Noname manuscript No.
(will be inserted by the editor)

On the nature and impact of self-similarity in real-time systems

Enrique Hernández-Orallo and
Joan Vila-Carbó

the date of receipt and acceptance should be inserted later

Abstract In real-time systems with highly variable task execution times simplistic task models are insufficient to accurately model and to analyze the system. Variability can be tackled using distributions rather than a single value, but the proper characterization depends on the degree of variability. Self-similarity is one of the deepest kinds of variability. It characterizes the fact that a workload is not only highly variable, but it is also bursty on many time-scales. This paper identifies in which situations this source of indeterminism can appear in a real-time system: the combination of variability in task *inter-arrival times* and *execution times.* Although self-similarity is not a claim for all systems with variable execution times, it is not unusual in some applications with real-time requirements, like video processing, networking and gaming.

The paper shows how to properly model and to analyze self-similar task sets and how improper modeling can mask deadline misses. The paper derives an analytical expression for the dependence of the deadline miss ratio on the degree of self-similarity and proofs its negative impact on real-time systems performance through system's modeling and simulation. This study about the nature and impact of self-similarity on soft real-time systems can help to reduce its effects, to choose the proper scheduling policies, and to avoid its causes at system design time.

1 Introduction

The simplest way of characterizing the execution time of a real-time task is by describing it with its WCET (Worst-Case Execution Time). The analysis of hard real-time systems is mostly based on this parameter. However, in

Departamento de Informática de Sistemas y Computadores.
Universidad Politécnica de Valencia. Valencia, Spain.
E-mail: ehernandez@disca.upv.es · E-mail: jvila@disca.upv.es

multimedia and network based real-time systems, workloads are highly variable and the WCET approach faces important drawbacks: on one hand, the problem of accurately determining the WCET is difficult and, on the other, this WCET usually leads to poor resource utilization.

Typical multimedia applications are similar to hard real-time systems in many ways: they periodically process a set of input data (like video frames) and they must complete this processing before a deadline. The main differences with traditional systems is in a major degree of non-determinism: execution is much more variable. This, in turn, motivates different system requirements: CPU bandwidth requirements become more important while deadline requirements can be relaxed to a certain point.

Variability in multimedia tasks mainly affects two parameters: the task *execution time* and the task *inter-arrival time*. Although these parameters usually move around a steady mean value, they experience a high dispersion in their values. That introduces considerable trouble in task scheduling and analysis.

The variability of multimedia *execution times* is due to several factors. One is the *nature of the workload*: there are multiple video frame types (I, P and B in MPEG) exploiting different types of redundancy, and unexpected sequences of these frames. Another factor is the *nature of the algorithms*: frame and image processing algorithms are complex; the percentage of ALU (integer and floating point), branch,and memory instructions in each run can vary significantly, leading to different execution traces. Finally, a third factor comprises *architectural issues*: variations in memory access times due to cache accesses and bus arbitration can also affect seriously the execution times. The experiments of [21] evaluate these factors in MPEG decoding. They show that, despite the differences in frame types, execution time is rather predictable: 90% of the frames has execution times within 10% of the immediately preceding frame. However a 10% of unpredictable cases is still high. Contrary to the common perception, experiments also show that aggressive architectural features and caches have a limited effect on execution time variability, at least on these applications. The variability is shown to be mostly caused by the application algorithm and the amount of media input.

According to previous studies, the variability of task *inter-arrival times* is one of the main reasons for variability. In multimedia systems tasks are triggered by data arriving periodically from communication channels or events produced by sensors. The unpredictability of communication systems (such as wireless networks) and external events makes task activation times to have a high jitter or variable period.

Because of the fatal combination of the variability of task *inter-arrival times* and *execution times*, the processing time of tasks over a sampling period may vary across non-trivial ranges, even when these distributions spread around a well known mean value. However, this high variability does not always implies self-similarity.

Self-similarity [25, 38] characterizes the fact that a workload is not only highly variable, but it is also bursty on many time-scales or, in other words the bursty behavior may itself be bursty. The degree of self-similarity is measured by the Hurst parameter.

Closely related to self-similarity are heavy-tailed distributions. In the last decade it was stated that network traffic shows two properties [22, 29, 37]: i) *self-similarity*: counts of packet arrivals in equally-spaced intervals of time are long-range, time-dependent and have a large coefficient of deviation; and ii) *heavy-tailed*: packet inter-arrival times have a marginal distribution that has a longer tail than the exponential distribution. The Pareto and Weibull distributions are often used to characterize heavy-tailed distributions. Some studies show that multimedia streams exhibit self-similar properties with Hurst values between 0.75 and 0.95 [14, 31]. The frame sizes of VBR video (MPEG) are best matched with a heavy-tailed Pareto distribution [14].

This paper analyzes in which situations the workload of a real-time system can be regarded as self-similar, its characterization, its modeling and its implications on system performance and on the schedulability analysis.

The impact of self-similarity and heavy-tailed distributions on non real-time scheduling disciplines has been studied in related fields, like Unix scheduling, network traffic, and web servers.

In [4, 17, 23] it is shown that CPU requirements of Unix processes are self-similar and have heavy-tailed distributions. This finding was very important from the scheduling point of view because it motivated to reevaluate Unix scheduling disciplines in terms of their behavior with these heavy-tailed workloads [16].

The strong impact of self-similarity on packet scheduling has been clearly shown [8, 11, 12, 20, 30]. The main effect is an increase in the packet loss rate and in the packet delays due to greater buffer delays.

Finally, studies of the response times (sojourn times) for heavy-tailed and self-similar workloads on web server applications show that FCFS and other non-preemptive scheduling disciplines, perform very poorly [4] while, in contrast, Priority Sharing (PS) and Shortest Remaining Processing Time (SRPT) are optimal [7, 15].

The influence of variability on real-time scheduling has been recently addressed. Several statistical characterizations and analysis methods have been proposed [2, 10, 13, 32, 35, 36] but they have not taken into account the self-similar nature of media workloads. Nevertheless, it is important to note the strong impact of variability on soft real-time scheduling in the experiments of [32].

This paper models tasks variability using an "on-off" model [5] where the "on" state represents time intervals where the task is active and the "off" state represents the intervals of task inactivity. The duration of the "on" or "off" states are assumed to be highly variable. Such a set of tasks behaves statistically as a fractional Brownian motion (fBm). The analytical analysis of schedulability is based on applying the concept of *synthetic utilization* [1] to the previous "on-off" model. Results show the dependency of schedulability on the Hurst parameter. Furthermore, in a set of periodic and aperiodic tasks, the aperiodic workload can seriously jeopardize the schedulability of periodic tasks. Analytical results are further checked and confirmed via simulations. The study allows us to conclude that a bandwidth isolation or reservation mechanism (such as the Constant Bandwidth Server (CBS) [3]) is recommended for environments with a mix of hard-deadline periodic tasks and soft-deadline sporadic tasks. As proven in the experiments, the CBS scheduler assures the hard-deadline of the periodic tasks independently of the level of self-similarity of the sporadic tasks.

This paper extends the work in [19] in several ways. First, it introduces some motivating examples taken from real applications where the self-similarity property is detected and its impact shown. It also proposes a simplified system model and a workload modeling method. Finally, it uses a new schedulability analysis based on the concept of deadline tightness. Experiments have also been extended and updated.

The remainder of the paper is organized as follows: first, section 2 presents two motivating examples that shows the impact of self-similarity in scheduling efficiency; section 3 introduces the model and assumptions of real-time system used in this paper; section 4 presents the basic "on-off" model and its application to modeling real-time tasks; section 5 contains the schedulability analysis; section 6 evaluates the influence of self-similarity on the Earliest Deadline First (EDF) scheduler, and section 7 presents some concluding remarks. Finally, appendix A reviews the concepts and definitions of self-similarity and heavy-tailed distributions and it is intended for those readers not familiar with the self-similarity background.

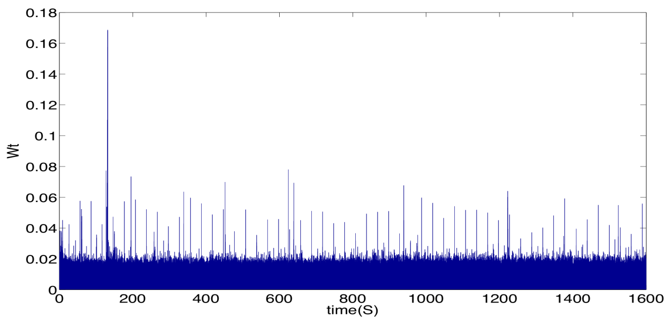
2 Motivation

This section presents some examples that prove the existence of highly variable and self-similar real-time computations in multimedia and network applications and motivate the proposed task characterization and analysis of the following sections. The impact of such workloads on real-time systems scheduling is also outlined through a simple experiment.

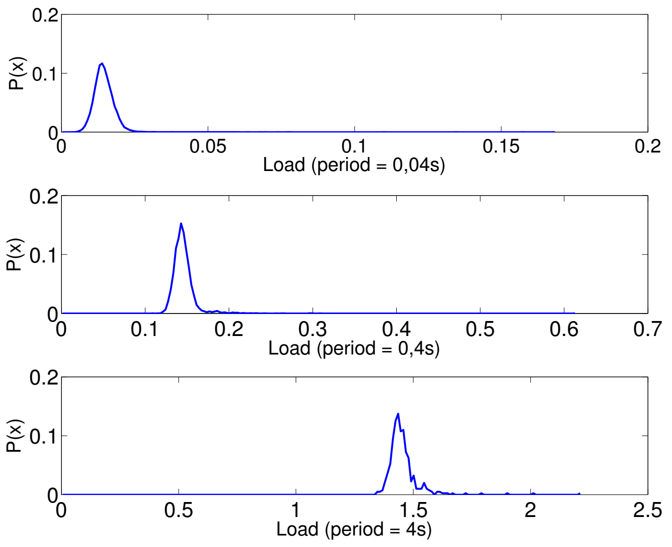
2.1 Analysis of a self-similar multimedia workload

Figure 1a shows a workload trace of a video processing system in a Unix system that comprises 8 tasks: 6 periodic tasks and 2 sporadic tasks. The trace W_t was captured using the *DTrace* utility and it represents the time evolution of the computation intensity, that is, the amount of CPU load over a sampling period T of 0.04s. This workload produces the probability distribution shown in Figure 1b (top). This distribution is highly variable and can be modeled as a heavy-tailed distribution.

An interesting question about the variability level concerns its dependence on the sampling period. Intuitively, it could be expected that increasing the sampling period would smooth the variability in the computation intensity.



(a)



(b)

$T(sec)$	$E(W_t)$	$Var(W_t)$	$E(W_t)/T$	$Var(W_t)/T$
0.04	0.014558	0.000015	0.364	0.000382
0.4	0.145583	0.000234	0.364	0.000584
4	1.455833	0.005427	0.364	0.001357

(c)

Fig. 1: Workload trace of a Unix system processing video image. (a) Workload versus time using a period of 0.04s and duration of 1,600s. (b) Distribution using three periods: 0.04s, 0.4 and 4s. (c) Mean value and variance for different sampling periods.

This would allow to apply more conventional analysis methods and obtain more deterministic long-term results. However, we surprisingly observed that this does not hold for these workloads. Figure 1b also shows the workload distribution for two greater sampling periods: 0.4, and $4s$ (that is, an increase of 10 and 100 over the original period). It can be observed that the variability remains and the shape of the distributions are very similar. This is confirmed through table 1c that shows the average computation time, its variance, and their normalized values (that is, divided by the sampling period). It can be observed that, although the normalized mean value is the same, the normalized variance increases with the sampling period more than expected. This shows the fact that the workload is bursty on many time scales or, in other words, self-similar.

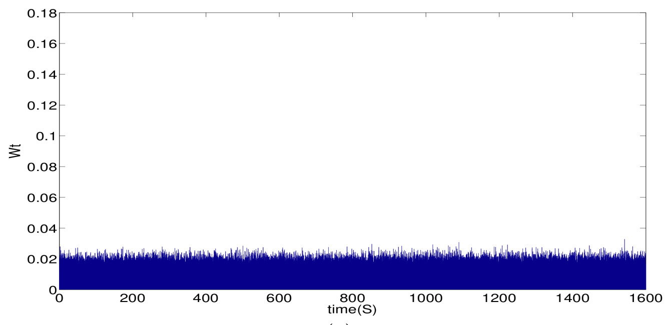
It is important to note that a self-similar workload is not a *white noise*. Figure 2a compares the previous results to a random traffic workload. It is also a bursty workload, but if we examine the distributions in Figure 2b, the shapes of the distributions get narrow and the variability decreases with the sampling period. This allows to state that a white noise is only bursty in small time scales. In this case, the normalized variance remains constant. Moreover, if the workload is shuffled (that is, workload positions are exchanged randomly), results are similar. That is, the shuffled workload is bursty only in low scales and it is an evidence that a variable workload is not equivalent to a self-similar workload.

The self-similarity nature of a workload can be shown and measured using a *variance-time plot*. This plot represents the normalized variance versus the increment of the period in a log scale. Figure 3 shows the points for the variance-time plot of the video and white noise workloads. It also represents the least-square lines for these points. The slope of this line is known as β. The Hurst parameter that characterizes self-similar workloads, can be calculated as $H = 1 - \frac{|\beta|}{2}$. Self-similarity holds when $0.5 < H < 1$. In the case of the white noise (and also the shuffled multimedia workload) the slope is -1, so $H = 0.5$ and the workload is not self-similar. But in the case of the video workload, that has a flatter slope, $H = 0.73$, evidencing its self-similar nature.

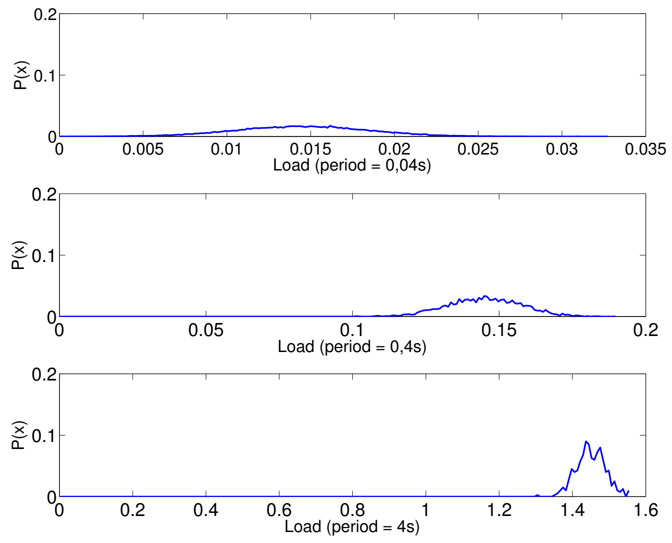
2.2 Impact of self-similarity on schedulability

The strong impact of self similarity on schedulability can be also shown through a simple example.

Consider a real-time system with four deterministic tasks with a period (equal to deadline) of 1 and a utilization of 0.25. This workload will be referred as NO-VAR and it has a full processor utilization of 1. The systems is schedulable because this full utilization is achievable with equal task periods. Now we introduce a variability in the system by making the execution time and inter-arrival time (period) variable. Four set of tasks, or workloads, will be formed using the same distribution of execution times and four different degrees of variability (variance) in the inter-arrival times. These workloads are



(a)



(b)

Fig. 2: Synthetic workload trace (white noise). a) Workload versus time using a period of 0.04s and duration of 1,600s. b) Distribution using three periods: 0.04s, 0.4 and 4s.

shown in table 1 and are referred as LOW-VAR, HIGH-VAR, SELF-SIM1 and SELF-SIM2.

The distribution of the execution time for all workloads is a normal distribution with a mean of 0.25s and a variance of 0.05. The inter-arrival times distributions has also a common mean value of 1 but each workload has a different variability of the inter-arrival time: normal distribution with variance 0.05 for the LOW-VAR workload, normal distribution with variance 0.5 for the HIGH-VAR workload, and finally, two self-similar workloads SELF-SIM1 and SELF-SIM2 using Pareto distributions with a Hurst parameter of 0.75

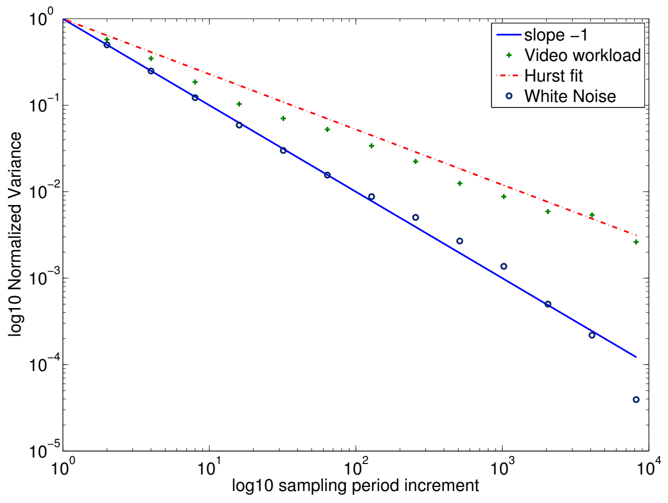


Fig. 3: Variance time plot of the video workload and white noise workload The Hurst parameter for the video workload is 0.72.

Workload	Execution time		Inter-arrival time		Deadline miss ratio %	
	$Mean$	Var	$Mean$	Var	$D = 1$	$D = 2$
NO-VAR	0.25	0	1	0	0	0
LOW-VAR	0.25	0.05	1	0.05	4.975	2.478
HIGH-VAR	0.25	0.05	1	0.5	6.175	2.761
SELF-SIM1	0.25	0.05	1	H=0.75	7.823	6.062
SELF-SIM2	0.25	0.05	1	H=0.9	12.387	9.246

Table 1: Motivating example: deadline miss ratio depending on the variability of the task set

and 0.9 respectively. How these self-similar tasks are modeled, generated and evaluated is further detailed in this paper.

The workloads were scheduled using an EDF scheduler using two deadlines: a tight deadline $D = 1$ and a loose deadline $D = 2$. The deadline miss ratio for each workload is shown in the table. Results are very illustrative: increasing the variance has a great impact on the efficiency of the EDF scheduler, as shown for the HIGH-VAR workload. This is a very well known fact in real-time systems [32]. However, the results for the self-similar workloads (SELF-SIM1 and SELF-SIM2) shows a more important increase in the deadline miss ratio over the HIGH-VAR workload, and this effect is more important for loose deadlines.

This simple example shows the impact of self-similarity workloads in real-time system scheduling and motivates the need for the correct modeling and analysis of these tasks in soft real-time systems. How to properly model such

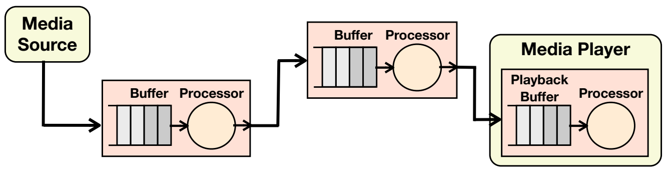


Fig. 4: Multimedia pipeline processing

a self-similar workload, the effect of the sampling period and the Hurst parameter on schedulability is further analyzed in the following sections.

3 System Model

The model of real-time systems assumed in this work was originally devised in the scope of multimedia transmission and processing, although it can be generalized to task systems with highly variable processing times.

Computations in a media processing system are usually structured in a pipeline fashion as shown in Figure 4. Data are produced continuously by media sources (encoders) at highly variable rates. Each task of the pipeline reads data, processes it and sends it to the next task. Tasks are interconnected through buffers. The last task of a pipeline is usually a media player, and its input buffer is usually known as the *playback buffer*. Arrival patterns determine whether tasks will be treated as periodic or sporadic. *Periodic tasks*, such as video rendering, execute their invocations within strict regular time intervals, but their computation time is highly variable. Periods are usually related to application parameters, like video frame frequency or audio sampling period. *Sporadic tasks*, such as video encoding and video processing, are usually data-driven. These tasks are defined to be activated at arbitrary points in time with a defined minimum inter-arrival times between two consecutive invocations. In our system, *sporadic tasks* model tasks triggered by data arriving with a variable period. This period is highly variable around a mean value. This is also often referred as *jitter*. Buffering plays a key role in these systems: it allows to pipeline periodic and sporadic tasks asynchronously.

The task model proposed for the above system is really a statistical extension of the classical real-time task model. The model assumes a mix of periodic (P^i) and sporadic tasks (S^i) with the following attributes (see Figure 5):

- *Computation time* (C^i): it is defined as the intensity or CPU load of a task. It is specified as an statistical distribution. It differs from the classical real-time model where C^i is a deterministic value that corresponds to the worst-case execution time (WCET).
- *Inter-arrival time* (I^i): it is defined as the time between the arrival of two consecutive task invocations (jobs). For periodic tasks it is a deterministic value which specifies the *period*. For sporadic tasks it is an statistical distribution.

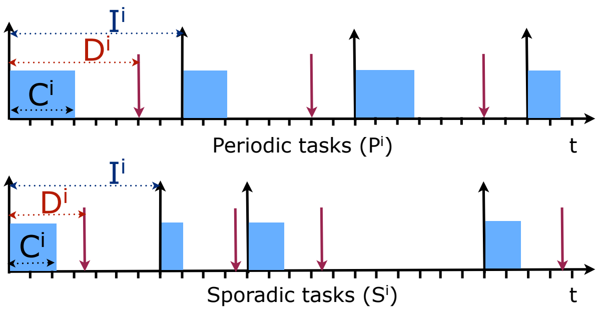


Fig. 5: Task model for periodic and sporadic tasks. Task arrivals are represented as up-arrows. They are equally spaced for periodic tasks and variably spaced for sporadic tasks. Computation times are represented as variable-length boxes. Deadlines are represented as down-arrows and they are always constant.

– *Deadline* (D^i): it is a deterministic value that expresses the maximum allowed time between task arrival and time completion.

Statistical characterizations were used by the authors of this paper in previous studies [18, 36] and also by other authors [2, 13, 35]. The novelty of this paper is in properly modeling highly variable tasks using heavy-tailed distributions and extending task characterization with the inclusion of the Hurst parameter.

4 A Self-similar Workload Model

This section presents a self-similar workload model for real-time tasks. The goal of this model is to enable a performance analysis of a real-time system with highly variable tasks. The analysis method can be either analytical, as the one presented in section 5, or simulation based, as the one presented in section 6.

Real-time computations with variable execution times can be modeled as an "on-off" process. This model is based on a proposal of [39] to describe the random nature of the network traffic, originally due to Mandelbrot [5]. Using this model a real-time task could be modeled as a task that goes through two states that strictly alternate: "on" and "off" . The arrival of a task instance triggers the start of the "on" state whose duration is given by the computation time. The "off" state represents the time between task termination and the arrival of the next task instance (or job). Figure 6 shows a system example with 3 tasks. The first job of task one arrives at time 1 and has an expected execution of 3.8 time units. The next job of this task is at time 9.1 and has an execution time of 2. The first job of task two arrives at time 1.9 and has an expected execution time of 2.5. Note that the model parameters reflect the

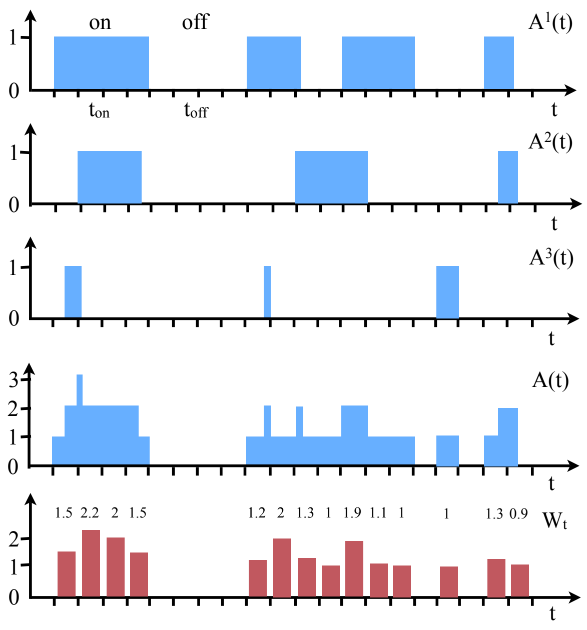


Fig. 6: Example of superposition of 3 on-off streams. Note that $A(t)$ is continuous-time and that W_t is discrete-time so it is represented using bars.

arrival and the execution times of the jobs and not when the jobs are executed (this will depend on the scheduler).

The superimposition of many "on-off" tasks, where the "on" and "off" periods strictly alternate, can be proved to be a self-similar workload and statistically behaves as a fractional Brownian motion under certain requirements on the distributions of the "on" and "off" states.

4.1 Formal "on-off" model

Let $\{A(t) : t \geqslant 0\}$ ($\{A(t)\}$ for short) be a continuous-time stationary process that is the superimposition or aggregation of N-independent, "on-off" tasks in which:

$$A^i(t) = \begin{cases} 1, \text{ for interval "on"} \\ 0, \text{ for interval 'off"} \end{cases} \quad i = 1, 2, 3, \ldots, N \tag{1}$$

so:

$$A(t) = \sum_{i=1}^{N} A^i(t) \tag{2}$$

Consider the cumulative process $\{A^*(t) : t \geq 0\}$ defined as:

$$A^*(t) = \int_0^t A(u)du \tag{3}$$

The increment process $\{W_t : t = 1, 2, \ldots\}$ ($\{W_t\}$ for short) is defined as:

$$W_t = A^*(tT) - A^*((t-1)T) = \int_{(t-1)T}^{tT} A(u)du \qquad (4)$$

where T is the sampling period. Then, W_t is a discrete-time random variable representing the amount of workload entering a system during the t^{th} sampling interval (that is, the workload trace)[1].

The mean intensity (or utilization) of the workload W_t is defined as $E[W_t]/T$. A sample realization of processes $\{A(t)\}$ and $\{W_t\}$ are shown in Figure 6.

The behavior of process $\{W_t\}$ depends on the distributions of the "on" and "off" periods (t_{on} and t_{off} for short). Define P_{on} as the probability that a task is in the "on" state:

$$P_{on} = \frac{\bar{t}_{on}}{\bar{t}_{on} + \bar{t}_{off}} \qquad (5)$$

where $\bar{t}_{on}$ and $\bar{t}_{off}$ are the mean of the "on" and "off" periods respectively. Self-similarity requires that either the distributions of t_{on} or t_{off} are heavy-tailed, or both [39]. This is formally stated in the following theorem:

Theorem 1 *[28]: If t_{on} is Pareto distributed with index α and for large N and m (aggregation factor), the cumulative process $\{A^*(mt)\}$ behaves statistically like:*

$$P_{on}Nmt + CN^{1/2}m^H Z_H(t) \quad H = \frac{3-\alpha}{2} \qquad (6)$$

where $Z_H(t)$ is a fractional Brownian motion (fBM) with parameter H, and $C > 0$ is a constant depending on the distribution of t_{on} and t_{off}. Thus $\{A^*(mt)\}$ asymptotically behaves as a fractional Brownian motion (that is, equation 23) fluctuating with mean $\mu = P_{on}Nmt$ and variance $\sigma = CN^{1/2}m^H$, with Hurst parameter H.

If $\{A^*(mt)\}$ is asymptotically self-similar, then its increment process $\{W_t\}$ is also asymptotically self-similar [28]. Thus, $\{W_t\}$ with t_{on} described by a Pareto distribution and with t_{off} independently distributed is *asymptotically second-order self-similar* with a Hurst parameter given by $H = (3-\alpha)/2$. The same holds if the "off" states are heavy-tailed, or both.

Theorem 1 holds for homogenous sources, that is, all the streams have the same α and d parameters. There is a generalization of this theorem for heterogeneous sources:

Theorem 2 *[34]: Assume M types of sources. For $i = 1, 2, \ldots, M$ let α^i and d^i be the characteristics of source i. This theorem states that $\{A^*(mt)\}$ asymptotically behaves as fractional Brownian with a Hurst parameter given by:*

$$H = \frac{3-\alpha^{min}}{2} \qquad (7)$$

[1] Note that the aggregate process of W_t can be obtained as $W_t^{(m)} = \int_{(t-1)Tm}^{tTm} A(u)du$

This means that the smallest α ultimately dominates the Hurst parameter. Note that this theorem allows the distribution to be not self-similar for some source type.

Using the previous theorems, we can define our task model as an "on-off" model using the Pareto distribution for the "on" state. This implies that this model is self-similar with Hurst parameter H given by equations 6 or 7. The average time interval for an "on" state for $1 < \alpha < 2$ is (the Pareto mean):

$$\bar{t}_{on} = \frac{\alpha d}{\alpha - 1} \tag{8}$$

For the "off" state, we can use any distribution or even a constant value. The only requirement is that it must be statistically independent from the "on" distribution. Using equation 5, we can obtain the probability that a task is in the "on" state so the expected intensity of a given on-off stream is given by:

$$E[A^i(t)] = P_{on}, \quad i = 1, 2, \ldots, N \tag{9}$$

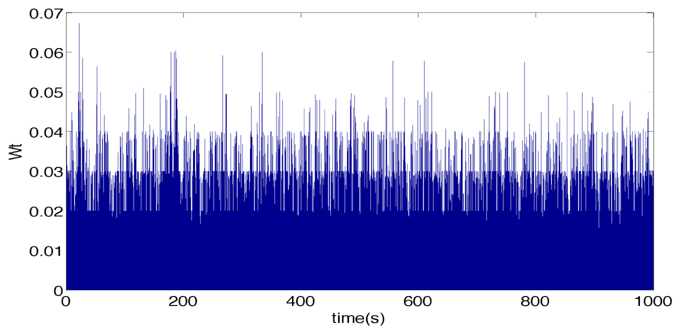
and the overall intensity (or system load L) is:

$$L = E\left[\sum_{i=1}^{N} A^i(t)\right] = N \cdot P_{on} \tag{10}$$

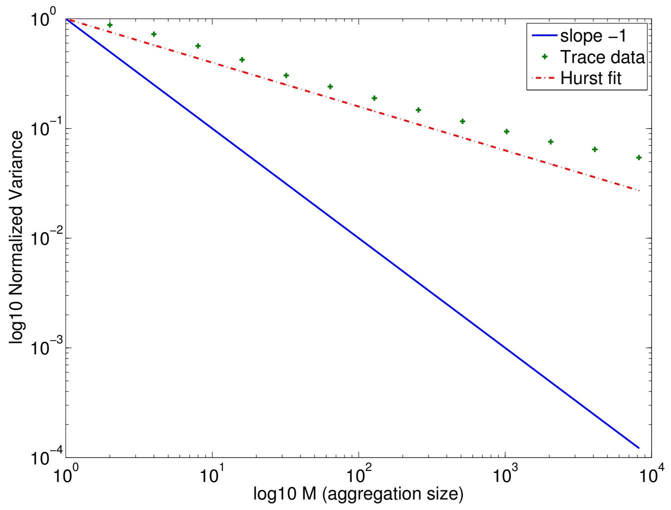
Figure 7a shows a synthetic workload trace of 10 tasks ($N = 10$), with a Hurst parameter of $H = 0.8$. This trace was generated for a time of 1000 seconds with a sampling period $T = 0.01s$. The "on" period was Pareto distributed with $d = 0.01$ and $\alpha = 1.4$ $(3 - 2 \cdot 0.8)$. The average time for t_{on} can be calculated using equation 8 and is 0.035. In order to have an intensity for each stream of about 0.1, t_{off} was obtained using equation 5. Thus, t_{off} is constant with value 0.315. The mean intensity or load of this workload is 1. In order to test the self-similarity of this workload, we used the variance time log plot. Figure 7b shows the variance time plot of the workload using a base period of 0.01s. The least-square line with slope $\beta = -1$ determines whether a traffic is self-similar or not. If the variance plot is above this line, the traffic is self-similar; otherwise, the traffic is not self-similar. The fitted line has slope $\beta = -0.4$ that corresponds to a Hurst parameter of $H = 1 - |\frac{\beta}{2}| = 0.8$, which is the expected value.

4.2 Modeling real-time tasks

This section describes how to generate an "on-off" workload whose first-order (mean) and second-order (self-similarity) statistics properties are equivalent to a given set of periodic and sporadic tasks. The goal is setting the parameters of the "on-off" model of the periodic and sporadic tasks described in section 3. The process is depicted in figure 8 and comprises the following steps:



(a)



(b)

Fig. 7: a) Workload trace for $H = 0.8$, $N = 10$ and $T = 0.01s$. b) Variance time plot of the workload (base period for m=1 is 0.01s).

1. Obtaining a trace of all task invocations of the evaluated system that provides the following parameters for each task invocation: task id (i), arrival time (t), and computation time (C_t). This can be done using a high resolution monitor.
2. Processing the previous trace to filter each task separately, providing the following parameters: inter-arrival time (I_t), and computation time (C_t)
3. Fitting the inter-arrival time and computation of tasks to an statistical distribution.

One of the most important aspects is to fit each task to a distribution. This is a manual procedure where statistical fitting tools may be needed. It

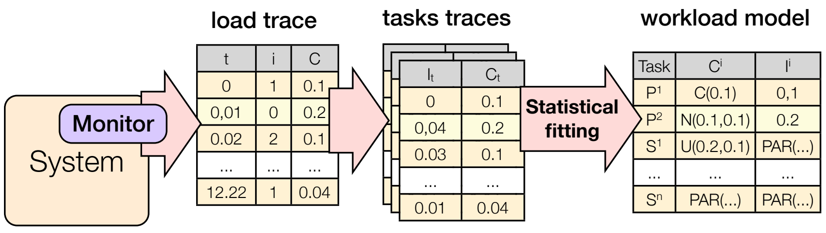


Fig. 8: Process for modeling real-time tasks

Type	Description
C(c)	Constant distribution with value c
U(a, b)	Uniform over interval (a, b)
min + EXP(λ)	Exponential with a min value
N(μ, σ)	Normal
PAR(d, α)	Pareto

Table 2: Distribution classes for the workload model

starts by checking whether the inter-arrival time is constant, so periodic and sporadic tasks can be identified. Then, computation times (and inter-arrival times) must be adjusted to some typical distribution, as the ones shown in Table 2 (some other distributions can be used as well). Some periodic tasks, like video playing, have heavy-tailed distributions for computation times, while others may have variable computation times, but not necessarily heavy-tailed. Sporadic tasks usually have an inter-arrival distribution which is heavy-tailed, specially those processing inputs delivered through the network. On the other hand, data processing can also be heavy-tailed and independent from the inter-arrival distribution. Using theorem 2, this mix of periodic and sporadic tasks generates a self-similar workload. Using this model we can generate a workload trace that can be used to analyze a real-time system. Tasks are modeled using a t_{on} which corresponds to the computation time (C^i) and $t_{off} = E[I^i] - t_{on}$, where $E[I^i]$ is the mean inter-arrival time. Finally, the deadlines (D^i) are obtained from the requirements of each task.

As an example, let's make a synthetic workload model for the real system described in section 2.1. The system has 6 periodic tasks and 2 sporadic tasks. The most important task is P^1, which models a task that processes a video stream. Sporadic tasks (S^1 and S^2) are associated with network streams and, thus, their inter-arrival distributions are heavy-tailed. We generated a workload trace for a time of 1600s with a period of 0.04s. The result was similar to the traffic trace of Figure 1a. The mean utilization of this workload trace is 0.45 and is shown in Figure 9a. Figure 9b shows the variance time plot of the workload using a base period of 0.01s. It can be seen that the calculated variance values are almost linearly distributed and can be adjusted to a line. The slope of the line corresponds to a Hurst parameter of 0.72, which allows us to state that the workload generated is self-similar.

Task	C^i	D^i	I^i	L^i
P^1	$\mathrm{PAR}(0.002, 1.4)$ $(H = 0.8)$	1	0.04	0.175
P^2	$\mathrm{C}(0.1)$	1	25	0.004
P^3	$\mathrm{U}(0.02, 0.03)$	0.05	0.05	0.013
P^4	$\mathrm{N}(0.01, 0.001)$	0.1	0.1	0.1
P^5	$0.001 + \mathrm{EXP}(0.001)$	0.1	0.1	0.01
P^6	$\mathrm{U}(0.001, 0.002)$	0.01	0.1	0.015
S^1	$\mathrm{PAR}(0.01, 1.2)$ $(H = 0.9)$	0.5	$\mathrm{PAR}(0.2, 1.2)$	0.047
S^2	$\mathrm{U}(0.01, 0.05)$	0.1	$\mathrm{PAR}(0.1, 1.2)$	0.047

Table 3: Sample real-time workload. The first columnis the task id: P^i for periodic and S^i for sporadic. The last column L^i shows the intensity or load of each task. All time values are in seconds.

5 Schedulability Analysis

This section shows how the schedulability probability of a self-similar workload depends on the Hurst parameter and how not accounting for the self-similar property of the workload can lead to optimistic performance predictions.

There is a clear evidence of the importance of the Hurst parameter in network traffic models based on traditional queueing models [8, 11, 20, 26]. This paper shows the importance in real-time analysis too.

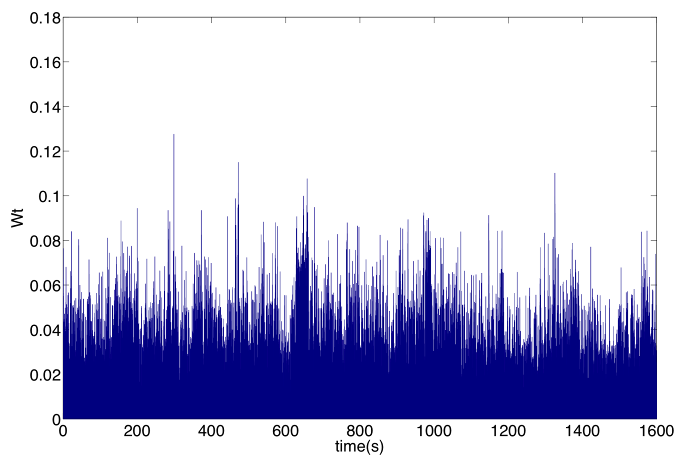
The analysis described in this section is based on a recent schedulability test by [1] that basically requires the *synthetic utilization*, denoted as $U(t)$, to be under a certain utilization bound B. Using this expression, we derive a new schedulability test, that is based on the *pending computation* at time t ($C(t)$). For the previously proposed "on-off" task model, this pending computation can be obtained from the expression of the accumulated workload $A^*(t)$ given by equation 6. The final expression is the probability of not passing this schedulability test that is $P(C(t)) > x)$. The calculus of this expression performed below shows that schedulability clearly depends on the Hurst factor.

A straight approach for analyzing the schedulability of a real-time system is via the utilization bounds. The seminal work [24] established a bound for aperiodic tasks when deadline equals to their period. In a recent work [1], a bound for aperiodic tasks was derived using a utilization-like metric called *synthetic utilization.* The synthetic utilization is defined as the utilization contributed by the set of current task invocations at time t, given by the expression:

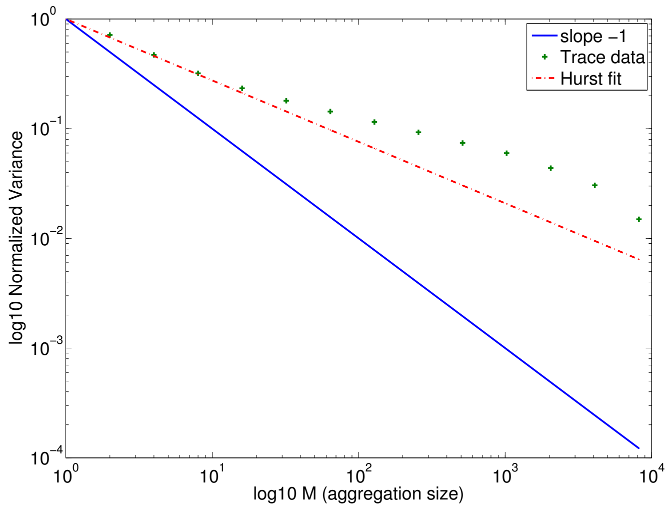
$$U(t) = \sum_{\tau^i \in S(t)} \frac{C^i}{D^i} \tag{11}$$

where $S(t)$ is defined as the set of all task invocations τ^i (jobs) that have arrived after the last CPU idle time s (known as CPU gap) but whose deadlines have not expired. According to this test, a system is schedulable if:

$$U(t) < B = \begin{cases} \frac{1}{2} + \frac{1}{2n} & n < 3 \\ \frac{1}{1+\sqrt{\frac{1}{2}(1-\frac{1}{n-1})}} & n \geq 3 \end{cases} \tag{12}$$



(a)



(b)

Fig. 9: a) Workload trace generated using the workload described in Table 3 b) Variance time plot of the workload (Base period for m=1 is 0.01s). The Hurst parameter is 0.72.

where n is the number of current tasks (that is, the size of $S(t)$). When n increases the bound approaches $B = \frac{1}{1+\sqrt{1/2}}$. This bound is for the optimal time-independent scheduling policy Deadline Monotonic (DM). For dynamic priority scheduling policies, such as Earliest Deadline First (EDF), this bound is 1.

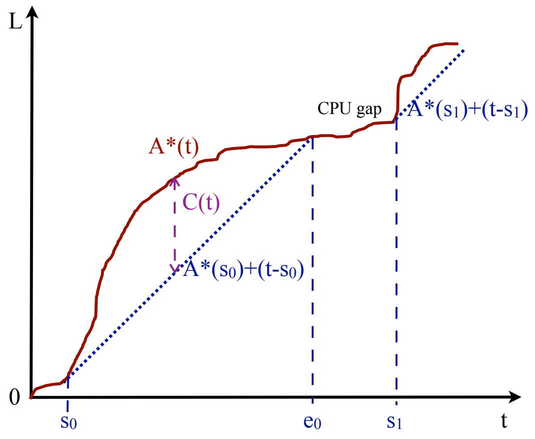


Fig. 10: Evolution of the workload. $A^*(t)$ is the accumulated workload. s_0 and s_1 are CPU gaps.

The goal now is to calculate the expression of $U(t)$ for a self-similar arrival process $A(t)$. According to Theorem 1 (and 2), our "on-off" task model asymptotically behaves like a fractional Brownian motion and the accumulated workload has the following expression:

$$A^*(t) = \mu t + \sigma Z_H(t) \tag{13}$$

where μ is the mean load, and σ^2 is the variance and can be calculated using the expressions of Theorem 1.

Assuming a processor with capacity 1 (that is, in one second it processes one unit of workload) and $\mu < 1$ (the stability condition) the current workload pending to execute can be expressed using the Reich's formula:

$$C(t) = sup_{s \leq t}(A^*(t) - A^*(s) - (t - s)) \tag{14}$$

$C(t)$ is also known as the *fractional Brownian storage* model [27]. The fractional Brownian storage model is the simplest long-range dependent storage system having strictly self-similar input variation. The impact of the Hurst parameter can be very clearly illustrated with this model.

The graphical interpretation of this storage model is shown in Figure 10. It can be seen that up to time s_0 there is no overload (load is less than 1). This means that jobs are not enqueued and all jobs are immediately dispatched. This is equivalent to the definition of a CPU gap. At time s_0 load increases and jobs are enqueued, so there are pending tasks. The end of this interval (e_0) is when all pending jobs are dispatched. The expression of the pending workload through this interval is $C(t) = A^*(t) - A^*(s_0) - (t - s_0)$. After e_0 there is a CPU gap. A new pending workload interval starts at time s_1.

Assuming that all deadlines are the same[2] (that is, $D_i = D \quad \forall i$) then, by definition, $C(t)$ is equivalent to $\sum_{\tau^i \in S(t)} C^i$. Therefore, the synthetic utilization can be expressed as:

$$U(t) = \sum_{\tau^i \in S(t)} \frac{C^i}{D^i} = \frac{C(t)}{D} \tag{15}$$

Our goal is to calculate the probability that $U(t)$ is greater than the schedulability bound B: $P(U(t) > B)$, and how it depends on the Hurst parameter H. Note that $P(U(t) > B)$ represents the *non*-schedulability probability:

$$P(U(t) > B) = P(C(t)/D > B) = P(C(t) > D \cdot B) \tag{16}$$

If we set $x = D \cdot B$, then, this problem is reduced to evaluate the tail behavior $(P(C(t)) > x))$ of a storage process $C(t)$ with a fractional Brownian workload as input. A bound of this tail behavior was introduced in [26]:

$$P(C(t) > x) \geq 1 - \Phi\left(\frac{1}{\sigma}\left(\frac{1-\mu}{H}\right)^H \left(\frac{x}{1-H}\right)^{1-H}\right) \tag{17}$$

where $\Phi(y)$ is the standard normal probability distribution, μ is the mean value of $A^*(t)$ of equation 13 (that is, $E[A^*(t)]$), σ is its variance, H is the Hurst parameter, and x is $D \cdot B$ (B is 1 for EDF scheduling). The value of μ is equivalent to the mean intensity or load L as defined in equation 10. Therefore, the deadline D must be greater than μ ($\mu < D$). Note that the ratio μ/D gives an idea of how tight is the deadline: lower values (loose deadlines) are easy to schedule than higher values (tight deadlines).

An asymptotic approximation to the tail distribution of equation 17 is:

$$P(C(t) > x) \sim e^{-cx^{2-2H}} \qquad x \to \infty \tag{18}$$

where c is a constant. It is easy to see that for $H = 1/2$ the previous expression is reduced to the well-known exponential asymptotic for a system with a Poisson arrival process:

$$P(C(t) > x) \sim e^{-c'x} \qquad x \to \infty \tag{19}$$

Formulas 18 and 19 are essentially different. The second allows more optimistic forecasts compared to the first one. Nevertheless, these formulas are asymptotical bounds that hold for a very high x. These approximations can be used in network models to calculate buffer sizes, since they are usually very high.

In our model for the utilization bound, x is very low, so asymptotical approximations cannot be used. That makes necessary to use expression 17 to evaluate the influence of the Hurst parameter on the tail behavior.

Figure 11 shows the tail probability $P(C(t) > x)$ for several Hurst values as a function of the deadline tightness μ/D with a workload variance (σ) of 0.2.

[2] Assuming that all deadlines are the same is a *weak condition*. For different values of D_i, we can obtain the same results if we select, for example, the minimum deadline $D = \min_i D^i$ (a *tight condition*) or the maximum deadline $D = \max_i D^i$ (a *loose condition*).

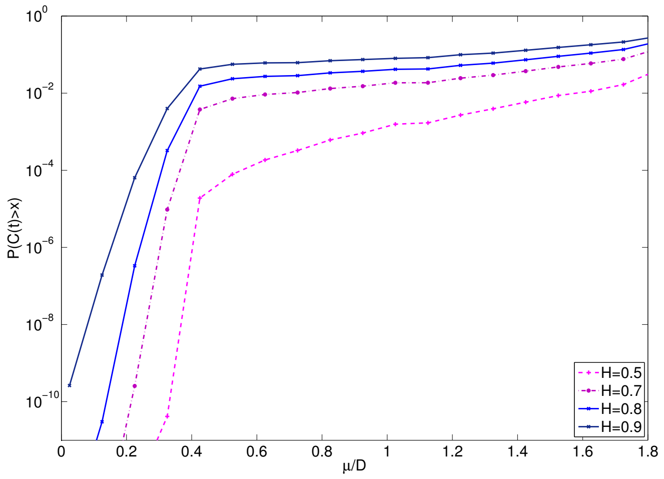


Fig. 11: Non schedulability probability depending of the deadline tightness for different Hurst values. The bound B=1 was set for EDF

It can be seen that the probability that $U(t)$ is greater than the schedulability bound B (that is, the non-schedulability probability) increases with the Hurst parameter. This increment is much more significant for loose deadlines ($\frac{\mu}{D} < 0.4$). In other words, a greater value of the Hurst index has the same effect that tighting the deadline. Similar results were obtained for the bound of deadline monotonic schedulers ($B = 1/(1 + \sqrt{1/2})$).

In summary, this section has shown that self-similarity of the workload has a strong influence on the schedulability of sporadic tasks in real-time system. These results are consistent with the results presented in queueing performance [8, 11, 26].

6 Experimental Evaluation

This section evaluates, through simulations, the efficiency of real-time schedulers when applied to self-similar workloads. The experimental evaluation process is the following (Figure 12): i) set the parameters of the workload model: System load (L), Deadlines (D) and Number (N) and type of tasks (T), ii) generate a sequence of jobs using this workload model: each job is a tuple (t, i, C, D) representing the arrival time (t), task stream number (i), computation time (C) and deadline (D), and iii) simulate a preemptive EDF scheduling for the previous sequence and obtain the deadline miss ratio (percentage of jobs that miss their deadlines). In every experiment the simulation time was 10,000s and each simulation was repeated 100 times in order to obtain 95% confidence intervals.

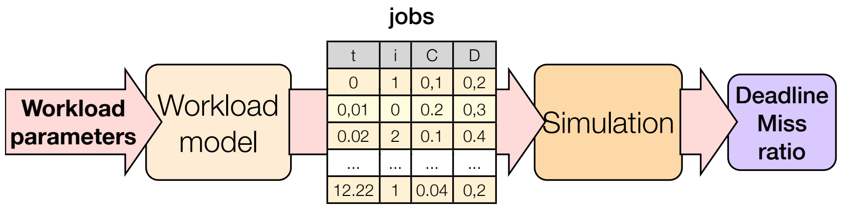


Fig. 12: Experimental evaluation process.

6.1 Sporadic workload

In the following set of experiments, a workload with a constant number of sporadic tasks ($N = 10$) was used. In each experiment, a global intensity, or system load L, and a deadline D were set. The corresponding load factor for each task was $L^i = L/N$. The execution time was kept constant and the inter-arrival time was heavy-tailed distributed. Specifically, tasks were assigned constant execution times ($C^i = \mathrm{C}(0.1s)$). This means that for a load of L^i, the inter-arrival mean $\bar{A}^i$ should be $(1/L^i - 1) \cdot 0.1$. Thus, the parameters for the Bounded-Pareto distribution used for inter-arrival times were the following: $p = 10s$, $\alpha = 3 - 2H$, and d was calculated in order to have a mean of $\bar{A}^i$: $d = \bar{A}^i \cdot (\alpha - 1)/\alpha$.

In the first experiment the values of L, D and H were set in order to have a non self-similar workload: $L = 0.75$, $D = 0.1$ and $H = 0.5$. The simulation results reflect no deadline miss. The experiment was repeated with the same L and D and increasing H to 0.8 in order to have a self-similar workload. The deadline miss ratio in this case was of 0.5%. This proofs that simplistic workload models can hide the probability of deadline misses of self-similar workloads and lead to inaccurate results.

The deadline miss ratio versus the Hurst value H is shown in Figure 13. The deadline miss ratio increases with the Hurst parameter. The first curve shows the deadline miss ratio for a utilization of 0.75 and a deadline of 1s. For Hurst parameters above 0.7 the miss ratio increases dramatically. The second and third curves show the deadline miss ratio with a more stringent deadline. The miss ratio increases more moderately and is more significant for Hurst values greater than 0.8.

Figure 14 shows the mean loss probability for several Hurst values as a function of the deadline tightness. This plot is very similar to the one presented in the analytical evaluation (figure 11).

The results of the experiments presented in this subsection confirm the results of the schedulability analysis of Section 5. The effect of the Hurst parameter is more important for loose-deadlines and lower utilizations (that logically has lower ratios). For tight-deadlines the influence of self-similarity on the deadline miss ratio is reduced.

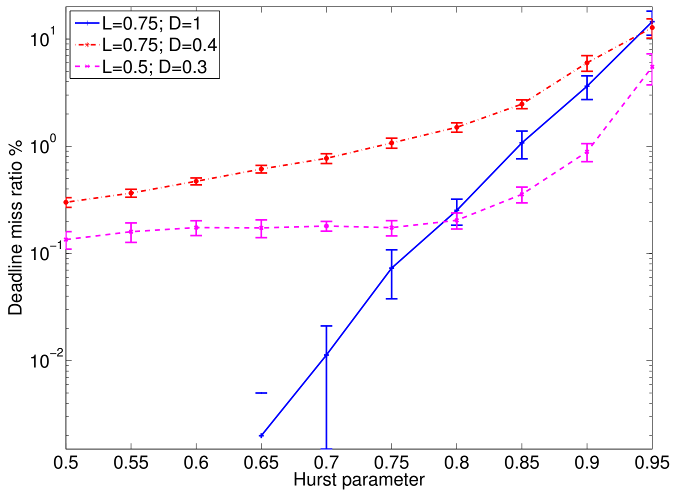


Fig. 13: Deadline miss ratio for a heavy-tailed workload with $N = 10$, constant execution times and inter-arrival times Pareto distributed. Note the log-scale of the loss ratio and the 95% confidence intervals.

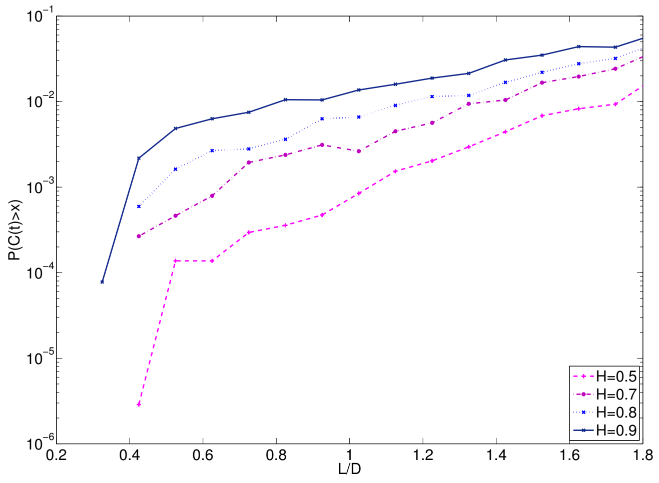


Fig. 14: Deadline mis ratio versus deadline tightness for the sporadic workload

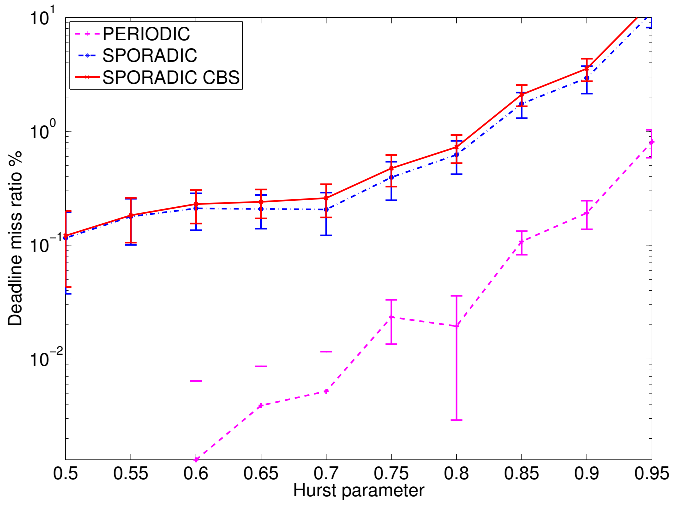


Fig. 15: Deadline miss ratio for a mixed workload. The PERIODIC and SPORADIC curves show the deadline miss ratio for periodic and sporadic heavy-tailed workload on the EDF scheduler. The SPORADIC CBS curve shows the deadline miss ratio for the sporadic task miss for the CBS scheduler.

6.2 Mixed workload

The following experiment uses a mixed set of periodic and sporadic tasks in order to show that the effect of self-similarity on such a workload is very significant. The periodic task set is the same of Table 3 except for P^1 (that is, the same non heavy-tailed tasks). That yields a mean utilization of 0.142 due to these periodic tasks. The sporadic task set consists of 5 heavy-tailed tasks with a constant execution time ($C^i = \mathrm{C}(0.1s)$) and a Pareto distributed inter-arrival time A^i. The Pareto parameters were calculated in the same way than in previous experiments in order to maintain a mean load of 0.1 for each task. The resulting system load was 0.64. The EDF simulation results show the miss ratio for sporadic and periodic tasks separately (see Figure 15). The results for sporadic tasks ("SPORADIC" curve) show that the miss ratio increases when H is greater than 0.7. For periodic tasks ("PERIODIC" curve), deadline misses occur when the Hurst parameter is higher than 0.6. In other words, increasing the self-similarity of aperiodic tasks causes deadlines misses in the non heavy-tailed periodic tasks.

6.3 Self-similar Overload Management

The impact of self-similar workloads on real-time systems really can be stated as the effect of introducing transient overloads. Previous experiments show that workloads with a Hurst index higher than 0.6 are more likely to produce such transient overloads.

Depending on system requirements, this problem can be handled in different ways. *Hard real-time* systems, where all tasks have hard deadline requirements, should be addressed using classic analysis methods (WCET). An increase of the resources (CPU, etc.) would be strictly required in order to face these transient overloads. In *soft real-time* systems the main goal is getting some CPU bandwidth (average utilization) and some amount of deadline misses can be tolerated. The survey of [33] describes several approaches to deal with overload management in these systems. A specially suitable approach is handling transient overloads through the resource reservation technique. A simple and effective mechanism for implementing temporal isolation under EDF is the Constant Bandwidth Server (CBS) [3]. A CBS is characterized by a budget c^i, a dynamic server deadline d^i, and by an ordered pair $(Q^i;T^i)$, where Q_i is the maximum budget and T^i is the period of the server. The ratio $U^i = Q^i/T^i$ is denoted as the server bandwidth. CBS behaves as a work-conserving algorithm, exploiting the available slack in an efficient way. An important property is that a CBS with bandwidth U^i will never demand more than U^iL for any interval of time of length L, regardless of the current task requests. This property allows the CBS to be used as a bandwidth reservation strategy in order to allocate a fraction of the CPU time to soft tasks whose computation time cannot be easily bounded.

The benefits of using a CBS scheduler on a self-similar system were evaluated in a new experiment based on repeating the previous experiment, but now each task was assigned a bandwidth or utilization. Periodic tasks were assigned a utilization U^i based on their WCET and the period T^i. Sporadic tasks were assigned a utilization based on their mean utilization. The results of the deadline miss ratio are shown in the curve “SPORADIC CBS” of Figure 15. In the performed simulations no periodic task ever missed its deadline, but it came at the cost of increasing the deadline miss rate of sporadic tasks. However, this increase is moderated and probably it can be allowed most of the times.

7 Conclusions

This paper has shown that self-similarity can appear in real-time systems with a combination of variable task *inter-arrival times* and *execution times*. This hypothesis are common in the processing of multimedia and network streams and simplistic task models of these systems can mask the probability of deadline misses and thus lead to inaccurate results.

The paper proposes a task simulation model based on the “on-off” model whose behavior captures the self-similar nature of the workload. Based on this model the influence of self-similarity on the scheduler efficiency has been evaluated analytically and trough simulations. The analytical study derives an expression for the dependence of the deadline miss ratio on the Hurst parameter. Simulations confirm the trends of the analytical expression and show how a a system that performs correctly under non self-similar workload starts to miss deadlines when the Hurst parameter is increased. This increase is dramatic for values above 0.6. Furthermore, the self-similar workload due to sporadic tasks can jeopardize the deadlines of periodic tasks. Based on this observation, we can conclude the convenience of using schedulers that enforce bandwidth isolation (like CBS) in these systems.

As a general conclusion, we can state that conventional task models, analysis methods and scheduling policies should be revised for self-similar workloads.

Acknowledgment

This work was developed under a grant from the European Union (FRESCOR-FP6/2005/IST/5-03402).

A Background on Self-Similarity

The analysis of stationary time series data or stochastic processes can reveal the property of self-similarity. This is the case, for example, of the CPU utilization over a sampling period. This has been well studied in network traffic. For a detailed discussion of self-similarity see [6].

A.1 Definition of Self-Similarity

A phenomenon that is self-similar looks the same at different scales on a dimension. This means that at different time scales (microseconds, milliseconds, seconds) the statistical properties are similar.

More formally, let $\{X_t : t = 0, 1, 2, \ldots\}$ ($\{X_t\}$ for short) be a discrete stationary stochastic process representing the evolution in time of a statistical distribution (for example, the workload of a system). Let σ^2 be the variance and let $r(k) = Cov(X_t, X_{t+k})/\sigma^2$ be the autocorrelation function of $\{X_t\}$. We define $\{X_t^{(m)}\}$ as the m-aggregated process of $\{X_t\}$, which is obtained by aggregating and averaging the data in X_t by blocks of size m:

$$X_t^{(m)} = \frac{1}{m}(X_{mt} + X_{mt+1} + \cdots + X_{m(t+1)-1}) \tag{20}$$

and $r^{(m)}(k)$ is defined as the autocovariance function of $\{X_t^{(m)}\}$. Aggregating really means multiplying the sampling period by a factor m. An important effect of the process aggregation is to smooth the traffic rate in each period. Therefore, the variation is reduced. The question of how this variation changes depending on the aggregation factor is related to the study of the *self-similarity* of a process.

A process is *distributionally* self-similar if the processes $\{X_t\}$ and $\{X_t^{(m)}\}$ have the same distribution, up to a scaling factor. A *self-similar process* has the property that, when it is aggregated, the new process has the same autocorrelation function as the original one. Formally, the process $\{X_t\}$ is called *second-order self-similar* with Hurst parameter $H = 1 - |\frac{\beta}{2}|$ if:

$$\begin{cases} Var(X_t^{(m)}) = \sigma^2 m^{-\beta} \\ r^{(m)}(k) = r(k) \end{cases} \quad m = 1, 2, \ldots \tag{21}$$

If the second condition only holds when $m \to \infty$, then a process is *asymptotically second-order self-similar.* If $0 < H < 0.5$, the process is Short-Range Dependent (SRD), and if $0.5 < H < 1$, the process is Long-Range Dependent (LRD).

The variance of the m-aggregated process can be obtained as:

$$Var(X_t^{(m)}) = m^{2H-2} \cdot Var(X_t) = m^{2H-2}\sigma^2 \tag{22}$$

Processes that are LRD exhibit correlations over a wide range of times scales, while processes that are SRD exhibit correlation functions that decay exponentially fast. This implies that time-aggregation quickly results in white noise characterized by the absence of any significant temporal correlations.

The Hurst parameter can be evaluated in several ways [6]. In this paper, we use the *variance-time plot*, which is the graph of the variance $log(Var(X_t^{(m)}))$ versus $log(m)$. With this graph, we can obtain the Hurst parameter by fitting a least-square line with slope $\beta = 2H - 2$ through the resulting points, ignoring those for small m.

Since a self-similar process has observable bursts at a wide range of scales, it can exhibit *long-range dependence*: values at any instant are correlated with values at all future instants. Although the terms *self-similarity* and *long-range dependence* are not exactly equivalent, we use them in an interchangeable fashion throughout the paper.

One of the advantages of using self-similar models is that the degree of variation on multiple time scales can be expressed using only a single parameter: the Hurst parameter.

A.2 Fractional Brownian Motion

A common way to model self-similar processes is using a fractional Brownian model. The *fractional Brownian motion* (denoted fBm) model can be viewed as an extension of the standard Brownian Motion models that have been used in heavy traffic analysis. Formally, the normalized fractional Brownian motion $\{Z_H(t) : t \geq 0\}$ is a continuous zero mean Gaussian process with stationary increments and variance $|t|^{2H}$. The correlation of the increments is characterized by the Hurst index, H. Unlike the standard Brownian motion, the fractional one has a long-range dependency property when $H > 1/2$. Since this process is self-similar: $Z_H(at) = |a|^H Z_H(t)$. Using the normalized fBM $Z_H(t)$, we can define a new fBM $A^*(t)$ with mean μ and variance σ^2:

$$A^*(t) = \mu t + \sigma Z_H(t) \tag{23}$$

In section 4 we see that the proposed "on-off" workload model behaves statistically as this fractional Brownian motion.

A.3 Heavy-tailed distributions

A distribution is heavy-tailed if

$$P(X > x) \sim x^{-\alpha}, \quad \text{as} \quad x \to \infty, 0 < \alpha < 2 \tag{24}$$

That is, if the asymptotic shape of the distribution is hyperbolic, it is heavy tailed. Heavy-tailed distributions have a number of properties that are different from exponential or normal distributions. If $\alpha \leq 2$, then the distribution has an infinite variance; if $\alpha \leq 1$, then the distribution has an infinite mean. Thus, as α decreases, a larger portion of the probabilistic mass function may be present in the tail of the distribution. In practical terms, a random variable that follows a heavy-tailed distribution can have extremely large values with non-negligible probability.

The simplest heavy-tailed distribution is the Pareto distribution:

$$F(x) = P(X \leq x) = \begin{cases} 0 & t < d \\ 1 - (\frac{d}{t})^\alpha & t \geq d, 1 < \alpha < 2 \end{cases} \tag{25}$$

where d is the minimal value of the distribution and α is known as the Pareto index.

The problem with using the Pareto distribution is its infinite variance. This poses problems when simulating a system with heavy-tailed distributions. Specifically, in [9], it is shown that simulations become infeasible for $\alpha < 1.5$. A practical approach for solving this problem is to bound the Pareto distribution [15]. A *Bounded-Pareto* distribution is characterized by three parameters: the Pareto index α, the minimal value d, and the largest value p:

$$F(x) = \begin{cases} 0 & t < d \\ 1 - \frac{(\frac{d}{t})^\alpha}{(\frac{d}{p})^\alpha} & d \leq t < p, 1 < \alpha < 2 \\ 1 & t \geq p \end{cases} \qquad (26)$$

It is easy to see that if p is very high, the distribution is very similar to the unbounded one. Therefore, for large values of p it can be assumed that it is heavy-tailed [15].

Finally, note that self-similarity as well as heavy-tailed distributions are convenient mathematical idealizations and can never be fully validated from finite data sets. However, these idealizations are powerful mathematical tools for modeling important aspects of time series.

References

1. Abdelzaher, T.F., Sharma, V., Lu, C.: A utilization bound for aperiodic tasks and priority driven scheduling. IEEE Transactions on Computers **53**(3), 334–350 (2004)
2. Abeni, L., Buttazzo, G.: QoS guarantee using probabilistic deadlines. In: Proc. of the Euromicro Confererence on Real-Time Systems (1999)
3. Abeni, L., Buttazzo, G.: Resource reservation in dynamic real-time systems. Real-Time Systems **37**(2), 123–167 (2004)
4. Anantharam, V.: Scheduling strategies and long-range dependence. Queueing Systems **33**(1-3), 73–89 (1999)
5. B.B.Mandelbrot: Long run linearity, locally gaussian processes, h-spectra and infinite variances. International Economics Review **10**, 82–113 (1969)
6. Beran, J.: Statistics for Long-Memory Processes. Chapman and Hall, London (1994)
7. Boxma, O., Zwart, B.: Tails in scheduling. SIGMETRICS Performance Evaluation Review **34**(4), 13–20 (2007)
8. Brichet, F., Roberts, J., Simonian, A., Veitch, D.: Heavy traffic analysis of a storage model with long range dependent on/off sources. Queueing Systems **23**(1), 197–215 (1996)
9. Crovella, M., Bestavros, A.: Self-similarity in world wide web traffic: evidence and possible causes. IEEE/ACM Transactions on Networking **5**(6), 835–846 (1997)
10. Dìaz, J., Garcìa, D., Kim, K., Lee, C., Bello, L.L., López, J., L.Min, S., O.Mirabella: Stochastic analysis of periodic real-time systems. In: Proc. of the 23rd IEEE Real-Time Systems Symposium,, p. 289ñ300 (2002)
11. Erramilli, A., Narayan, O., Willinger, W.: Experimental queueing analysis with long-range dependent packet traffic. IEEE/ACM Transactions on Networking **4**(2), 209–223 (Apr 1996)
12. Erramilli, A., Roughan, M., Veitch, D., Willinger, W.: Self-similar traffic and network dynamics. Proceedings of the IEEE **90**(5), 800–819 (May 2002)
13. Gardner, M.: Probabilistic analysis and scheduling of critical soft real-time systems. Phd thesis, University of Illinois, Urbana-Champaign. (1999)
14. Garrett, M.W., Willinger, W.: Analysis, modeling and generation of self-similar vbr video traffic. In: ACM SIGCOMM (1994)
15. Harchol-Balter, M.: Task assignment with unknown duration. Journal of ACM **49**(2), 260–288 (2002)
16. Harchol-Balter, M.: Foreword: Special issue on new perspective in scheduling. SIGMETRICS Performance Evaluation Review **34**(4), 2–3 (2007)
17. Harchol-Balter, M., Downey, A.B.: Exploiting process lifetime distributions for dynamic load balancing. ACM Trans. Comput. Syst. **15**(3), 253–285 (1997)

18. Hernandez-Orallo, E., Vila-Carbo, J.: Network performance analysis based on histogram workload models. In: Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 331–336 (2007)
19. Hernandez-Orallo, E., Vila-Carbo, J.: Analysis of self-similar workload on real-time systems. In: IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 343–352. IEEE Computer Society (2010)
20. Hernández-Orallo, E., Vila-Carbó, J.: Network queue and loss analysis using histogram-based traffic models. Computer Communications **33**(2), 190 – 201 (2010)
21. Hughes, C.J., Kaul, P., Adve, S.V., Jain, R., Park, C., Srinivasan, J.: Variability in the execution of multimedia applications and implications for architecture. SIGARCH Comput. Archit. News **29**(2), 254–265 (2001)
22. J.Beran, Sherman, R., Taqqu, M., Willinger, W.: Long-range dependence in variable-bit-rate video traffic. IEEE Transactions on Communications **43**(2), 1566–1579 (1995)
23. Leland, W., Ott, T.J.: Load-balancing heuristics and process behavior. SIGMETRICS Perform. Eval. Rev. **14**(1), 54–69 (1986)
24. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM **20**(1), 46–61 (1973)
25. Mandelbrot, B.: Self-similar error clusters in communication systems and the concept of conditional stationarity. IEEE Transactions on Communications **13**(1), 71–90 (1965)
26. Norros, I.: A storage model with self-similar input. Queueing Systems **16**(3), 387–396 (1994)
27. Norros, I.: Queueing beahavior under fractional brownian traffic. In: K. Park, W. Willinger (eds.) Self-Similar Network Traffic and Performance Evaluation, chap. 4. John Willey & Sons, New York, NY, USA (2000)
28. Park, K., Willinger, W.: Self-similar network traffic: An overview. In: K. Park, W. Willinger (eds.) Self-Similar Network Traffic and Performance Evaluation, chap. 1. John Willey & Sons, New York, NY, USA (2000)
29. Paxson, V., Floyd, S.: Wide area traffic: the failure of poisson modeling. IEEE/ACM Transactions on Networking **3**(3), 226–244 (1995)
30. Rolls, D.A., Michailidis, G., Hernández-Campos, F.: Queueing analysis of network traffic: methodology and visualization tools. Comput. Netw. **48**(3), 447–473 (2005)
31. Rose, O.: Statistical properties of mpeg video traffic and their impact on traffic modeling in atm systems. In: Conference on Local Computer Networls (1995)
32. Roy, N., Hamm, N., Madhukar, M., Schmidt, D.C., Dowdy, L.: The impact of variability on soft real-time system scheduling. In: RTCSA '09: Proceedings of the 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 527–532. IEEE Computer Society, Washington, DC, USA (2009)
33. Sha, L., Abdelzaher, T., årzén, K.E., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., Mok, A.K.: Real time scheduling theory: A historical perspective. Real-Time Systems **28**(2), 101–155 (2004)
34. Taqqu, M.S., Willinger, W., Sherman, R.: Proof of a fundamental result in self-similar traffic modeling. SIGCOMM Comput. Commun. Rev. **27**(2), 5–23 (1997)
35. Tia, T., Deng, Z., Shankar, M., Storch, M., Sun, J., Wu, L., Liu, J.: Probabilistic performance guarantee for real-time tasks with varying computation times. In: Proc. of the Real-Time Technology and Applications Symposium, pp. 164–173 (1995)
36. Vila-Carbó, J., Hernández-Orallo, E.: An analysis method for variable execution time tasks based on histograms. Real-Time Systems **38**(1), 1–37 (2008)
37. W.E.Leland, M.S.Taqqu, W.Willinger, D.V.Wilson: On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Transactions on Networking **2**(1), 1–15 (1994)
38. Willinger, W., Taqqu, M., Erramilli, A.: A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks. Stochastic Networks: Theory and applications pp. 339–366 (1996)
39. Willinger, W., Taqqu, M.S., Sherman, R., Wilson, D.V.: Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. IEEE/ACM Transactions on Networking **5**(1), 71–86 (1997)