

The probability function of a discrete random variable $\mathcal{X}$ can be stored as a vector, $\boldsymbol{x}$, with an infinite number of components, such that the i-th component of this vector is the probability of $\mathcal{X}$ being equal to i. This notation allows for writing the Markovian property as a matrix equation. If the sequence of random variables $\{\mathcal{X}_k\}$ is a Markov chain, then:

$$\boldsymbol{x}_{k+1} = \mathbf{P}\boldsymbol{x}_k$$

Where $\boldsymbol{x}_k$ is the PF of $\mathcal{X}_k$, stored in a vector as explained, and $\mathbf{P}$ is the so called "transition matrix", or "Markov matrix". Given the PF of one of the random variables in the sequence, it is possible to obtain the PF of the next one, by multiplying by $\mathbf{P}$ matrix, and from this, the next one can be computed, an so on. The key fact for which the sequence is a Markov chain is that $\mathbf{P}$ is always the same matrix and do not depends on k. We will see now that this is the case for the random variable $\mathcal{W}_k$, i.e, the initial backlog in the k-th hyperperiod.

Definition 7. *We will call* ***backlog process*** *of a system S to the sequence of random variables $\{\mathcal{W}_k\}$, for $k = k_0, k_0+1, k_0+2, \ldots$*

Theorem 2. *The backlog process of any system S is a Markov chain.*

Proof. The probability of $\mathcal{W}_k$ being equal to any given value i can always be expressed in function of the probabilities of $\mathcal{W}_{k-1}$, by using conditional probabilities as follows:

$$\mathbb{P}\{\mathcal{W}_k = i\} = \sum_j \mathbb{P}\{\mathcal{W}_{k-1} = j\} \cdot \mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\} \tag{7}$$

The above expression is always true, with independence of $\mathcal{W}_k$ being or not a Markov chain. Now then, the term $\mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\}$ represents the probability of having a backlog of i units at the end of the hyperperiod $k-1$, *for the case* in which the backlog was j units at the beginning of that hyperperiod. This probability is obtained from the activation sequence of the jobs, and the PFs of their execution times. Since, from the first complete hyperperiod on, the sequence of job activations is the same for all hyperperiods, and also it is the same their execution time PFs, the conditional probabilities are the same in all hyperperiods, and thus equal to those on the first complete hyperperiod. That is:

$$\mathbb{P}\{\mathcal{W}_k = i | \mathcal{W}_{k-1} = j\} = \mathbb{P}\{\mathcal{W}_{k_0+1} = i | \mathcal{W}_{k_0} = j\} \quad \text{for } k > k_0$$

If we represent this probability by $b_j(i)$, eq. (7) gives:

$$\mathbb{P}\{\mathcal{W}_k = i\} = \sum_j \mathbb{P}\{\mathcal{W}_{k-1} = j\} \cdot b_j(i) \qquad \text{for } k > k_0 \tag{8}$$

In this equation it can be observed that the probabilities of $\mathcal{W}_k$ only depend on the probabilities of $\mathcal{W}_{k-1}$, and not on those of the previous one. Coefficients $b_j(i)$ are always the same, and do not depend on k (they only depend on the system parameters). Then, the sequence $\mathcal{W}_k$ is a Markov chain. □

3.3.2. Markov matrix computation

Equation (8) can also be written in matrix form:

$$\mathbf{b}_k = \mathbf{P}\mathbf{b}_{k-1} \tag{9}$$

Where $\mathbf{b}_k$ is the probability function of $\mathcal{W}_k$, expressed as a column vector (the i-th row in this vector contains the probability of $\mathcal{W}_k$ being equal to i), and $\mathbf{P}$ is the transition matrix, defined as:

$$\mathbf{P}(i, j) = b_j(i)$$

That is, each column $\mathbf{P}(\cdot, j)$ in matrix $\mathbf{P}$ contains the PF of the backlog at the end of the first complete hyperperiod, *for the case* in which the initial backlog was j units for that hyperperiod.

The definition of $\mathbf{P}$ gives us a mechanism for computing the components of $\mathbf{P}$. In fact, to obtain the components of the j-th column, it suffices to apply the algorithm "convolve and shrink" among the first complete hyperperiod, assuming an initial workload of j units. When the end of the hyperperiod is reached, the obtained PF will be the j-th column of $\mathbf{P}$. Note that this implies that each column in $\mathbf{P}$ has a sum equal to 1, since each column is a probability function.

This way, each column has a finite number of non-null elements, because it is the result of a finite sequence of "convolutions and shrinkages", and the PFs involved in the convolutions have a finite number of points. However, the very matrix $\mathbf{P}$ is infinite, because it requires an infinite number of columns to be fully specified. This poses an apparent impossibility for completely writing this matrix. However, it happens that

the columns present a regularity, and this will allow to write the *general form* of the matrix. Using this form, the complete matrix will be specified from a finite number of numerical values.

The general form of $\mathbf{P}$, as we will prove in short, follows this schema:

$$\mathbf{P} = \left(\begin{array}{cccc|cccc} b_0(0) & b_1(0) & b_2(0) & \dots & b_r(0) & 0 & 0 & 0 \\ b_0(1) & b_1(1) & b_2(1) & \dots & b_r(1) & b_r(0) & 0 & \vdots \\ b_0(2) & b_1(2) & b_2(2) & \dots & b_r(2) & b_r(1) & b_r(0) & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots & b_r(2) & b_r(1) & \ddots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & b_r(2) & \ddots \\ b_0(m_r) & b_1(m_r) & b_2(m_r) & \dots & b_r(m_r) & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & \dots & 0 & b_r(m_r) & \vdots & \ddots \\ 0 & 0 & 0 & \dots & 0 & 0 & b_r(m_r) & \ddots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \ddots \end{array}\right) \tag{10}$$

We see that the matrix can be divided in two parts. The left part, shaded in dark gray and composed of the first r columns from $\mathbf{P}(\cdot,0)$ to $\mathbf{P}(\cdot,r-1)$ has not structure. That is, the coefficients are in general different, although all these columns have at maximum m_r non null elements[3]. The right part, starting in column $\mathbf{P}(\cdot,r)$ presents a regularity: the same coefficients are repeated in each column, copied from the previous column and shifted one position down. This regularity has been shaded with pale gray. In the next theorem we will show that this structure appears always in $\mathbf{P}$, for any given system.

Theorem 3. *There exist two integers r y m_r such that:*

$$\mathbf{P}(i,j) = \begin{cases} b_j(i) & j < r, \quad i \leq m_r \\ \mathbf{P}(i-1,j-1) & j \geq r, \quad i \in [j-r, j-r+m_r] \\ 0 & \textit{for any other } i,j \end{cases} \tag{11}$$

Proof. First of all, it has to be noted that expression (11) is only a different way of mathematically enunciating the regular structure of $\mathbf{P}$ already shown in eq. (10).

Let r be the maximum amount of time in which the system can be idle, in any hyperperiod after the first complete hyperperiod. We will call r the "maximum spare time". This quantity can be computed with the formula:

$$r = T + W^{\min} - \sum_i \frac{T}{T_i} C_i^{\min} \tag{12}$$

where T is the hyperperiod length, $W^{\min}$ is the backlog at the end of the first complete hyperperiod for the case in which all jobs require their minimum execution time, and $C_i^{\min}$ is the minimum execution time of task τ_i.

[3] Of course, in these columns any of the elements $b_j(i)$ can also be zero

The r-th column in $\mathbf{P}$ represents the backlog probability function at the end of a complete hyperperiod, for the case in which the initial backlog was r units. But in this case, the initial backlog is equal to the maximum spare time, so it is guaranteed that there are not idle periods in this hyperperiod. That is, the whole hyperperiod will be a busy period. This is also true for any initial backlog greater than r. In these cases, there is no instant in the hyperperiod in which the backlog reaches zero, so the final backlog is simply the sum of the initial backlog, plus all the workload generated in the hyperperiod, minus the hyperperiod length. That is, if $\mathcal{W}(k_0 T) = r$, then $\mathcal{W}((k_0+1)T)$ can be computed simply as:

$$\mathcal{W}((k_0+1)T) = r - T + \sum_i \frac{T}{T_i} \mathcal{C}_i \tag{13}$$

Now then, since the sum is composed of random variables, the final backlog is also a random variable. Its PF can be obtained by convolution of all the execution time PFs, for all the jobs activated in the hyperperiod. Note that the term $r - T$ is not random, but deterministic, so it represents a shift to the right of the resulting PF.

To summarize, the r-th column in matrix $\mathbf{P}$ is computed by convolving the execution time PFs of all the jobs in the hyperperiod, and shifting to the right the result by $r - T$ units. The $(r+1)$-th column is computed the same way, with the only difference that the final shifting has to be $r + 1 - T$. And so on for all columns from the r-th onwards. This proves that all these columns have the same coefficients, but shifted one position down.

In addition, the r-th column has a finite number of non-null coefficients. If we call m_r to the index of the last one, then m_r represents the backlog at the end of a hyperperiod such that all the jobs require their worst execution time, and the initial backlog is equal to r (the maximum spare time). It is self evident that, if the initial backlog were less than r, the final backlog will be less than or equal to m_r. This proves that all columns previous to the r-th column have zeros from the row m_r onwards.

From the two remarks above, the theorem follows. □

Then, the complete matrix $\mathbf{P}$ can be specified from a finite number, $(r+1) \times (m_r + 1)$, of coefficients. These coefficients can be obtained by the "convolve and shrink" method, by computing the backlog at the end of the hyperperiod, assuming an initial backlog deterministically equal to $i = 0, 1, 2, \ldots, r$.

3.3.3. Stationarity and convergence

A Markov chain, defined by a matrix $\mathbf{P}$, can be of the *recurrent* type if $\mathbf{P}$ fulfills some properties (to be stated later). If it does, it has been proven that there exist a vector $\boldsymbol{\pi}$, unique except for multiplicative factors, such that:

$$\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi} \tag{14}$$

If, in addition, the chain is of *positive* type, then $\boldsymbol{\pi}$ is summable, so it can be normalized (divided by it sum), and can be considered this way as a probability function. If

the chain is not positive, the sum of $\boldsymbol{\pi}$ would be infinite, and this case would not have practical interest.

Moreover, the stochastic processes theory not only proves the existence of $\boldsymbol{\pi}$, but also that, for *positive recurrent* chains, the probability function of the Markov chain "converges" towards this vector. That is, if the Markov chain $\{\mathcal{X}_i\}$ is positive recurrent, then $f_{\mathcal{X}_i}(\cdot) \to \boldsymbol{\pi}$, when $i \to \infty$.

This is very interesting for our analysis, because it means that, if our matrix $\mathbf{P}$ fulfill certain properties, then the chain would be positive recurrent. Then it would exist a *stationary backlog distribution*, i.e., a statistical distribution of the backlog such that, if the initial backlog follows that distribution, the backlog at the end of the hyperperiod would also follow that distribution. Moreover, if this stationary backlog distribution exists, the system tends towards it; after a sufficient number of hyperperiods, the backlog PF at the beginning of each hyperperiod would be practically the same, and equal to the stationary distribution.

Existence and uniqueness of the stationary distribution In this section we will prove with rigor that, if $\bar{U} < 1$, the Markov chain is positive recurrent. This is why these kind of systems shows a tendency towards a stable distribution. We will give the general form of this distribution, and provide methods for computing it numerically.

The condition $\bar{U} < 1$ is easily understandable for the analyst, and it its easy to verify from the system parameters. However, in order to prove the recurrence and positivity of the Markov chain, it would be preferable to have a condition over the coefficients of $\mathbf{P}$. Next proposition will show that condition $\bar{U} < 1$ is equivalent to say that the distribution given by the r-th column in $\mathbf{P}$ has a mean less than r. This condition, although looks more strange, is more convenient for the proof of the recurrence and positivity of the Markov chain.

Proposition 1. *If $\mathbf{P}(i,j) = b_j(i)$ is the transition matrix of the Markov chain $\{\mathcal{W}_k\}$, which represents the backlog at the beginning of each hyperperiod for a system S with average utilization $\bar{U}$, the following equivalences are true:*

- $\bar{U} < 1 \iff \sum_i i b_r(i) < r$
- $\bar{U} = 1 \iff \sum_i i b_r(i) = r$
- $\bar{U} > 1 \iff \sum_i i b_r(i) > r$

Proof. Let $\bar{B}_r$ denote the expected value for the random variable whose PF is given in the r-th column of $\mathbf{P}$. This value can be computed from the coefficients of the r-th column of $\mathbf{P}$, using the statistical definition of the expected value:

$$\bar{B}_r = \sum_i i\mathbf{P}(i,r) = \sum_i i b_r(i) \tag{15}$$

In addition, the r-th column is special, because r is the "maximum spare time" in the system. If one hyperperiod begins with a backlog of r units, it will be busy the whole hyperperiod, even in the best case scenario. So in this case the final backlog will be

equal to the sum of all the execution times, plus the initial backlog, minus the elapsed time T, as already seen in eq. (13). The expected value of the final backlog is then equal to the sum of the expected values of the terms composing it. Then, the expected value can also be computed by the following equation:

$$\bar{B}_r = r - T + \sum_i \frac{T}{T_i} \bar{C}_i$$

If T is drawn from the sum (since it is constant), what remains in the sum is the system average utilization, as was defined in (6), so:

$$\bar{B}_r = r - T + T\bar{U} = r + T(\bar{U} - 1)$$

From this equation, the equivalences stated in the proposition can be trivially proven. □

Next theorem shows that the Markov matrix is positive recurrent if $\bar{U} < 1$. However, for this proof it is necessary to add the additional hypothesis of the chain being *irreducible*. We will explain later (page 26) the meaning of this additional hypothesis, and we will show that it is possible imagine systems for which this condition is not met, and even that this kind of systems is very common. However, we will also show that, even if the irreducibility hypothesis does not hold, we can still use the condition $\bar{U} < 1$ in order to classify the chain as positive recurrent.

Theorem 4. *Let S be a system composed by a set of periodic tasks $\{\tau_i\}$, and let $\{\mathcal{W}_k\}$ be the backlog process for that system, which is a Markov chain. If this chain is irreducible, and the system S fulfills $\bar{U} < 1$, then the chain is positive recurrent.*

Proof. Recent advances in the field of the Markov chains theory [Meyn and Tweedie, 1993; Tweedie, 2000] allows us to say that, in order to prove that an irreducible Markov chain is positive recurrent, it suffices to prove that the coefficients in the Markov matrix fulfill certain *drift conditions*.

In particular, an irreducible Markov chain is recurrent if and only if it exists a non-negative function $V(x)$, $x \in \mathbb{Z}^+$, $V(x) \to \infty$ when $x \to \infty$, and a number $N \geq 0$ such that

$$\sum_j \mathbf{P}(j,x)V(j) \leq V(x) \qquad x > N \tag{16}$$

In addition to being recurrent, the irreducible chain would be also positive if and only if there exists a non-negative function $V(x), x \in \mathbb{Z}^+$ and a pair of numbers $N \geq 0, \epsilon > 0$ such that:

$$\sum_j \mathbf{P}(j,x)V(j) \leq V(x) - \epsilon, \qquad x > N; \tag{17}$$

$$\text{y} \quad \sum_j \mathbf{P}(j,x)V(j) < b < \infty, \qquad x \leq N. \tag{18}$$

We will show next that condition $\sum_i ib_r(i) \leq r$ implies recurrence, and furthermore, condition $\sum_i ib_r(i) < r$, implies positive recurrence. Since these conditions are respectively equivalent to $\bar{U} \leq 1$ and $\bar{U} < 1$, this would complete the proof of the theorem.

In order to prove that condition (16) holds, it suffices to take $V(x) = x$ and $N = r$. In fact, thanks to the repetitive structure of $\mathbf{P}$, from column r onwards the coefficients repeat, so we can write

$$\mathbf{P}(j,x) = b_r(j+r-x) \qquad x > r$$

If this fact is put in the drift condition (16), along with the function $V(x) = x$ and the integer $N = r$, we obtain

$$\sum_j b_r(j+r-x) \cdot j \leq x \qquad x > r$$

By operating in this inequality, moving x to the first member and adding r to both:

$$(r-x) + \sum_j jb_r(j+r-x) \leq r \qquad x > r$$

In addition, since $b_r(\cdot)$ is a probability function, its sum is 1, so

$$(r-x)\sum_j b_r(j+r-x) + \sum_j jb_r(j+r-x) \leq r \qquad x > r$$

And taking out the common factor

$$\sum_j (j+r-x)b_r(j-x+r) \leq r \qquad x > r$$

Through a variable change it is easy to see that the inequality to which we have arrived is equivalent to $\sum_i ib_r(i) \leq r$, so if this inequality holds, the chain is recurrent, q.e.d.

In order to prove that the chain is also positive, we have to check if conditions (17) and (18) are met. On one hand, it is trivial to show, following the same steps than above, that condition (17) is equivalent to $\sum_i ib_r(i) < r$. On the other hand, condition (18) is equivalent to $\sum_i ib_x(i) < b < \infty$ for all $x \leq r$. This condition is demanding that the expected value for any of the first r columns has to be bounded by a finite number b. This condition is met due to the special structure of our matrix $\mathbf{P}$, because it has zeros from row m_r onwards in all the first r columns, as it was seen in (10).

Therefore, when $\sum_i ib_r(i) < r$ both conditions are met, and thus the chain is positive recurrent. In addition, proposition 1 showed this condition equivalent to $\bar{U} < 1$, so the proof of the theorem is complete. □

The problem of irreducibility In Markovian theory terminology, the expression "state i communicates with state j" means that, from state i it is possible to reach state j in a finite number of transitions. Translating this to the language of our domain, an initial backlog i "communicates" with a different initial backlog j when, given an initial backlog of i units for one hyperperiod, there exist a non-null probability of finding a hyperperiod in the future whose initial backlog is j units.

A Markov chain is said **irreducible** when all states communicate with each other. In the language of our problem, this hypothesis implies that any value of the initial backlog should be possible among the different hyperperiods, no matter how the initial backlog was at the time origin.

We will show some counter-examples, in which it can be seen that it is easy to envision systems for which this hypothesis do not hold, and that they are not rare, indeed.

A trivial case of non-irreducible system is one with $U^{\min} > 1$. In this system, the backlog at the end of each hyperperiod is always increasing. Then, if we start with an initial backlog of 1 unit, for example, the system will never reach in the future a backlog of 0 units. Then, state 1 does not communicate with state 0, and thus the chain is not irreducible. However, this case has not practical value, because a system with $U^{\min} > 1$ is clearly unrealistic.

A more interesting and less trivial example is the following. Consider a system composed of a single task, whose parameters are:

- *Offset*, $\Phi_1 = 3$,
- Period, $T_1 = 4$,
- Execution time, $\mathcal{C}_1$ can only take the values 2 or 6, both with equal probability (but any value between 3 and 5 has zero probability)
- Deadline and priority are not relevant for the example.

Figure 5 shows the activation pattern for this "system". The hyperperiod length, obviously, coincides with the period of the single task in the system. The first complete hyperperiod appears for $k_0 = 0$.

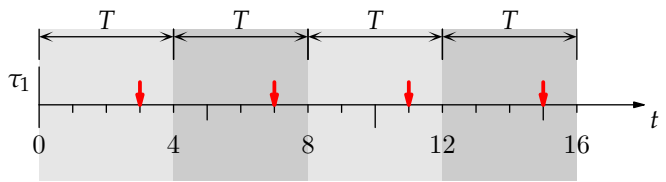


Figure 5: A simple example of a non-irreducible system

It can be seen that, assuming an initial backlog of zero, at the end of the first hyperperiod (i.e., at $t = 4$), the backlog can only take the values 1 or 5 (because the task activated at $t = 3$ can only take the execution times 2 or 6). Then, from state 0, we can only reach states 1 and 5.

If the initial backlog is 1, the situation is the same, because 1 unit of initial backlog is suck on the 3 units gap existent before the task activation. So, from state 1 we can reach again states 1 and 5.

If the initial backlog is 5, when the task arrives there are still 2 units of backlog, which will be added to the execution time of the task, so the final backlog can be 3 or 7 units. So, from state 5 we can only reach states 3 or 7.

Finally, if the initial backlog is 3, it will be suck on before the task arrival, so this is the same case than the zero initial backlog. So, from state 3 we can only reach states 1 or 5.

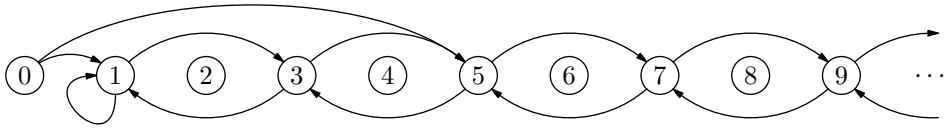


Figure 6: Inter-state communication, starting from state zero, for the non-irreducible example

In figure 6 the general transition pattern starting from state zero is shown. From zero, only states 1 and 5 can be reached, and from an odd state, only another odd state can be reached. Any even state is unreachable from zero, so the chain is non-irreducible. It can be seen also that, starting from an odd value for the initial backlog, it is impossible to reach an even or zero backlog. Another additional reason of non-irreducibility.

This situation is not as artificial as it may seem. It may appear if the execution times of the tasks have "forbidden values", i.e., values with zero probability. This may cause the appearing of forbidden values also for the backlog, and in this case the backlog can only evolve taking values in a certain subset of the integers (as in the previous example, in which the subset was the odd integers).

What if the initial backlog were 4, for example? It can be seen that, in this case, the final backlog can only take the values 2 or 6. Apparently, from an even state it is possible to reach other even states. However, note that, once the state 2 is reached (and it will be reached certainly, because the backlog has a non-null probability of decreasing), the chain will get "trapped" in the subset of the odd states. In fact, if the backlog ever reaches the value 2, these 2 units will be suck on in the 3 units of spare time existent at the beginning of the next hyperperiod, and then the situation is the same than that with zero initial backlog. From state 2, only states 1 and 5 are reachable. Figure 7 shows the complete state transition diagram.

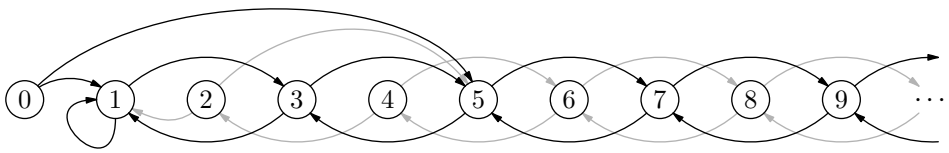


Figure 7: Inter-state communication, starting from any state, for the non-irreducible example

It is appropriate to insist on the fact that, starting from an even state greater than 2, the chain will "move" for a while among the even states, but eventually it will "fall" into state 2, and then it will be unable to leave the sequence of odd states. If the system is running for an infinite amount of time, the fraction of time spent in the even states is negligible in front of the total time. The long term behavior of the system is dominated by the chain evolving only among odd states. This is a crucial idea.

In general, we will show that for any system with $\bar{U} < 1$, it is possible to find a integer set, C, such that for any pair of integers i, j in C, state i communicates with state j. In the above example, C is the set of the odd numbers. If a new Markov chain is built, restricted to C, this new chain will be irreducible, so theorem 4 would apply. In addition, we will show that set C is "absorbent", which means that if the initial state of the system is not in C, it is guaranteed anyway (with probability 1) that after a sufficient number of transitions (hyperperiods) the state of the system will be in C. If the restricted chain was recurrent positive, the original unrestricted chain will also adopt a stationary distribution, once entering in the state set C.

We will show now how to build the set C. It has to be stressed that the interest of this section is purely academic, because later on we will give a method for computing the stationary distribution $\boldsymbol{\pi}$ which does not require knowledge of the irreducibility of the system, and far less to obtain beforehand the set C. This section is only aimed to convince the reader that the fact of the original chain being irreducible or not has not practical importance, because the existence of C and its "absorbency" property guarantee that the steady state regime will be reached anyway, also for non-irreducible systems, provided that $\bar{U} < 1$.

Let $W^{\min}$ be the minimum value that the backlog can reach at the end of any hyperperiod. We can compute this value by studying the final backlog in a complete hyperperiod in which all jobs take their minimum computation time, and the initial backlog is zero. For example, for the system of fig. 5, the value of $W^{\min}$ would be 1, which is obtained for the case in which the task demands its minimum execution time (2), and the initial backlog is zero.

The value of $W^{\min}$ represents an state which is reachable from any other state. This is because, under the hypothesis $U^{\min} < 1$, there exist a non-null probability of the backlog to decrease in one hyperperiod, and therefore there exist a non-null probability of the backlog decreasing during n consecutive hyperperiods, until the value $W^{\min}$ is reached from whatever initial backlog. We define C as the set of all states which are reachable from the state $W^{\min}$. In the above example, this would be the set of all odd integers. It is clear that all states in C communicate mutually (any state is reachable from any other going through state $W^{\min}$).

Now, a new Markov chain can be built evolving only in the state space defined by C. In order to define this new chain, a new matrix $\mathbf{P}_C$ can be written, from the original matrix $\mathbf{P}$, by taking from $\mathbf{P}$ only the elements whose subindexes (column and row) are in C. Let us call $g(\cdot)$ to the function which maps the indexes of $\mathbf{P}_C$ onto the indexes of $\mathbf{P}$. In other words, element $\mathbf{P}_C(i,j)$ is a copy of element $\mathbf{P}(g(i), g(j))$.

In the example shown in fig. 5, we will have the following situation: C is the set of all odd integers; matrix $\mathbf{P}_C$ is built by taking the odd rows and columns from matrix $\mathbf{P}$; and

function $g(\cdot)$ is defined as $g(x) = 2x + 1$.

The Markov chain defined by matrix $\mathbf{P}_C$ is irreducible, because all states in C communicate mutually. In addition, it is easy to show that, if the original system defined by $\mathbf{P}$ fulfilled the condition $\bar{U} < 1$, then the matrix $\mathbf{P}_C$ will fulfill the drift conditions given in (17) and (18). It suffices to use $V(x) = g(x)$ as drift function, and taking N equal to the first integer such that $g(N) \geq r$.

From the above, we can conclude that, if the system has $\bar{U} < 1$, then the Markov matrix restricted to C is irreducible, recurrent and positive. Therefore, the restricted chain will have a stationary distribution, $\boldsymbol{\pi}_C$ such that $\boldsymbol{\pi}_C = \mathbf{P}_C\boldsymbol{\pi}_C$. From this vector $\boldsymbol{\pi}_C$ a stationary distribution $\boldsymbol{\pi}$ can be built for the original unrestricted matrix $\mathbf{P}$, in the following way:

$$\boldsymbol{\pi}(x) = \begin{cases} \boldsymbol{\pi}_C(g^{-1}(x)) & \forall x \in C, \\ 0 & \forall x \notin C. \end{cases} \tag{19}$$

In fact, this function $\boldsymbol{\pi}(\cdot)$ is a probability function, since its sum is 1 (because the sum of $\boldsymbol{\pi}_C$ was 1). In addition, when $\bar{U} < 1$, the backlog has a tendency to decrease, and the probability of the backlog eventually reaching the value $W^{\min}$ is 1. Since this value is i nC, it is guaranteed that, waiting enough, the Markov chain will enter in C, and then it will not be able to leave C. Therefore, in the long run, the probability of the chain being out of C is zero, because this kind of states will be visited only a finite number of times, before entering forever in C (which will be visited an infinite number of times). This is why we define $\boldsymbol{\pi}(x)$ as zero, for any x out of C. However, if $x \in C$, the probability of being in state x will be given by $\boldsymbol{\pi}_C(\cdot)$, after "undoing" the index transformation through function $g(\cdot)$.

Applying these ideas to the example in fig. 5, the stationary backlog for that system could be found in the following way:

1. Build the matrix $\mathbf{P}_C$, by taking from $\mathbf{P}$ only the elements with both indexes odd.
2. Obtain the stationary distribution $\boldsymbol{\pi}_C$ of matrix $\mathbf{P}_C$ (in the next section we will attack this problem)
3. Build the stationary distribution $\boldsymbol{\pi}(x)$ from $\boldsymbol{\pi}_C(\cdot)$, by assigning zero to all the points with even x, and assigning $\boldsymbol{\pi}_C((x-1)/2)$ to all the points with odd x.

Fortunately, all the above is not necessary in practice. The above method was presented from a purely academic point of view. If, instead, we use the method which will be presented in the next section to directly compute the stationary distribution of $\mathbf{P}$, forgetting about the irreducibility of the Markov chain, we will obtain a distribution $\boldsymbol{\pi}$ which automatically will have zero in the right places (those corresponding to indexes out of C). The only condition $\bar{U} < 1$ guarantees that there exist a unique stationary distribution, so we can compute it directly, without needing to restrict the chain to the set C. This is very convenient, because in general the determination of the set C can be an extremely difficult task.

Task	Offset	Period	Priority	Execution time ($\mathcal{C}_i$)
τ_1	0	4	High	$\{1,2\}$ equi-probable
τ_2	0	6	Low	$\{2,3,4\}$ with prob. 0.2, 0.3 and 0.5, respect.

Table 2: Example system for computing the backlog steady-state distribution

3.3.4. Computation of the steady-state distribution

In the precedent section we have show that, if the system has $\bar{U} < 1$, then there exist a unique vector $\boldsymbol{\pi}$, whose sum is 1, and such that

$$\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$$

Now the question is, would it be possible to know the values of the components of this vector? The above equation suggests that $\boldsymbol{\pi}$ would be an eigenvector of matrix $\mathbf{P}$, corresponding to the eigenvalue 1. If $\mathbf{P}$ were a finite matrix, say of dimension $N \times N$, then, the above matrix equation will give rise to a set of N equations and N unknowns (the N components of $\boldsymbol{\pi}$), and solving this system the desired answer would be obtained.

Unfortunately, matrix $\mathbf{P}$ is infinite, so if we try to develop the above matrix equation, we will obtain a system with infinite equations and infinite unknowns, which cannot be solved.

In this section we will present a general method which allows for the obtaining of the infinite coefficients of $\boldsymbol{\pi}$. This method, will provide in numerical form the first few elements, along with a closed form expression which allows to compute any of the remaining elements. The interest of this equation resides in that it shows the general form of $\boldsymbol{\pi}$, for any given system. Additionally, if the system is small, this method can be applied to compute as many elements of $\boldsymbol{\pi}$ as desired. However, for big systems, the method complexity can be too high (due to the high number of equations to solve), and it could be preferable to apply some of the approximated methods shown in section 4.

Instead of providing a formal description of the method, which would be too involved, we will give an example which will be solved step by step. At appropriate places, it will be explained how the example could be generalized to any other system. The example system is shown in table 2.

The hyperperiod length for this system is 12, and the activation pattern is shown in figure 8.

Since all the task have zero offset, the first complete hyperperiod starts at $t = 0$. The $U^{\max}$ of the system is $16/12$, which is greater than one, so the backlog at the end of the first hyperperiod will have different distribution than the initial backlog. Therefore, we cannot simply analyze the first hyperperiod and assume that the same results apply to any other hyperperiod. Instead we have to find the steady-state distribution of the backlog. The average utilization, $\bar{U}$, of this system is 0.925, not greater than1, so according with theorem 4 the system will have a unique steady-state distribution such that $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$.

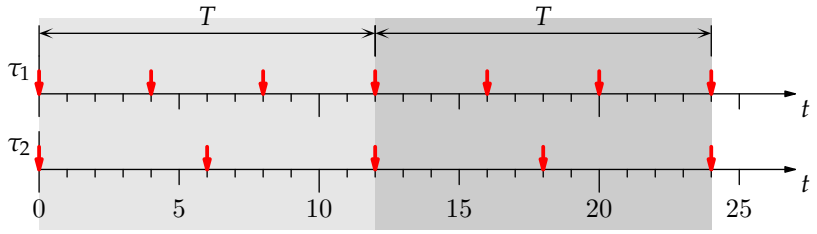


Figure 8: Activation pattern for the example

The first step is to compute the matrix **P**. To this end, it suffices to apply the method "convolve and shrink" along the first hyperperiod, starting at $k = 0$, and stopping at $t = 12$, and using as initial backlog a deterministic workload of k units. This way, the PF of the backlog obtained for $t = 12$ will be the k-th column of matrix **P**. According with theorem 3, matrix **P** has a regular structure, and we need only to compute $r + 1$ columns. The value of r was defined in eq. (12) which is repeated here for the reader convenience:

$$r = T + W^{\min} - \sum_i \frac{T}{T_i} C_i^{\min}$$

In the example we are solving, $r = 12 + 0 - (3 \times 1 + 2 \times 2) = 5$, so in order to completely describe **P** we need to compute only the columns with indexes 0 to 5. The result is the following (dashes represent zeros):

$$\mathbf{P} = \left(\begin{array}{ccccc|cc|c} 0.8375 & 0.595 & 0.3275 & 0.13125 & 0.035 & 0.005 & - & \dots \\ 0.13125 & 0.2425 & 0.2675 & 0.19625 & 0.09625 & 0.03 & 0.005 & \ddots \\ 0.03125 & 0.13125 & 0.2425 & 0.2675 & 0.19625 & 0.09625 & 0.03 & \ddots \\ - & 0.03125 & 0.13125 & 0.2425 & 0.2675 & 0.19625 & 0.09625 & \ddots \\ - & - & 0.03125 & 0.13125 & 0.2425 & 0.2675 & 0.19625 & \ddots \\ - & - & - & 0.03125 & 0.13125 & 0.2425 & 0.2675 & \ddots \\ - & - & - & - & 0.03125 & 0.13125 & 0.2425 & \ddots \\ - & - & - & - & - & 0.03125 & 0.13125 & \ddots \\ - & - & - & - & - & - & 0.03125 & \ddots \\ - & - & - & - & - & - & - & \ddots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots \end{array}\right) \quad (20)$$

We have computed also column 6, in order to verify that, as expected, it is identical to column 5, only shifted one row down. Column 5 has 8 non-null elements, so $m_r = 7$ in this example. In general, m_r will be equal to $r - T + \sum_i (T/T_i) C_i^{\max}$. As expected, none

of the first five columns has more than $m_r + 1$ non-null elements.

Let us remark another fundamental fact. Due to the repetition of the columns from r onwards, it also happens that the rows are repeated and shifted from row m_r onwards. Actually, in this particular example, this pattern at the row level appears before, at row 2 (starting the numbering from zero), but in the general case this kind of regularity can only be guaranteed after row m_r.

The problem, thus, consists on solving the equation $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$, being $\mathbf{P}$ the above matrix. If we develop the matrix equation into a system of equations, we will obtain an infinite number of equations and unknowns, which we call $\pi_0, \pi_1, \ldots$. They are the components of $\boldsymbol{\pi}$. However, only the first $m_r + 1$ are "truly different". Due to the row level regularity explained above, all equations from the m_r-th onwards have the same coefficients, but affecting different unknowns.

The system of equations can be divided in two parts. First, we have a set of $(m_r + 1)$ equations and $(m_r + r + 1)$ unknowns. In our example there are 8 equations and 13 unknowns. They are the following:

$$\begin{aligned}
\pi_0 &= 0.8375\pi_0 + 0.595\pi_1 + 0.3275\pi_2 + 0.13125\pi_3 + 0.035\pi_4 + 0.005\pi_5 \\
\pi_1 &= 0.13125\pi_0 + 0.2425\pi_1 + 0.2675\pi_2 + 0.19625\pi_3 + 0.09625\pi_4 \\
&\quad + 0.03\pi_5 + 0.005\pi_6 \\
\pi_2 &= 0.03125\pi_0 + 0.13125\pi_1 + 0.2425\pi_2 + 0.2675\pi_3 + 0.19625\pi_4 \\
&\quad + 0.09625\pi_5 + 0.03\pi_6 \\
&\vdots \\
\pi_7 &= 0.03125\pi_5 + 0.13125\pi_6 + 0.2425\pi_7 + 0.2675\pi_8 + 0.19625\pi_9 + 0.09625\pi_{10} \\
&\quad + 0.03\pi_{11} + 0.005\pi_{12}
\end{aligned} \tag{21}$$

Second, the remaining equations have in common the same general form, because the coefficients are repeated, only shifted to the right. We can write down all the remaining equations in a single expression, valid for $j > m_r + 1$:

$$\begin{aligned}
\pi_j = 0.03125\pi_{j-2} + 0.13125\pi_{j-1} + 0.24250\pi_j + +0.26750\pi_{j+1} + 0.19625\pi_{j+2} \\
+ 0.09625\pi_{j+3} + 0.03000\pi_{j+4} + 0.00500\pi_{j+5}
\end{aligned}$$

Note that the coefficients in this equation are the same than those in the r-th column of $\mathbf{P}$. As explained, these coefficients are the result of convolving the PFs of the execution time of all the jobs released in a complete hyperperiod.

Finding the last term (π_{j+5}) we obtain:

$$\begin{aligned}
\pi_{j+5} = -6.25\pi_{j-2} - 26.25\pi_{j-1} + 151.5\pi_j - 53.5\pi_{j+1} \\
- 39.25\pi_{j+2} - 19.25\pi_{j+3} - 6\pi_{j+4}
\end{aligned} \tag{22}$$

This equation is a recurrence relationship, which holds for any $j > m_r + 1$, in our example, $j > 8$. For example, for $j = 9$, this equation would allow us to compute π_{14} from $\pi_{13}, \pi_{12}, \ldots \pi_7$, if these first components were known. By applying the same

equation again for $j = 10$, we will find π_{15} from the previous 7 components, and so on. Therefore, if the first $m_r + r + 1$ components of $\boldsymbol{\pi}$ were known, we could find all the remaining components, by repeatedly applying eq. (22). Unfortunately we do not have still enough equations to determine the first components of $\boldsymbol{\pi}$.

We can exploit a fact about our system. Since $\bar{U} < 1$, we know for sure that $\boldsymbol{\pi}$ exists and it is unique. This will allow us to put some additional restrictions on the components of $\boldsymbol{\pi}$, in order to guarantee its summability. This way we will find additional equations which will allow us to solve the problem.

The first step is to rewrite eq. (22) in matrix form:

$$\begin{pmatrix} \pi_{j-1} \\ \pi_j \\ \pi_{j+1} \\ \pi_{j+2} \\ \pi_{j+3} \\ \pi_{j+4} \\ \pi_{j+5} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -6.25 & -26.25 & 151.5 & -53.5 & -39.25 & -19.25 & -6 \end{pmatrix} \begin{pmatrix} \pi_{j-2} \\ \pi_{j-1} \\ \pi_j \\ \pi_{j+1} \\ \pi_{j+2} \\ \pi_{j+3} \\ \pi_{j+4} \end{pmatrix} \tag{23}$$

Calling $\mathbf{Q}_j$ to the column vector $(\pi_{j-2}, \pi_{j-1}, \pi_j, \pi_{j+1}, \pi_{j+2}, \pi_{j+3}, \pi_{j+4})^\intercal$, and $\mathbf{A}$ to the coefficients matrix, the recurrence can be written concisely as $\mathbf{Q}_{j+1} = \mathbf{A}\,\mathbf{Q}_j$, valid for $j > 8$. Note that the dimension of matrix $\mathbf{A}$ is m_r, and that the last row in $\mathbf{A}$ can be trivially found from the coefficients of the r-th column in $\mathbf{P}$.

The matrix form allows us to see that $\mathbf{Q}_9 = \mathbf{A}\mathbf{Q}_8$, $\mathbf{Q}_{10} = \mathbf{A}^2\mathbf{Q}_8$, $\mathbf{Q}_{11} = \mathbf{A}^3\mathbf{Q}_8$, etc. In general:

$$\mathbf{Q}_n = \mathbf{A}^{n-8}\mathbf{Q}_8 \qquad \text{for } n > 8 \tag{24}$$

In order to compute easily $\mathbf{A}^{n-8}$, we will diagonalize this matrix. That is, we rewrite $\mathbf{A}$ in the form $\mathbf{A} = \mathbf{V}^{-1}\mathbf{D}\mathbf{V}$, where $\mathbf{V}^{-1}$ is a matrix whose columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_7$ are the eigenvectors of $\mathbf{A}$; matrix $\mathbf{D}$ is a diagonal matrix whose elements are $\lambda_1, \lambda_2, \ldots, \lambda_7$, the eigenvalues of $\mathbf{A}$; and matrix $\mathbf{V}$ is the inverse matrix of $\mathbf{V}^{-1}$. Once the diagonalization is done, the operation of raising $\mathbf{A}$ to any power is simplified, because it suffices to raise the matrix $\mathbf{D}$ to that power, i.e., $\mathbf{A}^x = \mathbf{V}^{-1}\mathbf{D}^x\mathbf{V}$. This leads to the following equation:

$$\begin{aligned} \mathbf{Q}_n &= \mathbf{V}^{-1}\mathbf{D}^{n-8}\mathbf{V}\,\mathbf{Q}_8 \\ &= c_1\lambda_1^{n-8}\mathbf{v}_1 + c_2\lambda_2^{n-8}\mathbf{v}_2 + c_3\lambda_3^{n-8}\mathbf{v}_3 + c_4\lambda_4^{n-8}\mathbf{v}_4 \\ &\quad + c_5\lambda_5^{n-8}\mathbf{v}_5 + c_6\lambda_6^{n-8}\mathbf{v}_6 + c_7\lambda_7^{n-8}\mathbf{v}_7 \end{aligned} \tag{25}$$

where c_i are real numbers given by equation

$$(c_1, c_2, c_3, c_4, c_5, c_6, c_7)^\intercal = \mathbf{V} \cdot \mathbf{Q}_8 = \mathbf{V} \cdot (\pi_6, \pi_7, \pi_8, \pi_9, \pi_{10}, \pi_{11}, \pi_{12})^\intercal \tag{26}$$

To find the eigenvalues of matrix $\mathbf{A}$, its characteristic polynomial has to be solved. This polynomial can be written in a very simple way, even without writing matrix $\mathbf{A}$ in advance, but directly using the coefficients of r-th column in $\mathbf{P}$. It can be easily shown that the general form of the characteristic polynomial is:

$$f(\lambda) = \Big(\sum_{i=0}^{m_r} b_r(i) \lambda^{m_r - i} \Big) - \lambda^{m_r - r} = 0 \tag{27}$$

Note that this polynomial has degree equal to m_r. In our example, after solving it, the following roots are found:

$$\begin{aligned}
\lambda_1 &= (-3.2976 + 1.825i) \\
\lambda_2 &= (-3.2976 - 1.825i) \\
\lambda_3 &= (-0.3099 + 3.0755i) \\
\lambda_4 &= (-0.3099 - 3.0755i) \\
\lambda_5 &= 1 \\
\lambda_6 &= 0.3476 \\
\lambda_7 &= -0.1325
\end{aligned}$$

Among these solutions always appears one with value 1, because all rows in $\mathbf{A}$ have sum equal to 1. Among the remaining solutions, some of them will have modulus greater than 1, and some other will have modulus less than 1. However, looking at eq. (25), it can be seen that this creates an apparent paradox. In fact, since all λ_i appear raised to $(n-8)$, if some of them are greater than 1, the corresponding terms will increase without limit as n increases, while those terms with $\lambda_i < 1$ will decrease, approaching to zero. Therefore, the vector $\boldsymbol{\pi}$ is not summable, i.e., its sum is infinite, because its components π_i tends to infinite as i increases. Then, the stationary distribution does not exist, which is a contradiction because we know for sure that it has to exist, because $U^{\max} < 1$ for this system.

This paradox disappears if the terms corresponding to $\lambda_i \geq 1$ disappear from eq. (25); that is, if the corresponding coefficients c_i are zero. Since c_i are still undetermined (they are unknowns in the problem), we can enforce some of them to be zero. This way we will obtain additional equations. In our example, the eigenvalues which have modulus greater than one are $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5, so, in order to guarantee the existence of $\boldsymbol{\pi}$, coefficients c_1, c_2, c_3, c_4 and c_5 must be zero.

This will originate 5 new equations. In order to write down these equations, matrix $\mathbf{V}$ is required, so it is necessary to find the eigenvectors of matrix $\mathbf{A}$. In our example the result is:

$$\mathbf{v}_1 = \begin{pmatrix} -0.00033 + 0.00004i \\ +0.00104 - 0.00073i \\ -0.00208 + 0.00430i \\ -0.00098 - 0.01798i \\ +0.03604 + 0.05751i \\ -0.22382 - 0.12387i \\ +0.96416 + 0.00000i \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} -0.00033 - 0.00004i \\ +0.00104 + 0.00073i \\ -0.00208 - 0.00430i \\ -0.00098 + 0.01798i \\ +0.03604 - 0.05751i \\ -0.22382 + 0.12387i \\ +0.96416 + 0.00000i \end{pmatrix}$$

$$\mathbf{v}_3 = \begin{pmatrix} +0.00089 - 0.00061i \\ +0.00161 + 0.00294i \\ -0.00954 + 0.00405i \\ -0.00951 - 0.03059i \\ +0.09704 - 0.01976i \\ +0.03069 + 0.30457i \\ -0.94623 + 0.00000i \end{pmatrix} \qquad \mathbf{v}_4 = \begin{pmatrix} +0.00089 + 0.00061i \\ +0.00161 - 0.00294i \\ -0.00954 - 0.00405i \\ -0.00951 + 0.03059i \\ +0.09704 + 0.01976i \\ +0.03069 - 0.30457i \\ -0.94623 + 0.00000i \end{pmatrix}$$

$$\mathbf{v}_5 = \begin{pmatrix} -0.37796 \\ -0.37796 \\ -0.37796 \\ -0.37796 \\ -0.37796 \\ -0.37796 \\ -0.37796 \end{pmatrix} \qquad \mathbf{v}_6 = \begin{pmatrix} -0.9376558 \\ -0.3258970 \\ -0.1132706 \\ -0.0393690 \\ -0.0136833 \\ -0.0047558 \\ -0.0016530 \end{pmatrix} \qquad \mathbf{v}_7 = \begin{pmatrix} -0.99118 \\ +0.13132 \\ -0.017398 \\ +0.002305 \\ -0.000305 \\ +0.000040 \\ -0.000005 \end{pmatrix}$$

These vectors are the columns of matrix $\mathbf{V}^{-1}$, and by computing the inverse of this matrix, we obtain matrix $\mathbf{V}$ (we have rounded the coefficients to the third decimal, in order to save space):

$$\mathbf{V} = \begin{pmatrix} 0.512 + 0.911i & 2.147 + 3.549i & -12.443 - 23.182i & 4.289 + 14.778i & 4.116 + 1.739i & 0.844 + 1.873i & 0.536 + 0.331i \\ 0.512 - 0.911i & 2.147 - 3.549i & -12.443 + 23.182i & 4.289 - 14.778i & 4.116 - 1.739i & 0.844 - 1.873i & 0.536 - 0.331i \\ 0.503 - 0.009i & 2.094 - 0.199i & -12.331 - 0.453i & 4.562 + 3.908i & 4.270 - 1.651i & 0.880 - 1.348i & 0.021 - 0.248i \\ 0.503 + 0.009i & 2.094 + 0.199i & -12.331 + 0.453i & 4.562 - 3.908i & 4.270 + 1.651i & 0.880 + 1.348i & 0.021 + 0.248i \\ +0.092 & +0.478 & -1.749 & -0.963 & -0.386 & -0.103 & -0.015 \\ -0.373 & -2.636 & +1.445 & +0.968 & +0.445 & +0.131 & +0.021 \\ -0.691 & +2.314 & -0.712 & -0.539 & -0.271 & -0.086 & -0.015 \end{pmatrix}$$

Once $\mathbf{V}$ has be computed, the matrix equation (26) can be written in form of set of equations:

$$\begin{aligned} c_1 &= (0.512 + 0.911i)\pi_6 + (2.147 + 3.549i)\pi_7 + (-12.443 - 23.182i)\pi_8 \\ &\quad + (4.289 + 14.778i)\pi_9 + (4.116 + 1.739i)\pi_{10} + (0.844 + 1.873i)\pi_{11} \\ &\quad + (0.536 + 0.331i)\pi_{12} \\ c_2 &= (0.512 - 0.911i)\pi_6 + (2.147 - 3.549i)\pi_7 + (-12.443 + 23.182i)\pi_8 \\ &\quad + (4.289 - 14.778i)\pi_9 + (4.116 - 1.739i)\pi_{10} + (0.844 - 1.873i)\pi_{11} \\ &\quad + (0.536 - 0.331i)\pi_{12} \\ &\vdots \\ c_5 &= 0.092\pi_6 + 0.478\pi_7 - 1.749\pi_8 - 0.963\pi_9 - 0.386\pi_{10} - 0.103\pi_{11} - 0.015\pi_{12} \\ c_6 &= -0.373\pi_6 - 2.636\pi_7 + 1.445\pi_8 + 0.968\pi_9 + 0.445\pi_{10} + 0.131\pi_{11} + 0.021\pi_{12} \\ c_7 &= -0.691\pi_6 + 2.314\pi_7 - 0.712\pi_8 - 0.539\pi_9 - 0.271\pi_{10} - 0.086\pi_{11} - 0.015\pi_{12} \end{aligned} \tag{28}$$

As explained before, to guarantee the existence of the solution $\boldsymbol{\pi}$, coefficients c_1, c_2, c_3, c_4 and c_5 must be zero. Using this knowledge in the above equations, the first five become a set of five additional restriction to be fulfilled by the first components of $\boldsymbol{\pi}$. These 5 equations, together with the 8 equations derived from the 8 first rows of the Markov matrix $\mathbf{P}$, give a total of 13 equations and 13 unknowns $(\pi_0, \ldots, \pi_{12})$, so apparently the system could be solvable.

However, this is not so, because one of the five equations just obtained is lineally dependent on the other. It relates to equation resulting from the restriction $c_5 = 0$. This coefficient affects the eigenvalue λ_5, which is 1, which is an eigenvalue which always appears and do not convey additional information. So we actually have only 12 equations and 13 unknowns.

It can be proved that this will always happen for any system. That is, at the end we will have a set of $m_r + r + 1$ equations, from which one has to be discarded, and $m_r + r + 1$ unknowns. These equations come from two different approaches: the first $(m_r + 1)$ equations come directly from the first rows in $\mathbf{P}$; the remaining r equations come from the additional restriction of some c_i being zero in eq. (26). To obtain this second subset of equations, matrix $\mathbf{A}$ needs to be diagonalized. In order to obtain r equations in this subset, it is necessary that there always exist r eigenvalues with modulus greater or equal than one, among the eigenvalues of $\mathbf{A}$. In other words, the polynomial (27) has to have r roots in the exterior of the unit disc. It can be shown that this always happens, provided that $\bar{U} < 1$. The proof of this can be found in appendix A.1.

Anyway, we still need an additional equation in order to solve the set. Without this equation, all what we can do is to put any component of $\boldsymbol{\pi}$ as a function of its first component π_0. This way we obtain an family of solutions for $\boldsymbol{\pi}$, as a function of a single parameter. But there exist a single solution, among this family, for which the sum of components in $\boldsymbol{\pi}$ is 1. So, the last equation needed for completely determine $\boldsymbol{\pi}$ is:

$$\sum_{i=0}^{\infty} \pi_i = 1$$

A simple way for finding $\boldsymbol{\pi}$ consist on simply make $\pi_0 = 1$, and solve the system of $(m_r + r)$ equations, which has now $(m_r + r)$ unknowns, obtaining this way the first $(m_r + r)$ components of $\boldsymbol{\pi}$. Once these components are found, we use them in system (28) so we can find the values of the coefficients c_i different from zero (in our example, we will obtain this way the values of c_6 and c_7). Finally, using these values of c_i in eq. (25), along with the eigenvalues λ_i with modulus less than 1, we will obtain the general form of π_n for any $n > m_r$. So we will have found the complete vector $\boldsymbol{\pi}$.

Let us follow these steps in our example

1. make $\pi_0 = 1$ and solve the set of 12 equations made of
   - The 8 equations shown in (21)
   - The 4 equations obtained by enforcing $c_1 = c_2 = c_3 = c_4 = 0$ in (28). Note that some equations in (28) include complex numbers, as for example the

equation related to $c_1 = 0$. Since both the real and the imaginary part of c_1 must be zero, each of these equations will give rise to two equations. But, it can be also seen that, for each equation involving complex numbers, there exist another one with the same numbers but conjugated. This way, in our example, the conjugate equation of $c_1 = 0$ is $c_2 = 0$. The equations derived from the conjugate equation are the same, so at the end the total number of different equations obtained is 4, and not the double at it may seems at first sight.

After solving the set of equations we obtain:

$$\begin{array}{lll} \pi_1 = 0.21508 & \pi_5 = 0.003661 & \pi_9 = 0.000053 \\ \pi_2 = 0.09230 & \pi_6 = 0.001278 & \pi_{10} = 0.000019 \\ \pi_3 = 0.02976 & \pi_7 = 0.000443 & \pi_{11} = 6.4737\,10^{-6} \\ \pi_4 = 0.01065 & \pi_8 = 0.000154 & \pi_{12} = 2.2500\,10^{-6} \end{array}$$

2. Use these results into the system (28) in order to find c_6 y c_7 (all the remaining c_i must be zero, and this can be checked at this point). In our case, we obtain:

$$c_6 = -0.00136122 \qquad c_7 = -1.50568\,10^{-6}$$

3. Use these values of c_6 and c_7, along with the eigenvalues $\lambda_6 = 0.3476$ and $\lambda_7 = -0.1325$ and the eigenvectors $\mathbf{v}_6$ and $\mathbf{v}_7$, into eq. (25), to obtain:

$$\begin{pmatrix} \pi_{n-2} \\ \pi_{n-1} \\ \pi_n \\ \pi_{n+1} \\ \pi_{n+2} \\ \pi_{n+3} \\ \pi_{n+4} \end{pmatrix} = -0.00136 \begin{pmatrix} -0.93766 \\ -0.32590 \\ -0.11327 \\ -0.03937 \\ -0.01368 \\ -0.00476 \\ -0.00165 \end{pmatrix} (0.3476)^{n-8} - 1.5057\,10^{-6} \begin{pmatrix} -0.99118 \\ +0.13132 \\ -0.01740 \\ +0.00231 \\ -0.00031 \\ +0.00004 \\ -0.00000 \end{pmatrix} (-0.1325)^{n-8}$$

This matrix equation is valid for any $n > 8$, and it gives rise to 7 equations which describe the general form of $\boldsymbol{\pi}$. In fact, all the 7 equations describe the same shape, so we can take any of them. For example, the first one:

$$\pi_{n-2} = (-0.0136)(-0.93766)(0.3476)^{n-8} - (1.5057\,10^{-6})(-0.99118)(-0.1325)^{n-8}$$

valid for any $n > 8$. If we make the variable change $n-2$ to n, and operate, we obtain:

$$\pi_n = 0.0127635(0.3476)^{n-6} + 1.49242\,10^{-6}(-0.1325)^{n-6} \quad n > 6 \tag{29}$$

This way we have found an expression which gives all the components of $\boldsymbol{\pi}$, from the sixth one onwards (the first six ones were obtained before from the initial set of equations).

However, we assumed $\pi_0 = 1$ in order to find the remaining components. This assumption is false, because the sum of the obtained $\boldsymbol{\pi}$ is not 1. The last step consist on computing the sum of $\boldsymbol{\pi}$, and dividing all its components by this sum. The new $\boldsymbol{\pi}$ obtained this way would be a probability function, and thus it would be the desired backlog distribution.

The sum of the components of $\boldsymbol{\pi}$ can be easily found, despite the fact that there are an infinite number of terms to be summed. This is because from a given component onwards, all components follow the general form given in eq. (29) which is a sum of exponentials, and thus is summable. So,

$$\begin{aligned}
\sum_{i=0}^{\infty} \pi_i &= \sum_{i=0}^{6} \pi_i + \sum_{i=7}^{\infty} \pi_i \\
&= \sum_{i=0}^{6} \pi_i + \sum_{i=7}^{\infty} 0.0127635(0.3476)^{i-6} + 1.49242\,10^{-6}(-0.1325)^{i-6} \\
&= \sum_{i=0}^{6} \pi_i + 0.0127635 \sum_{i=7}^{\infty} 0.3476^{i-6} + 1.49242\,10^{-6} \sum_{i=7}^{\infty} (-0.1325)^{i-6} \\
&= \sum_{i=0}^{6} \pi_i + 0.0127635 \sum_{j=1}^{\infty} 0.3476^{j} + 1.49242\,10^{-6} \sum_{j=1}^{\infty} (-0.1325)^{j}
\end{aligned}$$

This is the sum of a geometric series, with ratio less than 1, so it converges and the sum can be computed by

$$\sum_{i=n}^{\infty} \rho^i = \frac{\rho^n}{1-\rho} \qquad \rho < 1$$

In addition, the sum of the first terms $\pi_0, \ldots, \pi_6$ is also known, because we have their numerical values. So we can finally compute the sum of $\boldsymbol{\pi}$ as:

$$\sum_{i=0}^{\infty} \pi_i = 1.353413768$$

Therefore, if we divide all the values $\pi_0, \pi_1, \ldots, \pi_{12}$ and equation (29) by the above sum, we will obtain the desired answer. The final result is shown in table 3. Note that some terms ($\pi_7, \pi_8, \ldots, \pi_{12}$), can be obtained by two different means. On one hand they can be obtained from the set of 12 equations, and on the other hand they can be obtained from the closed form of π_i, valid for $i > 6$. However, both methods should provide the same result.

Conclusions We have developed in detail a simple example, but the method can be applied to any system with $\bar{U} < 1$. However, it has to be stressed its high computational complexity. This complexity is due to several factors:

12 first elements					
π_0	0.738872	π_4	0.007869	π_8	0.000114
π_1	0.158917	π_5	0.002705	π_9	0.000040
π_2	0.068203	π_6	0.000944	π_{10}	0.000014
π_3	0.021987	π_7	0.000328	π_{11}	0.000005

Closed form for the remaining elements

$$\pi_i = 0.000943062(0.34766568)^{i-6} + 1.1027\,10^{-6}(-0.1324854)^{i-6} \qquad i > 6$$

Table 3: Normalized steady-state distribution

- It is necessary to find r columns of the Markov matrix. Each column requires to "convolve and shrink" over all jobs in one hyperperiod. The more the number of jobs in one hyperperiod, the more the cost of finding each column in $\mathbf{P}$.
- It is necessary to diagonalize matrix $\mathbf{A}$. This matrix has dimension m_r, which is the length of the r-th column in $\mathbf{P}$. This column is the result of convolving all the execution time PFs of all the jobs in one hyperperiod so, again, the more the number of jobs in one hyperperiod, the bigger matrix $\mathbf{A}$ will be, and the more the cost of its diagonalization.
- Finally, it is necessary to solve a set of $m_r + r$ equation with $m_r + r$ unknowns, and this again has more cost as the number of jobs per hyperperiod increases.

A more detailed study of the computational complexity is required, but the above ideas suggest that the computational complexity can be too high for "real world" systems. This is why we will develop in the next section some approximate methods in order to find the steady-state distribution more quickly.

Anyway, the analytical solution has a theoretical interest, because it shows the general form of the steady-state distribution. It can be seen that, in all cases, it will be made of two parts: a set of initial values (whose shape is arbitrary, depending on the system parameters) and a queue which approaches to zero. The general shape of this queue is a sum of exponentials, being the rates of these exponentials the roots of the polynomial (27) with modulus less than 1. Note that some of these roots can be complex, and in this case the queue will have a "waving" shape, due to the senoid components, which will attenuate to the right.

4. Approximated methods for obtaining the steady-state distribution

4.1. Problems with the analytical solution

The analytical solution presented above has two important problems, which can restrict its practical application. On one hand, the computational cost can be too high, as already said, and this restricts its use to systems in which the number of jobs per hyperperiod is not too high, and the PFs describing the execution time of the jobs have not too many points.

On the other hand, there is a different problem which can affect even to small systems, making impossible their solution using a computer. The described method requires solving a set of equation which will be very ill conditioned in most cases (that is, a little variation in any of its coefficients, can lead to a very different solution).

This is because the set of equations contains both extremely small coefficients (in the order of 10^{-10}, for example), and extremely big coefficients (in the order of 10^{10}, for example). This kind of systems is difficult to solve with a computer in which, typically, the real numbers are stored using a floating-point format. The computer necessarily introduces some small rounding errors, which can be amplified in this case, causing the solution provided by the computer to be very different from the "actual" solution.

The origin of this problem can be found in the way in which the set of equations to solve is built. The reader may recall that the first $(m_r + 1)$ first equations are obtained from matrix $\mathbf{P}$, so the coefficients in these equations are less than 1, and typically very small, because they are the result of several convolutions. The remaining r equations are obtained from a recurrence relationship which started in the r-th column of $\mathbf{P}$. This recurrence relationships uses the coefficient in the r-th column of $\mathbf{P}$, and in order to find the term with higher subindex, it is necessary to divide by the coefficient affecting that term. This coefficient is $b_{m_r}(r)$, cf. eq. (11), and it represents the probability of the total workload in one hyperperiod being equal to the maximum possible. Therefore, it is the product of the probabilities of the worst case for all the jobs in the hyperperiod. It is evident that its value has to be extremely small in all practical cases. When dividing by this value, the obtained coefficients will be very big, cf. eq. (22). These values will appear in the last row of matrix $\mathbf{A}$.

This combination of very small and very big coefficients, makes the set of equations very difficult to solve numerically by computer.

For these reasons, it is necessary to investigate in methods which allow to obtain an approximation of the desired solution, and do not have the mentioned inconvenients. That is, we need methods with fewer computational complexity and fewer sensibility to numerical errors.

In this section two trivial approximations are presented, but this could be an open field for a more exhaustive research.

4.2. Truncation of the Markov matrix method

Bear in mind that the problem to solve is to find $\boldsymbol{\pi}$ in equation $\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}$, and that this is not something immediate, because matrix $\mathbf{P}$ is infinite. An obvious approximation consist on truncating matrix $\mathbf{P}$, leaving a finite matrix $\mathbf{P}'$ of $M \times M$ dimension, by simply suppressing all elements beyond row $M-1$ and column $M-1$.

This way, we can raise a new approximate equation $\pi' = \mathbf{P}'\boldsymbol{\pi}'$. This equation can be solved by finding the eigenvector of $\mathbf{P}'$ which correspond to the eigenvalue 1. This problem can be solved numerically using any of the methods available in the numerical algorithms literature (as for example, those in [Press *et al.*, 1992]).

Well now, since $\mathbf{P}' \neq \mathbf{P}$, it is very possible that $\mathbf{P}'$ does not have any eigenvalue equal to 1. In this case, we will choose the one closer to 1, and $\boldsymbol{\pi}'$ would be the corresponding eigenvector.

It is evident that the solution $\boldsymbol{\pi}'$ is only an approximation of the real solution $\boldsymbol{\pi}$, in first term because $\boldsymbol{\pi}'$ has only M elements, while $\boldsymbol{\pi}$ had an infinite number of elements, and secondly because $\boldsymbol{\pi}'$ do not correspond to the eigenvalue 1, but to an approximation. It is also evident that, the bigger M, the better the approximation.

This approximation reduces the problem complexity, though not much, as nevertheless it is necessary to obtain the matrix $\mathbf{P}'$, which requires the application of the "convolve and shrink" method M times along a whole hyperperiod, and to solve a set of M equations with M unknowns, in order to find π'. Its main advantage over the "analytical" method is that matrix $\mathbf{A}$ is not required, and neither is required its diagonalization. This saves time, and above all, it is much more robust in relation to the possible numerical rounding errors in the coefficients of $\mathbf{P}'$.

However, it raises a serious problem. The approximated vector $\boldsymbol{\pi}'$, found by truncation, does not coincide in any point with the "real" solution $\boldsymbol{\pi}$. That is, when truncating $\mathbf{P}$ and solving the truncated problem, we do not obtain a "truncated version" of $\boldsymbol{\pi}$, but a different solution. The introduced error by truncating $\mathbf{P}$, is shown in all points of $\boldsymbol{\pi}'$. The main problem is that it is not possible to predict in which way this error will influence the subsequent analysis. This means, as a consequence of the truncation, the following analysis will give us deadline misses probabilities different from the real ones, but it is not possible to predict if these approximations will be above or below the real ones.

Therefore, this technique is only useful to obtain an approximated value of the deadline miss probability, that can be used as a quality metric in a soft real-time system. However, it can not be used to obtain an upper bound for this probability, which would be more useful for strict real-time systems.

4.3. Iterative method

A different approach, which does not require even the obtaining of matrix $\mathbf{P}$, would be to simply apply the method "convolve and shrink" along N hyperperiods. At the end of each hyperperiod, a distribution of the backlog is obtained, and is fed as initial backlog to the next hyperperiod. Since we have proved that, whenever $\bar{U} < 1$ the backlog will converge towards $\boldsymbol{\pi}$, it suffices to compare the PF obtained for the j-th hyperperiod

with that of the $(j-1)$-th one. When they are close enough (for example, when their quadratic difference drops below a given ϵ), we will stop iterating. The last obtained PF is an approximation of $\boldsymbol{\pi}$.

This method has two important advantages. First, it does not require to compute the Markov matrix, so its computational cost will be in general much lower than that of the exact solution, and even lower than that of the truncation method. Moreover, it is very robust against roundoff errors.

However, it also has disadvantages. On one hand, the number of required iterations until convergence is not known beforehand. It is to be expected that, as $\bar{U}$ approaches to 1, the number of iterations required to converge will increase. Experimental results have confirmed this. When $\bar{U}$ is high (in the order of 0.995, for example) the required number of iterations can very high, and it could be more effective to calculate the Markov matrix. However, for values of $\bar{U}$ below 0.9 this method converges surprisingly fast.

On the other hand there is an inconvenient similar to the explained for the truncation method. The obtained approximation is not valid for deriving upper bounds on the deadline miss probabilities. In fact, using the iterative approximation, the deadline miss probability obtained from the approximation will be *always less* than the "true" probability. The explanation is as follows. Let us imagine that the method converges after N iterations, and the obtained approximation is $\boldsymbol{\pi}'$. This approximation has a finite number of elements, because it is the result of a finite number of convolutions. Let us say that its number of elements is n. According to this distribution, the probability of the backlog being greater than $(n-1)$, is zero, for any hyperperiod. However, the true solution has an infinite number of terms, so according to it, the probability of the backlog being greater than n is not-null. Therefore, the "true" response times can be higher than the response times obtained from the approximation, because they have worse initial conditions.

The iterative method, thus, is useful to obtain a lower bound on the deadline miss probability. The bound is very tight, almost equal to the "true" deadline miss probability. However, since it is below, from a theoretical point of view it should not be used to take schedulability decisions for strict real-time systems.

As an example, table 4 shows the result of applying the iterative method to the example system given in table 2, whose exact solution was already obtained. Figure 9 shows the evolution of the quadratic error among iterations, in logarithmic scale. It can be seen that the graph is almost linear, which means an exponential decrease of the quadratic error (ergodic convergence).

A. Appendix

A.1. Number of roots with modulus greater than 1 in the characteristic polynomial of matrix A

Using the method described in section 3.3.4, the total number of unknows will always be $m_r + r + 1$. The first $m_r + 1$ rows of $\mathbf{P}$ always provide $m_r + 1$ linearly independent

	$\mathcal{W}_0$	$\mathcal{W}_1$	$\mathcal{W}_2$	$\mathcal{W}_3$	$\mathcal{W}_5$	$\mathcal{W}_{10}$	$\mathcal{W}_{20}$	$\mathcal{W}_\infty$
0	1	0.837500	0.789734	0.768523	0.750897	0.740816	0.738968	0.738872
1	-	0.131250	0.150109	0.155394	0.158160	0.158899	0.158919	0.158917
2	-	0.031250	0.050976	0.059129	0.065050	0.067794	0.068186	0.068203
3	-	-	0.008203	0.013632	0.018639	0.021485	0.021964	0.021987
4	-	-	0.000977	0.002906	0.005524	0.007464	0.007850	0.007869
5	-	-	-	0.000385	0.001372	0.002430	0.002690	0.002705
6	-	-	-	0.000030	0.000299	0.000779	0.000934	0.000944
7	-	-	-	-	0.000053	0.000238	0.000321	0.000328
8	-	-	-	-	0.000007	0.000069	0.000110	0.000114
9	-	-	-	-	0.000000	0.000019	0.000037	0.000040
10	-	-	-	-	0.000000	0.000005	0.000013	0.000014
11	-	-	-	-	-	0.000000	0.000004	0.000005
12	-	-	-	-	-	0.000000	0.000001	0.000001
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4: Evolution of the backlog probability function, using the iterative method

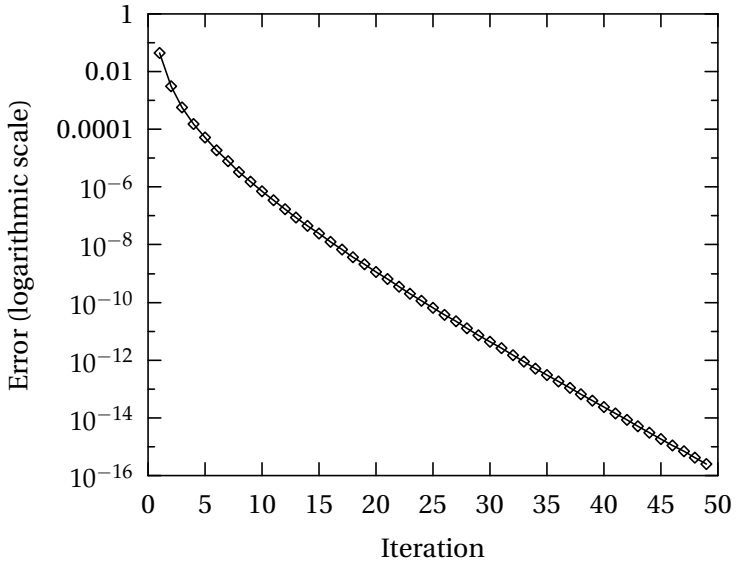


Figure 9: Decrease of the error in each iteration

equations. In addition, the condition of the sum of all elements in $\boldsymbol{\pi}$ being equal to 1 must hold. So they remain only $r-1$ equations to complete the equations set. By proving that the recurrence relationship always provides $r-1$ aditional equations, we would have the set complete.

We will show in this anppendix that, indeed, $r-1$ aditional equations can be obtained from the recurrence relationship, whenever the system has $\bar{U} < 1$. In fact, the number of aditional equations is r, but one among them is redundant, because it corresponds to the eigenvalue 1 which is always present.

Let us remember the origin of these extra r equations. The recurrence relationship can be expressed in matrix form, see eq. (24), where matrix $\mathbf{A}$ has a regular structure consisting on all elements being equal to zero, except its superior diagonal, whose elements are equal to 1, and the last row, whose elements are arbitrary real numbers (see eq. (23), in page 33). To solve this matrix recurrence equation, it is necessary to diagonalize matrix $\mathbf{A}$, and this involves the computation of its eigenvalues. By doing this, we discovered that some of these eigenvalues were greater or equal than 1, and these values must disappear from the final solution, in order to guarantee the summability of $\boldsymbol{\pi}$. In particular, the number of eigenvalues with modulus greater than or equal to 1, was r, and this produced the aforementioned r extra equations.

Then, what we want to prove here is that the characteristic polynomail of $\mathbf{A}$ has always r roots with modulus greater or equal to 1, whenever the stability condition, $\bar{U} < 1$, is met.

The characteristic polyinomial of $\mathbf{A}$ is given by its last row, and, since this row depends only on the coefficients $b_r(\cdot)$ in the r-th column of $\mathbf{P}$, it can be seen that the characteristic polynomial depends only on these coefficients. It is easy to show that the characteristic polynomial has the general form:

$$f(\lambda) = \Big(\sum_{i=0}^{m_r} b_r(i) \lambda^{m_r - i} \Big) - \lambda^{m_r - r} = 0 \tag{30}$$

In addition, proposition 1 (page 23) proved that condition $\bar{U} < 1$ is equivalent to condition $\sum_i i b_r(i) < r$. This is a restriction over the characteristic polynomial coefficients. In the next theorem we will prove that, if this restriction is met, the polynomial has r roots with modulus greater than or equal to 1.

Theorem 5. *A polynomial $f(z)$, of degree m_r, with the form*

$$f(z) = \sum_{i=0}^{m_r} a_i z^{m_r - i} - z^{m_r - r} = 0$$

with coefficients a_i fulfilling the following restrictions:

$$0 \leq a_i < 1; a_{m_r} \neq 0; \qquad \sum_{i=0}^{m_r} a_i = 1; \qquad \sum_{i=0}^{m_r} i a_i < r$$

has exactly r roots with modulus greater than or equal to 1.

Proof. The proof is based in Rouché's theorem, which is a classical result of the complex variable analysis [Rudin, 1966]. According with this theorem, if there exist another function $g(z)$ and a closed path C such that:

$$|f(z)+g(z)| < |g(z)| \quad \forall z \in C \tag{31}$$

then $f(z)$ and $g(z)$ have the same number of roots in the region enclosed by C.

In order to prove the current theorem, we will take $g(z) = z^{m_r - r}$, and C equal to the circle with center at the origin, and radius equal to $1-\epsilon$, being $\epsilon > 0$ very small.

Using function $g(z)$ it is easy to see that $f(z)+g(z) = \sum_{i=0}^{m_r} a_i z^{m-i}$. In order to bound its modulus, we can use the triangular inequality, thanks to the fact that all a_i are positive:

$$|f(z)+g(z)| \leq \sum_{i=0}^{m_r} a_i |z|^{m_r - i}$$

In particular, this inequality is true also for all $z \in C$, so then

$$|f(z)+g(z)| \leq \sum_{i=0}^{m_r} a_i |1-\epsilon|^{m_r - i} \qquad z \in C$$

Second member is equal to function $f(z)+g(z)$ evaluated at real number $(1-\epsilon)$, so

$$|f(z)+g(z)| \leq f(1-\epsilon)+g(1-\epsilon) = (f+g)(1-\epsilon) \qquad z \in C \tag{32}$$

In addition, $f(1-\epsilon)+g(1-\epsilon)$ is less than $g(1-\epsilon)$. This is due to two facts:

1. Functions $f+g$ and g are both equal to 1 when evaluated at $z=1$. That is $(f+g)(1) = g(1) = 1$, as the reader can easily check.
2. The derivative of function $f+g$ at $z=1$ is greater than the derivative of function g. In fact:

$$(f+g)'(1) = \sum_{i=0}^{m_r} (m_r - i)a_i = m_r - \sum_{i=0}^{m_r} i a_i > m_r - r$$
$$g'(1) = m_r - r$$

Since the slope of $f+g(z)$ is greater than the slope of $g(z)$ at $z=1$, the value of $f+g$ has to be less than the value of g at $(1-\epsilon)$, as is shown in fig. 10.

Therefore $(f+g)(1-\epsilon) < g(1-\epsilon)$. Using this in eq. (32), and taking into account that $g(1-\epsilon)$ is positive, because its value is close to 1, we finally obtain:

$$|f(z)+g(z)| < g(1-\epsilon) = |g(1-\epsilon)| \qquad z \in C$$

This means that the conditions of Rouché's theorem hold, so f has the same number of roots than g in the region enclosed by C. All the $m_r - r$ roots of g are zero, so all of them are in the region enclosed by C, so it follows that f has $m_r - r$ roots with modulus less than 1, and therefore r roots with modulus greater than or equal to 1. □

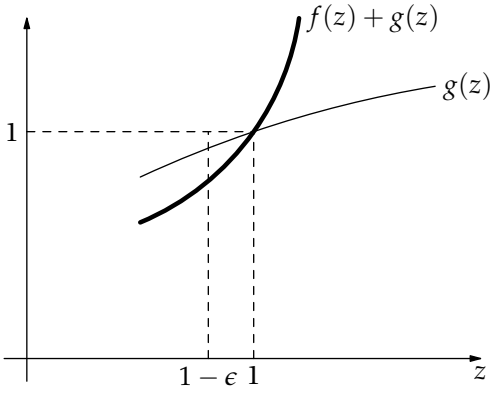


Figure 10: $f + g < g$ at abcise inmediately to the left of 1

References

L. Abeni and G. Buttazzo (1999). QoS Guarantee Using Probabilistic Deadlines. In *Proc. of the Euromicro Conference on Real-Time Systems*. York, UK.

A. K. Atlas and A. Bestavros (1998). Statistical Rate Monotonic Scheduling. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, pp. 123–132.

J. L. Díaz, J. M. López and D. F. García (2002). Probabilistic Analysis of the Response Time in a Real-Time System. In *Proc. of the 1st CARTS Workshop on Advanced Real-Time Technologies*. Aranjuez, Spain. Also available as Technical Report at http://www.atc.uniovi.es/research/PART01.pdf.

M. K. Gardner (1999). *Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems*. Ph.D. thesis, University of Illinois, Urbana-Champaign.

J. P. Lehoczky (1990). Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proc. of the 11th IEEE Real-Time Systems Symposium*, pp. 201–209.

C. L. Liu and J. Layland (1973). Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, **20**(1):pp. 46–71.

S. P. Meyn and R. Tweedie (1993). *Markov Chains and Stochastic Stability*. Control and Communication Engineering Series. Springer Verlag, London.

W. H. Press, W. T. Vettering, S. A. Teukolsky and B. P. Flannery (1992). *Numerical Recipes in C*. Cambridge University Press, 2nd ed.

W. Rudin (1966). *Real and Complex Analysis*. McGraw-Hill, New York, NY.

T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu and J.-S. Liu (1995). Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times. In *Proc. of the Real-Time Technology and Applications Symposium*, pp. 164–173. Chicago, Illinois.

K. Tindell, A. Burns and A. J. Wellings (1994). An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Real-Time Systems*, **6**:pp. 133–151.

R. Tweedie (2000). Markov Chains: Structure and Applications. In D. Shanbhag and C. Rao, eds., *Handbook of Statistics*, vol. 19, pp. 817–851. Elsevier Amsterdam.