Beyond the Weakly Hard Model: Measuring the Performance Cost of Deadline Misses

Paolo Pazzaglia
Scuola Superiore Sant'Anna, Pisa, Italy
paolo.pazzaglia@sssup.it

Luigi Pannocchi
Scuola Superiore Sant'Anna, Pisa, Italy
luigi.pannocchi@sssup.it

Alessandro Biondi
Scuola Superiore Sant'Anna, Pisa, Italy
alessandro.biondi@sssup.it

Marco Di Natale
Scuola Superiore Sant'Anna, Pisa, Italy
marco@sssup.it

Abstract

Most works in schedulability analysis theory are based on the assumption that constraints on the performance of the application can be expressed by a very limited set of timing constraints (often simply hard deadlines) on a task model. This model is insufficient to represent a large number of systems in which deadlines can be missed, or in which late task responses affect the performance, but not the correctness of the application. For systems with a possible temporary overload, models like the m-K deadline have been proposed in the past. However, the m-K model has several limitations since it does not consider the state of the system and is largely unaware of the way in which the performance is affected by deadline misses (except for critical failures). In this paper, we present a state-based representation of the evolution of a system with respect to each deadline hit or miss event. Our representation is much more general (while hopefully concise enough) to represent the evolution in time of the performance of time-sensitive systems with possible time overloads. We provide the theoretical foundations for our model and also show an application to a simple system to give examples of the state representations and their use.

2012 ACM Subject Classification Computer systems organization → Embedded software

Keywords and phrases control, real-time, cyber physical systems, weakly hard, deadline miss, performance

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2018.10

Supplement Material ECRTS Artifact Evaluation approved artifact available at `https://dx.doi.org/10.4230/DARTS.4.2.4`

1 Introduction and Motivation

The development of most control applications is based on the assumption that the design, definition and analysis of the controls functionality can be separated from the development and analysis of the software code implementing it. The functional model of the controls and of the program threads implementing them are connected by a suitable set of assumptions on the time properties of the code. In most cases, the assumptions relate to the activation periods, the maximum allowed response times (or deadlines) and possibly the output jitter.

© Paolo Pazzaglia, Luigi Pannocchi, Alessandro Biondi, and Marco Di Natale;
licensed under Creative Commons License CC-BY

In many instances, if the periodic tasks are schedulable within the deadlines, then the system is assumed to be correct. This assumption corresponds to the hard deadline model.

As highlighted by several authors (a comprehensive discussion of the issue is in [10]), this assumption is simplistic and may lead to overprovisioning of resources on one side (by requesting that all deadlines are met when in reality the system may be tolerant to some deadline misses) and the inability to model the impact of late responses as a performance degradation, given that the only possible outcome is a binary (feasible, infeasible) assessment of the system correctness.

To cope with the possibility of deadline misses, more sophisticated task models have been proposed, including analysis methods that compute the maximum lateness or number of consecutive deadline misses, or the Weakly Hard model, aimed at verifying whether the system can guarantee that at most m deadlines are missed for every set of K consecutive task instances. These models have been developed by the real-time analysis community, often inspired by generic requirements from controls developers, but mostly abstract from considerations on the performance of the controls.

Alternatively, the timing model of the software tasks is included in a general model of the system together with the model of the controls, that is, the two domains are not separated but jointly considered. Examples of these approaches include system models using hybrid or timed automata (such as those used by the Times tool [18]) and models that cosimulate the control functionality and the task timing, to assess the impact of scheduling delays on the performance (examples are the Jitterbug, TrueTime [10] and TRes tools [11]).

Contribution and Paper Structure

In this work, we propose to define a new abstraction for Cyber Physical Systems (CPSs) analysis that represents the performance degradation of the system in correspondence to possible deadline misses. In particular, the focus of the paper is on computing the evolution of control performance for a CPS in which its control tasks can suffer sporadic deadline misses, that can be described by a set of Weakly Hard constraints. We consider a task actuation implemented using the LET paradigm, where the control output is updated at the task deadlines. When a job misses its deadline, the control output is not updated. The LET implementation imposes fixed delays of the control output, thus enabling a precise analysis of the control system. First, the *freshness* of the control output is extracted for each step of the possible sequences of hit and missed deadlines, considering different handling methods for the deadline miss event. Then, the corresponding sequence of update matrices for the state variables is constructed, and a performance value is assigned to the sequence. The *sum of squared errors* is chosen as a representative performance index. The proposed approach allows extracting useful information such as worst-case performance bounds and critical sequences of deadline hits/misses with respect to a target performance.

Our model is more detailed than the Weakly Hard model (summarized in Section 2) since it considers the evolution of the system state in correspondence to miss events and, by taking into account the actual patterns of hits and misses, it includes an estimate of the system performance at each state. The proposed model is still much simpler than hybrid automata since it only considers the impact of a finite number of job completions and abstracts the time behavior by means of deadline miss/hit sequences. The model of the system assumed in this paper is summarized in Section 3, and the proposed approach with the overall objectives are discussed in Section 4.

We show how the size of the state description can be bounded by a set of scheduling assumptions, or as a result of timing analysis, and how the performance annotation can be computed for simple metric functions on Linear Time Invariant (LTI) systems. Furthermore, under the hypothesis that the system evolution is bounded by an exponential function, the number of steps in each analyzed sequence can be effectively limited with a bounded error on the performance index. Section 5 contains the description of how to compute the impact on the control values (update freshness) as a consequence of possible deadline misses, and Section 6 how to compute a state trajectory from sequences of hits and misses. These results are used in Section 7 to assign performance values to the sequences and possibly compute the worst-case values. Finally, in Section 8 we show the application of the proposed method to a case study consisting of a control of a Furuta pendulum.

1.1 State of the Art

The *weakly-hard* real-time schedulability analysis targets the problem of bounding the maximum number of deadline misses over a number of task activations. A dynamic assignment of priorities for streams with m-K requirements is proposed by Hamdaoui et al. [20] to reduce the probability of missing more than m deadlines every K iterations. Weakly hard real-time schedulability analysis can be traced back to the work of Bernat et al. [5] on the m-K model. The analysis in [5] and in other works assumes that there is an explicit initial state of the system, in which the initial offset of each task in the system is known. This limitation is removed in [34].

Recent developments in the study of overloaded systems allow to relax the requirement of knowing the initial system state. The approach proposed by Quinton et al. [23] consists in the *worst-case analysis* of a system model represented as the superposition of a typical behavior (e.g., of periodic task activations) that is assumed feasible, and a sporadic overload (i.e., a rare event). Under such an assumption, other works [16, 30] proposed methods for weakly-hard analysis that consists of two phases: 1) the system is verified to be schedulable under the typical scenario (by the classical hard analysis), and 2) when the system is overloaded, it can be guaranteed that out of K successive activations of a task, at most m of them will miss the deadline.

The analysis of overload conditions is also closely related to the co-design of control and CPU-time scheduling [3]. The influence of response times on the performance of control tasks has been studied in several works such as those by Xu et al [31, 32]. Aminifar et al. [2] proposed an integrated approach for controller synthesis, by selecting the task parameters that meet the expected control performance and guarantee the stability. The concept is later extended [1] to distributed Cyber Physical Systems, while in [15] FlexRay is considered as the communication medium. The m-K model has also been investigated in the co-design of controls (with respect to their performance) and scheduling [12, 24], and is used in [7, 8, 26, 27] to define the maximum number of samples (jobs) that can be dropped over any sequence (density of dropped samples), to guarantee a minimum level of quality to the controls. In [29] the problem of modifying the controller for improving the worst-case performance under m-K constraints is addressed. Moreover, the m-K model has been used for describing stability properties of a controlled system, e.g., to account for the maximum number of deadlines that can be missed in a row without making the system unstable (see [25]). Recent work by Blind and Allgöwer [6] showed that an unstable plant with feedback control that executes in open loop for finite time intervals under m-K constraints, can be subject to stability analysis by means of the Lyapunov method. In a subsequent work, Linsenmayer and Allgöwer [19] faced the problem of finding a controller that stabilizes the plant described in [6] under a given set of weakly hard constraints.

In the controls literature, Yoshimoto and Ushio [33] consider an overloaded real-time platform with multiple controllers. The system described is a LTI plant, with the deadline of the control task equal to the period of the task. They create an arbiter for skipping jobs that maintain the system schedulable while minimizing a performance degradation index based on the number of *consecutively* skipped jobs.

Frehse et al. [12] consider a Weakly Hard system, with a control task that is implemented using the Logical Execution Time (LET) paradigm [17]. The authors create a hybrid automaton that represent the connection between the physical system (described with piecewise affine functions) and the discrete controller, analysing the system with the TWCA approach [23]. They propose then the use of reachability analysis with the model checker SpaceEx to analyze if the trajectories guarantee the required performance.

2 The Weakly-Hard Task Model

The m-K model [20] and its generalization in the weakly-hard model [5] are an attempt at describing the impact of deadline misses on the *correctness* of the application. The approach can be used for a real-time system in which a given number of misses can be tolerated without critical consequences. When the number of deadline misses can be bounded in any time interval of a given length, the system is defined as *weakly-hard*.

In this paper, we are interested in Cyber-Physical Systems where control tasks have weakly-hard constraints. The main goal is to extract bounds for the control *performance* as a function of the weakly-hard constraints, and monitor the system at each step. To make the paper self-contained, this section recalls the standard definitions for weakly-hard real-time systems, and provides an overview of the major limitations of state-of-the-art approaches.

2.1 Definitions

In a weakly-hard system, the (m, K) constraint provides a bound on the number m of deadline misses that a task can experience every K instances (i.e., jobs). In the following, the definitions of *satisfaction set* and *hardness* of an (m, K) constraint, taken from the work of Bernat, Burns, and Llamosí [5], are used.

▶ **Definition 1** (Satisfaction set)**.** Given a constraint (m, K), the *satisfaction set* $\mathcal{S}_N$ of (m, K) is the set of all the sequences of hit and missed deadlines of length N that satisfy the constraint.

A sequence $s \in \mathcal{S}_N$ is represented by a string of N letters, using "M" for a deadline miss and "H" for a hit. When necessary, the sequence s will also be denoted as H/M. As a particular case, the satisfaction set of the constraint $(0, K)$ (for any K) includes sequences with all hits. This sequence is named hereafter as *H-sequence*. The H-sequence, also denoted as s_H, is the only sequence that satisfies all the possible (m, K) constraints.

When analyzing the weakly-hard properties of a real-time task, it is generally possible to extract a set of constraints under different values for K. A set of p weakly-hard constraints is denoted with Γ, and is defined as: $\Gamma = \{(m_1, K_1), (m_2, K_2), \ldots, (m_p, K_p)\}$. The definition of satisfaction set is then extended to a set Γ:

▶ **Definition 2.** Given a set Γ of weakly-hard constraints, the satisfaction set of Γ is the *intersection* of the satisfaction sets of all the constraints in Γ.

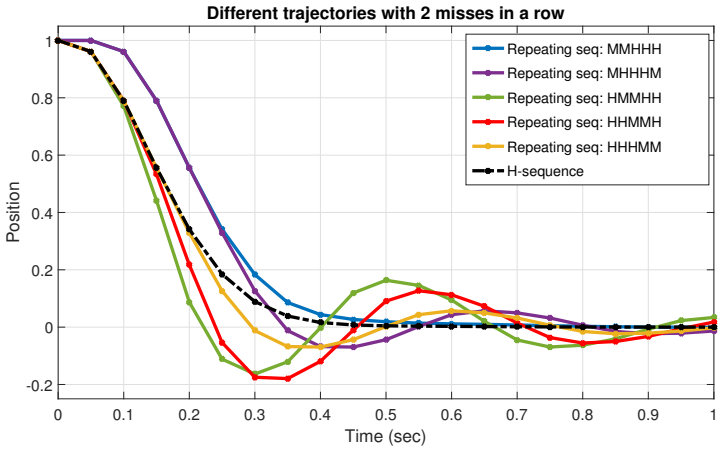

Figure 1 Comparing different state trajectories of a controlled roller moving a sheet of paper, with 2 deadlines missed in a row every 5 instances. Changing the order of the missed deadlines in a H/M sequence leads to different behaviors with different control performance.

2.2 Limitations of the Weakly-Hard Model

In this work, we are interested in finding a model that puts in correspondence the timing properties of a control task (expressed as a sequence of deadline hits and misses), the dynamics of the physical plant (the controlled system), and the control performance. Unfortunately, our objective cannot be achieved with a simple m-K weakly-hard model.

In fact, the weakly-hard model has several limitations when applied to the control domain. First of all, the (m, K) constraint is a model with a "binary" outcome, where either the system satisfies the constraint or not (e.g., for the purpose of stability analysis [25]), but it does not provide any information concerning the control performance that can be guaranteed when the constraint is met.

Another important limitation is that the *order* (i.e., the actual pattern) of deadline misses and hits cannot be fully described by using a combination of (m, K) constraints. This is true even when using the extended model proposed in [5], where the maximum number of *consecutive* deadline misses is considered. Indeed, *different H/M sequences in the same satisfaction set generally lead to different control performance values*, thus making the (m, K) constraint a coarse (and possibly misleading) description of the system when addressing the performance analysis. The explicit consideration of all the valid H/M sequences, together with the evolution of the system state during the sequence, may be important to enable a precise study of the control performance. For instance, if the system is near the steady state, the errors produced by an actuation that uses stale data (e.g., as a result of a deadline miss) are limited. However, if the system is in a transient condition, actuation errors will have a much higher impact. Since the *sensitivity* of the control performance changes with the system state and its evolution, the adoption of the existing weakly-hard models to analyze the control performance requires to always account for the overall worst-case scenario.

For example, consider the model of a *paper roller*, as described in [22] (pag. 40), which is controlled to zero with a periodic task with $T = 50ms$ and $D = 0.7T$. The control task satisfies a weakly-hard constraint with 2 deadline misses in a row every 5 instances. Five H/M sequences that satisfy the considered weakly-hard constraints are applied, and when a deadline is missed the control output is not updated. The results are reported in Figure 1 and

show that each H/M sequence leads to a different state trajectory. If the sum of quadratic errors (with respect to a reference) is considered as a control performance metric, then different values will be computed for each trajectory.

These observations motivate the development of a richer model, with an associated analysis technique to study the control performance under weakly-hard constraints.

3 System Model

This section presents the model of the considered Cyber-Physical System in terms of the physical plant, the controller, and its task implementation.

3.1 Plant and Control Description

We consider a physical system modeled as a Linear Time-Invariant (LTI) plant, multi-input-multi-output (MIMO), strictly causal, operating in a well-defined region $\Omega \in \mathbb{R}^n$ of the state space (possibly the entire state space $\mathbb{R}^n$). A continuous-time description of the plant is provided by the following state equation:

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t), \tag{1}$$

where $x_c(t)$ is the state vector, $u_c(t)$ the control output, and A_c and B_c are constant matrices of the dynamics with appropriate dimensions. We assume that a discrete-time controller is implemented as a periodic task τ_i, with period T_i and relative deadline $D_i \leq T_i$. The task is released at the system startup, i.e., time $t = 0$. The period T_i is chosen as a compromise between the stability properties of the system (e.g., the phase margin) and the schedulability constraints given by the available computational resources (speed of the processor, communication rates, etc.). A general heuristic for the choice of the sampling rate consists in guaranteeing that there are 4 to 10 samples during the rise time in response to a step input [28].

The time interval between the k-th and the $(k+1)$-th activation of the control task is defined as $[kT_i, (k+1)T_i)$.

The Logical Execution Time (LET) paradigm is adopted [12] for the control task. An example execution is shown in Figure 2 with three tasks for sensing (τ_S), control (τ_C), and actuation (τ_A). The k-th job of the control task uses the system state sensed at the activation time kT_i (without sensing jitter), and the output is used by the actuator at the deadline (time $kT_i + D_i$). Furthermore, we assume that the actuation value is kept constant until the next update, which occurs at time $(k+1)T_i + D_i$ (as shown by the $u[k-1]$, $u[k] = u[k+1]$ and $u[k+2]$ values in Figure 2). The choice of updating the control value at the task deadline leads to *regularity* of the output timing, with no output jitter, thus simplifying the control synthesis by means of classic techniques based on the assumption of a constant delay D_i (e.g., see [21] for the case with $D_i = T_i$). Moreover, the enforcement of fixed time instants for the control update enhances the *predictability* of the system.

The notation $x[k] = x_c(kT_i)$ is used to denote the discrete-time measure (or an estimate) of the system state. Analogously, $u[k]$ denotes the discrete-time representation of the control output. Note that, due to the delay D_i, $x[k]$ and $u[k]$ are not updated at the same time instant. We assume $x[k] = x[0], \forall k < 0$.

The output $u[k]$ is generated by a control function that is assumed to stabilize the discrete-time plant under consideration. The discrete-time representation of the plant is [4]

$$x[k+1] = A_d x[k] + B_{d1} u[k-1] + B_{d2} u[k], \tag{2}$$

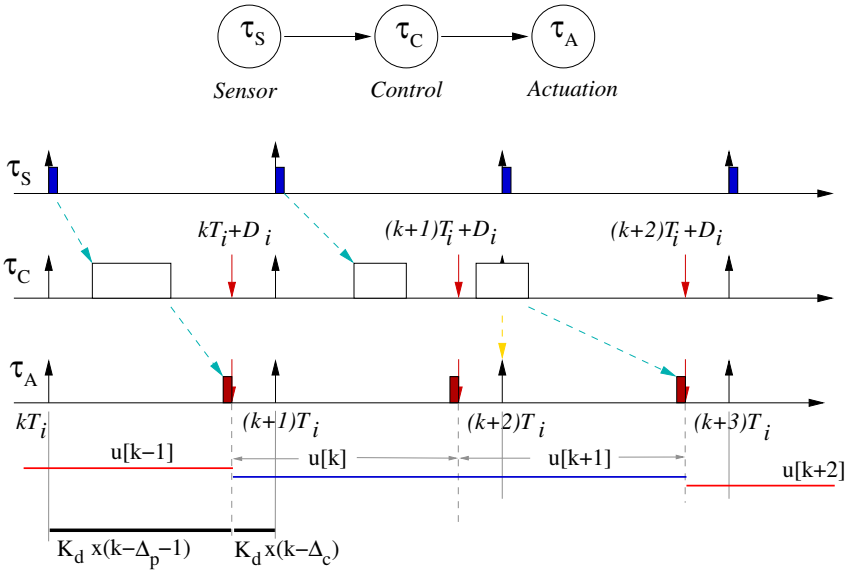

Figure 2 Time execution of sensor, control and actuation tasks.

where the involved matrices are defined as:

$$A_d = e^{A_c T_i}, \tag{3}$$

$$B_{d1} = \int_0^{D_i} e^{A_c s} ds\, B_c, \text{ and } \quad B_{d2} = \int_{D_i}^{T_i} e^{A_c s} ds\, B_c. \tag{4}$$

For this work, we consider a standard state-feedback controller of the following form:

$$u[k] = K_d(r - x[k]), \tag{5}$$

where K_d is the stabilizing control matrix, and r is the reference equilibrium state. Without loss of generality, from now on we consider that that the reference is equal to the zero vector, i.e., $u[k] = -K_d x[k]$. Finally, we assume that the initial state $x[0]$ (and thus the input $u[0]$) is known.

In the following, we are interested in reasoning about the output value before and after the deadline for each periodic instance. As shown in Figure 2, in the portion of the control period before the deadline, the value considered is $u[k-1] = K_d x(k - \Delta_p - 1)$, and in the other part is $u[k] = K_d x(k - \Delta_c)$. The delays Δ_p and Δ_c, which will be defined and analyzed in detail in Section 5, depend on possible deadline misses (for example, $\Delta_p = 0$ and $\Delta_c = 0$ for the first cycle in the Figure, and $\Delta_p = 1$ and $\Delta_c = 1$ for the third cycle).

3.2 Task Set Description

The control task τ_i is implemented on a real-time platform, together with N other tasks that can be either periodic or non-periodic. The Worst Case Execution Time (WCET) is assumed to be known for each task and the periodic control task τ_i has WCET $C_i \leq D_i$. We assume a fixed-priority, fully-preemptive scheduler, as in most established commercial standards. Of course, the approach is also applicable to multiple independent control tasks in execution on the same core.

3.3 The Task Model of the Controller

The effect of a deadline miss on the controller output depends on the structure of the controller task and the semantics of information passing. We assume that the task τ_i computes the actuation command using the data sensed at every activation. At its completion, the task copies the output data in a shared memory address, making it accessible to the process that handles the actuator. To match the considered LET paradigm, we also assume that the actuator is activated with the same period of τ_i (equal to T_i) but executed at its deadline (with very high priority and minimum jitter, possibly as a hardware implementation, Figure 2). The actuator reads the data from the shared memory and uses it as the actuation value during the next interval of length T_i. This means that if the output variable of τ_i is not ready at the deadline, the previously-stored value is used for the actuation.

To ensure a *one-to-one correspondence* between each element in an H/M sequence and the corresponding actuation update, we restrict our analysis to the case in which every execution of a control job is guaranteed to complete at least within $(T_i + D_i)$ time units from its activation. This means that the Worst-Case Response Time $WCRT_i$ of the task τ_i, is upper bounded as follows:

$$WCRT_i < T_i + D_i. \tag{6}$$

This condition ensures that there cannot be more than one pending invocation of τ_i at each deadline. The case in which the WCRT exceeds $(T_i + D_i)$ needs a richer description than H/M sequences, hence a considerable additional complexity for the model and the analysis: for this reason it is left as a future work. However, note that only part of the presented analysis relies on this assumption (further details are provided in Section 5.1.1).

4 Approach and Objective

The objective of this work is to propose a new model for relating the performance of control systems to schedulability conditions with possible deadline misses. Our model consists of a (finite) state-based representation, in which a control job belongs at every point in time to one state in the set. The new state is evaluated at each deadline hit or miss event, and represents a specific sequence of hits and misses. This state is annotated with a control performance value (as summarized in Figure 3).

This model can be used as a time contract between the design of the controls and their software (task) implementation. It enables

- the definition of monitors that can not only intercept at run time unforeseen timing faults, but also possible performance degradation that requires a recovery action, and
- the definition of the performance of the system at each deadline hit or miss event.

The approach presented in this paper is divided in steps:

1. A conventional m-K timing analysis (such as in [5] or [34]) is assumed to be performed on the tasks (using the timing model with WCETs). This analysis allows to bound the possible sequences of hits and misses and the number of states that are reachable. Safety monitors can be added to the system to guard against additional misses that can bring the system outside of the set of reachable states and would indicate an error in the estimate of the WCETs or other inaccuracies in the model of the task set.
2. For each state and each hit or miss event, two parameters (Δ_p, Δ_c) are computed (as shown at the bottom of Figure 2), that relate to the *freshness* of the control updates. This step requires only knowledge on the task execution model and the deadline handling

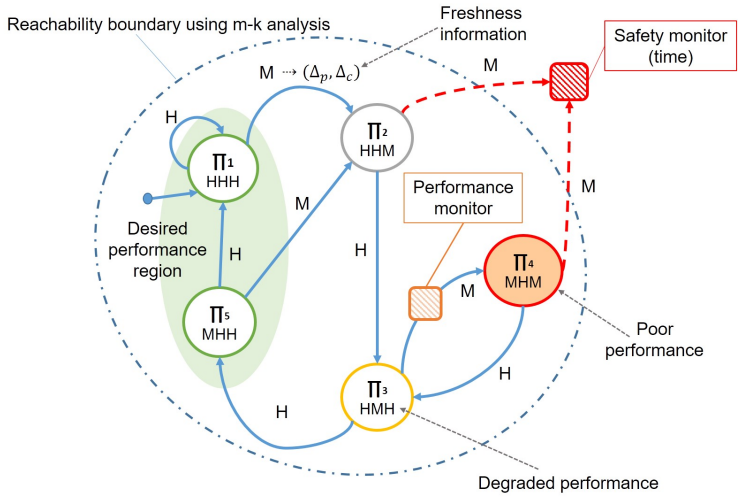

Figure 3 State machine representation of performance indexes for H/M sequences with constraint $(m, K) = (1, 2)$ and window of $N = 3$ steps. The reachability boundary restricts the analysis only to the possible combinations of N steps of hits and misses that satisfy the given (m, K) constraint.

strategy (extracted from the software implementation of the controls). The values Δ_p, Δ_c are computed using a set of state machines as shown in Figure 4 of Section 5 (different from the one of Figure 3).

3. Each state is annotated by the corresponding state update matrix, computed considering the plant model and the control parameters.
4. Performance bounds for the controlled system are extracted, and a description suitable for the on-line monitoring of the system behavior is created, in which each state is annotated by a performance matrix Π_i. Whenever the system transitions to a state with a degraded performance and before it can further progress into a poor performance condition, a monitor may be triggered to try to perform recovery actions.

In the following sections, steps number 2, 3 and 4 are analyzed in detail.

5 From Deadline Misses to Update Freshness

This section studies the impact of deadline misses on the *freshness* of the control updates. The presented approach is general enough to be applied to different deadline miss management policies. This analysis step only requires information related to the software implementation of the controller, i.e., it is independent of the physical system and the control parameters.

5.1 Handling Deadline Misses

Depending on the system implementation, and possibly on the configuration of the operating system, a job that misses its deadline can be handled in different ways. In this work, two common strategies are considered:

1. **Job killed:** the execution of the job that misses its deadline is aborted.
2. **Job continued:** a job that misses its deadline continues to execute until it completes. Multiple pending jobs are served in first-in-first-out order.

Each strategy not only results in a different effect on the timing properties (schedulability and response time) of the task set, but also in a different impact on the *freshness* of the control update and a different performance of the control system.

The update freshness is a parameter used to describe the age (as a number of control steps) of the current actuation value, which can be formally defined as follows.

▶ **Definition 3** (Update freshness Δ_k)**.** Let k' be the control step for which the state $x[k']$ is used to generate the k-th control update $u[k]$, with $k' \leq k$. The *update freshness* $\Delta_k \in \mathbb{N}_{\geq 0}$ is defined as $\Delta_k = k - k'$, that is

$$u[k] = -K_d x[k - \Delta_k].$$

Furthermore, we also introduce the *worst update freshness* Δ_{max}, that is the maximum number of aging steps for the active control value.

The proposed analysis considers each k-th control window (i.e., $[kT_i, (k+1)T_i)$). Due to the delay D_i in the generation of the actuation, two control outputs can exist within such windows, each with a corresponding update freshness. In the sub-window $[kT_i, kT_i + D_i)$ the control output is equal to the one generated in the previous control window, i.e., $u[k-1]$. Conversely, the control output in the window following the actuation update $[kT_i + D_i, (k+1)T_i)$ is equal to $u[k]$. Taking into account the update freshness, the two control outputs within the k-th control window are defined as

$$u[k-1] = -K_d x[k - 1 - \Delta_{k-1}] \tag{7}$$

$$u[k] = -K_d x[k - \Delta_k]. \tag{8}$$

For example, considering the example of Figure 2, the task instance released at time kT_i completes its execution within the deadline, and the actuation value $u[k]$ after the deadline is equal to $-K_d x[k]$ with update freshness 0. As the next deadline is missed, $u[k+1]$ is not updated and remains equal to $-K_d x[k]$, thus the value of the update freshness Δ_{k+1} is now equal to 1.

The next sections will show how to compute the sequences in time of the update freshness values Δ_{k-1} and Δ_k (also denoted as *update freshness pairs*) for all the possible H/M sequences that satisfy an (m, K) constraint. The possible time traces of the freshness pairs are represented using a state machine, where each node is described by a pair (Δ_{k-1}, Δ_k). For the sake of clarity, we will refer to the state machine defining the update freshness as *F-state machine*, and its vertexes as *F-states*. The F-state evolution rules for the job killed and job continued strategies need to be defined to construct the F-state machines.

5.1.1 Job Killed

Under this policy, whenever the job executing in the k-th control window misses its deadline, the actuator uses the previous value for the control output $u[k]$, consequently increasing the update freshness Δ_k. Hence, the definition of the F-state evolution rule for the update freshness follows.

Job killed: F-state evolution rule. Consider the k-th control window, characterized by the pair (Δ_{k-1}, Δ_k). The update freshness pair of the $(k+1)$-th control window is:

- $(\Delta_k, 0)$, if the $(k+1)$-th job of the control task hits its deadline;
- $(\Delta_k, \Delta_k + 1)$, otherwise (deadline miss).

The above rule comes from the following rationale. If a deadline hit occurs in the $(k+1)$-th control window, then the corresponding control output will use a fresh value, i.e., $\Delta_{k+1} = 0$. Otherwise, the control output of the k-th control window will also be used in the next window, with an increase of the update freshness. Therefore, the update freshness of $u[k+1]$ is given by the update freshness of $u[k]$ plus one period, i.e., $\Delta_{k+1} = \Delta_k + 1$.

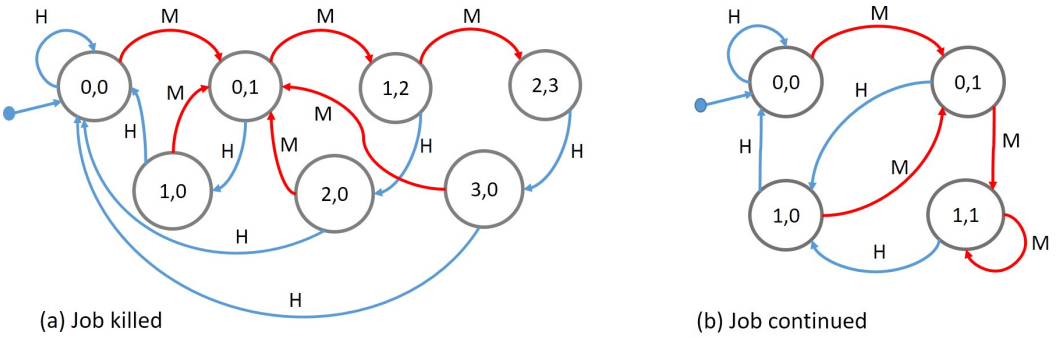

Figure 4 State machines expressing the evolution of the update freshness pair (Δ_p, Δ_c). The one related to the job killed strategy corresponds to the case with $\Delta_{max} = 3$.

Given a bound M^{max} on the number of consecutive deadline misses, it is possible to bound the maximum update freshness experienced in a control window, that is $\Delta_{max} = M^{max}$. Finally, it is important to observe that the above rule is independent of the assumption stated in Equation (6) that bounds the worst-case response time of the control task. In fact, under the job killed strategy, the response-time is always implicitly bounded by the relative deadline D_i.

5.1.2 Job Continued

Under the assumption of Equation (6), the evolution of the update freshness under the job continued strategy is determined by the following rule.

Job continued: F-state evolution rule. Consider the k-th control window, characterized by the pair (Δ_{k-1}, Δ_k). The update freshness pair of the $(k+1)$-th control window is:

- $(\Delta_k, 0)$, if the $(k+1)$-th job of the control task hits its deadline;
- $(\Delta_k, 1)$, otherwise (deadline miss).

Like for the job killed strategy, when the deadline is hit the next control output will dispose of a fresh value, i.e., $\Delta_{k+1} = 0$. On the other hand, if the deadline in the $(k+1)$-th control window is missed, the corresponding control output will be equal to $u[k]$. By Equation (6), a pending job of the control task cannot span more than two consecutive control windows. As a consequence, the state sensed at the beginning of the k-th control window, i.e., time kT_k, will be used at most for producing the $(k+1)$-th control output, which implies that the update freshness is implicitly bounded by one, i.e., $\Delta_{max} = 1$ (independent of M^{max}). Note that, despite this advantage with respect to the job killed strategy, the number of deadline misses under the job continued strategy can significantly increase because of self-pushing [34].

5.2 Constructing the State Machines for the Update Freshness

Given the F-state update rules introduced above, this section shows how to construct a F-state machine that describes the possible evolutions of the update freshness pairs. Each F-state machine is defined by a set of vertexes V, where each vertex is tagged with a freshness pair, and a set of directed edges E connecting the vertexes. An edge $e \in E$ is defined as a triplet consisting in the source vertex, the destination vertex, and a label that states if the vertex is taken when a deadline is hit or miss. For instance, $e = ((0,0),(0,1),M)$ is an edge

Algorithm 1 Construction of state machines for the update freshness.

```
1: global 𝒜 = {(0,0)}
2: global V = {(0,0)}
3: global E = {}
4:
5: function Entry_Point( )
6:     explore( (0,0), H )
7:     explore( (0,0), M )
8: end function
9:
10: function explore( (Δ_p, Δ_c), x )
11:     (Δ'_p, Δ'_c) = State evolution rule((Δ_p, Δ_c), x)
12:     V = V ∪ { (Δ'_p, Δ'_c) }
13:     E = E ∪ { ((Δ_p, Δ_c), (Δ'_p, Δ'_c), x) }
14:     if (Δ'_p, Δ'_c) ∉ 𝒜 then
15:         𝒜 = 𝒜 ∪ { (Δ'_p, Δ'_c) }
16:         explore( (Δ'_p, Δ'_c), H )
17:         if Δ'_c < M^max then
18:             explore( (Δ'_p, Δ'_c), M )
19:         end if
20:     end if
21: end function
```

that connects an F-state with update freshness defined by the pair $(0,0)$ to another vertex corresponding to the pair $(0,1)$ to represent the F-state evolution after a deadline miss.

These F-state machines are characterized by the following two properties: **(i)** the same update freshness pair can be obtained for different values of k (i.e., different control windows may have the same update freshness), and **(ii)** the evolution of the update freshness pair is only dependent on the immediately preceding pair (Δ_{k-1}, Δ_k). For this reason, when needed, the following short notation is adopted to get rid of the index k: Δ_c denotes the *current* update freshness, i.e., $\Delta_c = \Delta_k$ for the k-th control window; and Δ_p denotes the *previous* update freshness, i.e., $\Delta_p = \Delta_{k-1}$.

Algorithm 1 reports the pseudocode to generate the F-state machines. The algorithm exploits the recursive procedure EXPLORE that **(i)** computes the next state (Δ'_p, Δ'_c) by means of a state evolution rule given a deadline hit $(x = H)$ or miss $(x = M)$, **(ii)** adds and connects the new node to the F-state machine, and **(iii)** finally opens two recursive branches related to a deadline hit or miss, respectively. The algorithm termination is guaranteed by keeping track of the previously-visited states in the set $\mathcal{A}$ and by the bound Δ_{max}, both limiting the opening of recursive branches. Two illustrations of the resulting F-state machines are reported in Figure 4. It is worth noting that, given the strategy to handle the deadline misses and the bound Δ_{max}, such F-state machines are fixed and independent of the parameters of the control tasks and the control plant. As a consequence, they provide a very general model to study the evolution of the update freshness.

6 Computing State Trajectories

In the previous section, we defined the relation between H/M sequences and the freshness of the control update. The next step requires combining the update freshness parameters with the information coming from the plant model, which is used to compute the update matrices

of the plant state for each step of an H/M sequence. This leads to the definition of a chain of matrices that is then used to extract the state trajectory of the plant (subject to the control).

6.1 State Update Function and Stability Properties

Starting from Equation (2) and combining it with Equations (7) and (8), we derive the state update of the system with an arbitrary freshness pair (Δ_p, Δ_c). The resulting state equation can be rewritten as:

$$x[k+1] = A_d x[k] - B_{d1} K_d x[k-1-\Delta_p] - B_{d2} K_d x[k-\Delta_c] \tag{9}$$

In order to achieve a compact representation of the above equation for different freshness pairs (Δ_p, Δ_c), we introduce the *augmented plant state vector* $\xi[k]$ as

$$\xi[k] = \big[x[k];\, x[k-1];\, \cdots x[k-\Delta_{max}-1]\big], \tag{10}$$

which contains the state values of the last $\Delta_{max}+2$ control steps, i.e., from $x[k]$ to $x[k-\Delta_{max}-1]$. Note that $x[k-\Delta_{max}-1]$ is the last possible value of the state considered in Equation (9). Then, by leveraging this augmented state, it is possible to rewrite the state update function of the control system in Equation (9) as follows

$$\xi[k+1] = \mathbf{\Phi}(\Delta_p, \Delta_c)\xi[k]. \tag{11}$$

Here ,$\mathbf{\Phi}(\Delta_p, \Delta_c)$ is the *state update matrix*, which is defined as

$$\mathbf{\Phi}(\Delta_p, \Delta_c) = \begin{bmatrix} A_d & \cdots & -B_{d2}K_d & \cdots & -B_{d1}K_d & \cdots \\ \mathbf{I}_n & \mathbf{0}_n & \cdots & \cdots & \cdots & \cdots \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n & \cdots & \cdots & \cdots \\ \vdots & \cdots & \ddots & \ddots & \cdots & \cdots \end{bmatrix}, \tag{12}$$

where $\mathbf{0}_n$ and $\mathbf{I}_n$ are square matrices of zeros and the identity matrix, respectively, with dimension n (n denotes the size of the state space, i.e., $x[k] \in \mathbb{R}^n$). $\mathbf{\Phi}(\Delta_p, \Delta_c)$ is square with dimension $n \cdot (\Delta_{max}+2)$. The definition of $\mathbf{\Phi}(\Delta_p, \Delta_c)$ in Equation (12) is not a direct function of Δ_p and Δ_c: rather, Δ_p and Δ_c determine the position of the blocks $-B_{d1}K_d$ and $-B_{d2}K_d$. The value Δ_{max} determines the matrix size.

Using the state machines for the update freshness defined in the previous section, it is possible to assign each vertex (by means of the corresponding pair (Δ_p, Δ_c)) with a matrix $\mathbf{\Phi}(\Delta_p, \Delta_c)$. For instance, the matrices for the state machine when the job-continue strategy is used (reported in Figure 4(b)) are:

$$\mathbf{\Phi}(0,0) = \begin{bmatrix} A_d - B_{d2}K_d & -B_{d1}K_d & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix} \tag{13}$$

$$\mathbf{\Phi}(0,1) = \begin{bmatrix} A_d & -(B_{d1}+B_{d2})K_d & \mathbf{0}_n \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix} \tag{14}$$

$$\mathbf{\Phi}(1,0) = \begin{bmatrix} A_d - B_{d2}K_d & \mathbf{0}_n & -B_{d1}K_d \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix} \tag{15}$$

$$\mathbf{\Phi}(1,1) = \begin{bmatrix} A_d & -B_{d2}K_d & -B_{d1}K_d \\ \mathbf{I}_n & \mathbf{0}_n & \mathbf{0}_n \\ \mathbf{0}_n & \mathbf{I}_n & \mathbf{0}_n \end{bmatrix}. \tag{16}$$

Each such matrix corresponds to a possible mode in which the control system can operate, and such modes can be reached as a function of the pattern defined by an H/M sequence depending on the state machine that defines the update freshness. Furthermore, not all the possible transitions between such modes are actually possible: only the subset of H/M sequences that belong to the satisfaction set of Γ is possible, limiting the possible paths in the state machine of the update freshness. In this view, the control system under analysis can be considered as a particular case of a *constrained switched linear system* [29].

Equation (11) is particularly useful when studying the *stability* of the controlled system. According to the choice of K_d discussed in Section 3, the system is stable when no deadline misses occur (i.e., the dynamic related to matrix $\mathbf{\Phi}(0,0)$ is stable). However, no stability properties are guaranteed for the other operating modes. A performance analysis is meaningless for an unstable system, therefore all possible mode switches must lead to a stable behavior. An interesting approach for defining a controller that is stable for each possible H/M sequences (even if related to a simplified model with respect to the one presented here) is presented in [19]. However, in order to dispose of a bound on the number of steps of the H/M sequences under analysis, we require the stronger condition of *exponentially stability*: given the matrices $\mathbf{\Phi}(\Delta_p, \Delta_c)$, this property can be verified with the technique presented by Yu and Zhang in [35]. More details are provided in Section 7.2.

6.2 Mapping H/M Sequences to State Trajectories

After assigning a matrix $\mathbf{\Phi}(\Delta_p, \Delta_c)$ to each vertex of the state machine for the update freshness, it is possible to obtain a *sequence* of such matrices given an initial vertex and an H/M sequence. The compact notation $\mathbf{\Phi}_k$ denotes the matrix $\mathbf{\Phi}(\Delta_p, \Delta_c)$ obtained at the k-th step of an arbitrary H/M sequence. If the initial state $\xi[0]$ is known, it is possible to recursively compute the $(k+1)$-th state of the system as follows:

$$\begin{aligned}
\xi[k+1] &= \mathbf{\Phi}_k \xi[k] \\
\xi[k] &= \mathbf{\Phi}_{k-1} \xi[k-1] \\
&\dots \\
\xi[1] &= \mathbf{\Phi}_0 \xi[0],
\end{aligned}$$

so obtaining the following compact form:

$$\xi[k+1] = \mathbf{\Phi}_k \mathbf{\Phi}_{k-1} \cdots \mathbf{\Phi}_0 \xi[0]. \tag{17}$$

Thus, by simply multiplying a set of matrices $\mathbf{\Phi}_k$, it is possible to compute the corresponding *state trajectory* of the system. By considering each possible valid path in the state machine of the update freshness, it is also possible to compute all state trajectories subject to the weakly-hard constraints of the control task. To practically enable such computations, an initial state (in terms of update freshness) and a given length for H/M sequences are required. The former can be selected as the one characterized by the pair $(0,0)$, which matches the condition at the system startup. A bound on the length of H/M sequences is provided in the next section. Finally, it is worth noting that the initial state $\xi[0]$ contributes to the state trajectory with a fixed proportional constant. This means that the initial state can be treated as a *scaling factor*, while the *shape* of the trajectory is only determined by the sequence of hit and missed deadlines. This consideration is fundamental for the performance analysis that will be addressed in the following section, as it allows a significant reduction of its computational complexity.

7 Assigning Performance Values to H/M sequences

The first objective of this section is to assign a control performance index to each state trajectory, computed for a given H/M sequence (as defined in the previous section). As a result, H/M sequences are associated with a performance value, enabling the computation of the worst-case system performance over all the possible H/M sequences. Then, for the purpose of monitoring the evolution of the performance online, we provide a richer state-based performance model.

7.1 Mapping Trajectories to Performance

The proposed approach can be used with different performance metrics: as a representative case, we focus on the *sum of quadratic error* of the augmented plant state vector, which is formally defined for a given H/M sequence s as

$$P(s) = \sum_{i=0}^{N-1} \xi[i]^T \xi[i], \tag{18}$$

where N is the length of the sequence s. This index is indeed the discrete representation of the widely-adopted *integral of squared error* [14]. Note that our approach is also compatible with other performance indexes. Combining Equation (18) with Equation (17), we obtain the following expanded expression for $P(s)$:

$$\begin{aligned} P(s) &= \sum_{i=0}^{N-1} \xi[i]^T \xi[i] \\ &= \xi[0]^T \Big(\mathbf{I} + \mathbf{\Phi}_0^T \mathbf{\Phi}_0 + \mathbf{\Phi}_0^T \mathbf{\Phi}_1^T \mathbf{\Phi}_1 \mathbf{\Phi}_0 + ... + \mathbf{\Phi}_0^T \mathbf{\Phi}_1^T \cdots \mathbf{\Phi}_{N-1}^T \mathbf{\Phi}_{N-1} \cdots \mathbf{\Phi}_1 \mathbf{\Phi}_0 \Big) \xi[0] \\ &= \xi[0]^T \mathbf{\Psi}(s) \xi[0] \end{aligned} \tag{19}$$

The resulting matrix $\mathbf{\Psi}(s)$ is then a function of the ordered system states during the trajectory determined by the sequence s.

Likewise Equation (17), the initial state $\xi[0]$ can be treated as a proportional constant value also for the performance $P(s)$. This means that an order between the performance of different H/M sequences can be defined independently of the initial state. Motivated by this, we select the norm of $\mathbf{\Psi}(s)$, defined as $\Pi(s) = ||\mathbf{\Psi}(s)||_2$, as a scalar performance index.

7.2 Bounding the Number of Steps

To bound the complexity of the analysis, a *finite horizon* approach is selected, with N steps for the considered H/M sequences. Theoretically speaking, an exact performance analysis should consider an infinite horizon: however, given an arbitrary small error, a limited number of steps is sufficient to obtain a performance measurement.

As a lower bound, the value of N must never be less than the maximum window size K of the weakly-hard constraints, in order to avoid pathological cases where the constraint is not even completely defined on an input sequence. From a control perspective, it is important that the resulting performance analysis is applied to a sufficiently long interval of control steps, such that meaningful information can be extracted. For instance, the number of control steps N should be sufficient to include the settling time of the step response of the system. In general, the interval size must be carefully chosen to include the step response of *all* the possible H/M sequences. As the global switching system is exponentially stable by

hypothesis, following the results presented in [35], all the possible response dynamics of the plant state can be upper-bounded by an exponential function. Thus, a safe upper bound for N can be computed as the number of steps for which the exponential envelope converges within a given error of the reference.

7.3 Worst-Case Performance

The worst-case performance can be computed as a function of the weakly-hard constraints of the control task. In this paper, the performance is a quadratic error metric, and is therefore formally a cost; for a true performance (positive) metric, the maximum in (20) (in the following definition) should become a minimum.

▶ **Definition 4** (Worst-Case Performance)**.** Given a set Γ of weakly-hard constraints for the control task and a number of steps N, the worst-case performance for Γ is the *maximum* value of $\Pi(s)$ provided by all sequences in the satisfaction set of Γ with length N, i.e.,

$$WCP(\Gamma, N) = \max_{s \in \mathcal{S}_N} \Pi(s). \tag{20}$$

The normalized worst-case performance $WCPn(\Gamma, N)$ is computed with respect to the sequence s_H of all hits and is defined as

$$WCPn(\Gamma, N) = \frac{WCP(\Gamma, N)}{\Pi(s_H)}. \tag{21}$$

Interestingly, the worst-case performance is not necessarily associated with the sequence that has the largest number of deadline misses in the satisfaction set of Γ. Also, the proposed analysis allows to discern the effects on the performance of the *order* of deadline misses in the H/M sequence. Deadline misses in the early steps of the sequence s typically lead to worse performance with respect to deadline misses that occur late in the sequence. This behavior can intuitively be explained by the results derived by Yu and Zhang in [35]. In Equation (17), the j-th extended state $\xi[j]$ is computed as the result of the multiplication of j matrices $\mathbf{\Phi}_j$. Early misses contribute to the first terms, e.g., with only two or four matrices in the product. Since some matrices related to steps with missed deadlines can determine an unstable steady-state system (i.e., operating modes in which the system can diverge) these terms can result in a system that temporally tends to diverge from the control reference. Otherwise, since a switching control system is stable when the norm of the asymptotic multiplication of such matrices tends to zero, it is expected that longer terms (such as those towards the end of Eq. (17) representing late events in the hit/miss sequence) tend to be inherently more stable and contribute less to the dynamic (the unstable modes are dampened due to the asymptotic stability).

7.4 State-Based Representation of the Performance

The last step of our analysis is creating a *state machine* where every vertex represents the performance value of a H/M sequence of N control steps. For the sake of clarity, the states of this machine will be denoted as *P-states*. The edges between vertexes represent one step in the H/M sequence (hence a hit or missed deadline).

In the resulting P-state machine, it is possible to identify sets of sequences that lead to various degrees of *acceptable* or *poor* performance. Performance monitors can be inserted into the system for controlling (and possibly avoiding) transitions to a poor performance sequence. This can be obtained by increasing the priority of the control task, shedding higher

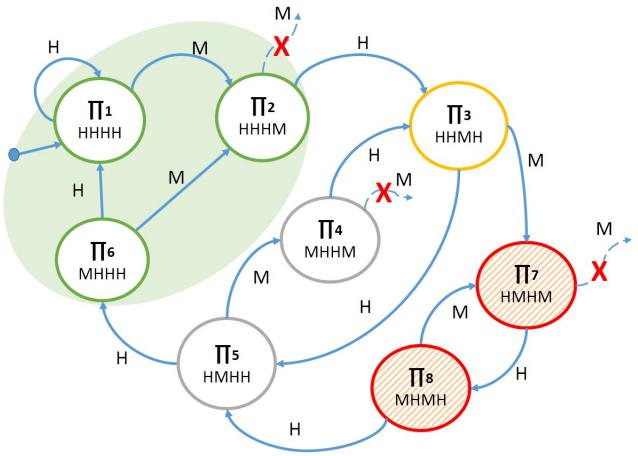

Figure 5 State machine representation of performance indexes for H/M sequences, for the case with $\Gamma = (1, 2)$ and $N = 4$. The vertexes in green are the best values for performance, the red ones are undesirable values and the orange one is a sequence that may lead to an undesirable behavior. The edges marked with a red X are unfeasible transitions.

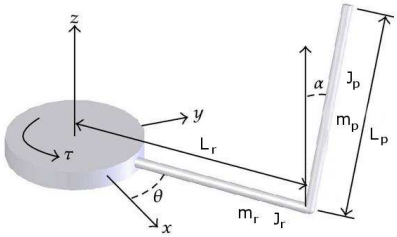

Figure 6 Scheme of the Furuta inverted pendulum, as the plant model used in the case of study.

priority load, or switching to a less computationally expensive implementation. Interesting information can also be extracted by analyzing the number of hits that are needed to recover from sequences with *poor* performance. If an unacceptable sequence is found, the design of the system must be changed in order to compensate this behavior. The solution can be, e.g., changing the control law parameters, or modifying some implementation strategies, such as the task implementation or the deadline miss handling policy.

The resulting P-state machine is a powerful tool for the performance analysis or possibly for the synthesis of runtime monitors controlling the evolution of the system and triggering recovery actions when needed. An illustrative example of the P-state machine for performance analysis is shown in Figure 5. Some transitions are marked with a red X and correspond to transitions that should never occur because of the results of the weakly hard analysis. The vertexes are colored differently following their performance values and critical P-states leading to poor performance can be easily identified.

8 Case Study

In this section, the proposed methodology is applied to a case study, consisting of a small rotary inverted pendulum (Furuta's Pendulum [13]) as illustrated in Figure 6. The objective of the control is to keep the pendulum in the upright position, which is an unstable equilibrium.

Table 1 Numerical values for the parameters describing the Furuta pendulum.

Parameters	Arm	Pendulum	Units	Motor		Units
m	0.4	0.02	kg	k_t	0.768	$N \cdot m/A$
L	0.1	0.2	m	k_e	0.768	$V \cdot s/rad$
l	0.06	0.1	m	R_m	3.3	Ω
J	0.0018	$2.67 \cdot 10^{-4}$	$kg \cdot m^2$	J_m	0.0015	$kg \cdot m^2$
b	0.0004	0.005	$kg/(m \cdot s)$			

The geometry and mass properties of the system are described by the set of parameters (L_p, l_p, J_p, m_p) and (L_r, l_r, J_r, m_r), which are the length, barycenter position, inertia and mass of pendulum stick (index p) and arm (index r), respectively. The viscous damping on the arm and pendulum joints have coefficients b_r and b_p, respectively. The state vector of the considered plant is defined as $x = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^T$, where θ is the angular position of the arm and α is the position of the pendulum with respect to the vertical. The controlled input voltage drives an electric motor on the arm joint, producing a torque Λ. The equation relating the voltage and the generated torque is $\Lambda = (k_t(V_m - k_m\dot{\theta}))/R_m$, where k_t is the torque constant of the motor, V_m is the voltage applied, k_e is the back e.m.f constant, and R_m is the resistance of the motor winding. The balancing control problem is considered, using a model linearized in the neighborhood of the upward position of the pendulum arm. Under the defined state vector, the following Linear Time Invariant (LTI) model [9] is obtained:

$$A_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m_p^2 L_p^2 L_r g}{\Upsilon} & \frac{J_p(-b_m - b_r)}{\Upsilon} & \frac{-m_p L_p L_r b_p}{\Upsilon} \\ 0 & \frac{(J_r + m_p L_r^2) g m_p L_p}{\Upsilon} & \frac{m_p L_r L_p (-b_m - b_r)}{\Upsilon} & \frac{-(J_r + m_p L_r^2) b_p}{\Upsilon} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ 0 \\ \frac{J_p k_t}{R_m} \\ \frac{m_p L_r L_p k_t}{R_m} \end{bmatrix}$$

where $b_m = k_e k_t / R_m$ and $\Upsilon = (J_r + m_p L_r^2) J_p - m_p^2 L_r^2 L_p^2$.

Performing the discretization of the system, following Equations (3) and (4), using a sample time $T_i = 0.1s$ and deadline $D_i = 0.2T_i$, and substituting the values with ones provided in Table 1, the model can be expressed in the following numerical form:

$$A_d = \begin{bmatrix} 1.0000 & 0.0036 & 0.0188 & -0.0007 \\ 0 & 1.2282 & -0.0332 & 0.0503 \\ 0 & 0.0266 & 0.0081 & -0.0032 \\ 0 & 3.7230 & -0.2448 & 0.2794 \end{bmatrix}, \quad B_{d1} = \begin{bmatrix} 0.0381 \\ 0.0109 \\ 0.0261 \\ -0.1006 \end{bmatrix}, B_{d2} = \begin{bmatrix} 0.0666 \\ 0.0320 \\ 1.2539 \\ 0.4166 \end{bmatrix}$$

The control law used for this example is a stabilizing static feedback of the state with constants $K_d = [-1.4557, 62.8126, -2.0459, 2.7210]$.

Experimental Setup

The tests have been carried out covering all the possible combinations of hits and misses that satisfy different (m, K) constraints, with an analysis horizon of $N = 20$ steps. For the purposes of this case study, this value is sufficient to capture the main features of the dynamics when different weakly-hard constraints are used. Both the job killed and job continued strategies have been considered. The scheduling parameters (m, K) are computed for $K \in [5, 8]$ and $m \in [1, K-3]$, respectively. In order to study the evolution of the control performance, for each strategy, the corresponding matrices $\mathbf{\Phi}_k$ have been computed.

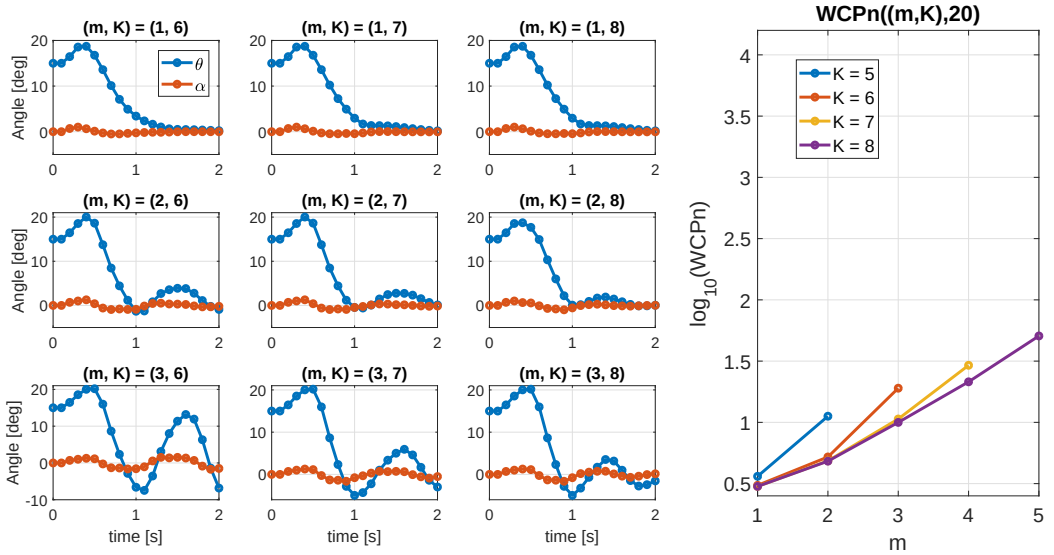

Figure 7 Results related to the **job continued** strategy. *Plots on the left*: state trajectories for the H/M sequence that leads to the worst-case performance under different (m, K) constraints. *Plot on the right*: normalized worst-case performance for the considered configurations.

Experimental Results

The experiments produced a big volume of data, thus, for the sake of clarity and compactness, only the most representative results have been represented and discussed here.

By considering different weakly-hard constraints (m, K), Figure 7 reports the trajectories of the system in correspondence to the H/M sequence that originates the worst-case performance. The figure also reports the normalized performance for each of the considered constraints. The plots of the state trajectory (on the left) show that, fixed the window K, the settling time and the width of possible oscillations (of the trajectory) increase as the number of deadline misses m increases. Looking at the plot on the right, it is noticeable that the worst-case performance is monotone with respect to the dimension of the scheduling window K. In particular, note that for the same number of deadline misses, better performance can be obtained for larger values of K. This can be explained by the fact that deadlines are to be placed with a lower density.

Figure 8 shows the experimental results under the same configurations considered in the previous figure but related to the job kill strategy. While it is possible to observe the same monotonic trend of the performance with respect to parameter K, the value of the normalized WCP is worse than the one obtained with the job continued strategy. As a matter of fact, for the same constraint (m, K), the job killed strategy also shows worse trajectories in terms of settling time and oscillations.

Also note that this strategy leads to unstable configurations (e.g., see the case for $(m, K) = (3, 6)$ in the figure), which tend to arise to the larger value of Δ_{max}. The performance value has also been computed for the particular cases of unstable systems only for the purposes of comparison with the job continued strategy of Figure 7. These results pose an interesting observation on the trade-offs between schedulability and performance analysis for the tested system: while the job killed strategy can improve the system schedulability, lowering the computational workload and possibly simplifying schedulability analysis, it tends to provide worse performance. For this reason, we believe that the proposed analysis framework may be valuable when facing with control-scheduling co-design activities.

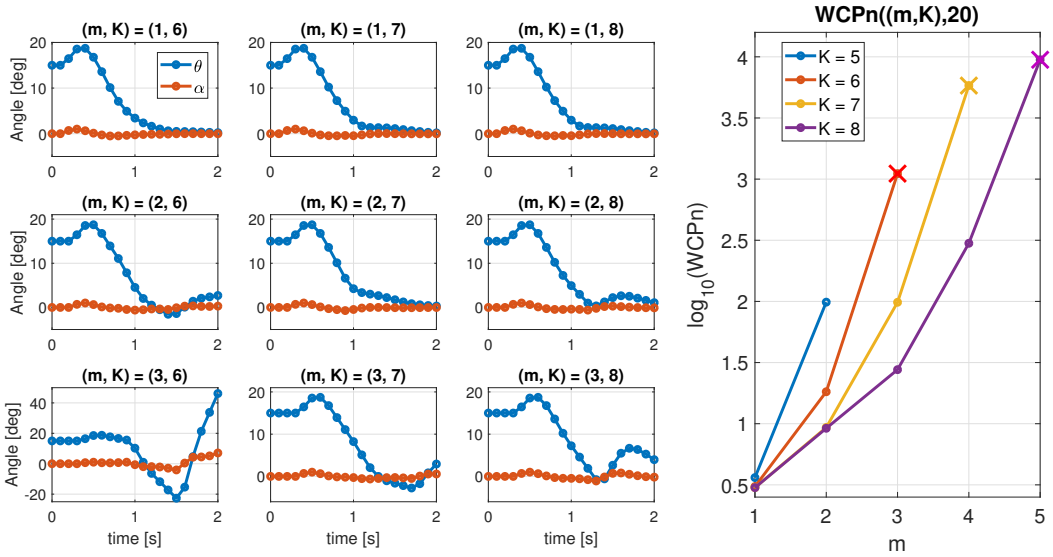

Figure 8 Results related to the **job killed** strategy. *Plots on the left*: state trajectories for the H/M sequence that leads to the worst-case performance under different (m, K) constraints. *Plot on the right*: normalized worst-case performance for the considered configurations. The elements marked with a "x" refer to unstable configurations.

9 Conclusions

We presented a new methodology to compute a state machine abstraction that allows to relate the performance of a control application to a sequence of deadline hits and misses, subject to weakly-hard constraints, in the execution of a control task updating a control value. We show the possibility of computing the state machine by formal derivation assuming knowledge of the deadline management policy, the LTI system and a simple performance metric. The size of our state representation is constrained by leveraging worst case timing analysis and assuming a finite time horizon.

Future work will include consideration of additional performance metrics and the possible use of simulation techniques to compute the state abstraction when an analytical derivation is not possible.

References

1 Amir Aminifar, Petru Eles, Zebo Peng, and Anton Cervin. Control-quality driven design of cyber-physical systems with robustness guarantees. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1093–1098. EDA Consortium, 2013.

2 Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. Designing high-quality embedded control systems with guaranteed stability. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, pages 283–292, 2012.

3 Karl-Erik Årzén, Anton Cervin, Johan Eker, and Lui Sha. An introduction to control and scheduling co-design. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, pages 4865–4870, 2000.

4 Karl Johan Åström and Björn Wittenmark. *Computer-controlled systems: theory and design*. Prentice-Hall, 1997.

5 Guillem Bernat, Alan Burns, and Albert Liamosi. Weakly hard real-time systems. *IEEE transactions on Computers*, 50(4):308–321, 2001.

6 Rainer Blind and Frank Allgöwer. Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 7510–7515. IEEE, 2015.

7 Tobias Bund and Frank Slomka. Controller/platform co-design of networked control systems based on density functions. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, pages 11–14, 2014.

8 Tobias Bund and Frank Slomka. Worst-case performance validation of safety-critical control systems with dropped samples. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, pages 319–326, 2015.

9 Benjamin Seth Cazzolato and Zebb Prime. On the dynamics of the furuta pendulum. *Journal of Control Science and Engineering*, 2011:3, 2011.

10 A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. E. Arzen. How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime. *IEEE Control Syst. Mag*, 23 (3):16–30, 2003.

11 F. Cremona, M. Morelli, and M. Di Natale. Tres: A modular representation of schedulers, tasks, and messages to control simulations in simulink. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1940–1947. ACM, 2015.

12 Goran Frehse, Arne Hamann, Sophie Quinton, and Matthias Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *Real-Time Systems Symposium (RTSS), 2014 IEEE*, pages 53–62. IEEE, 2014.

13 Katsuhisa Furuta, Masaki Yamakita, and Seiichi Kobayashi. Swing up control of inverted pendulum. In *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*, pages 2193–2198. IEEE, 1991.

14 Madan Gopal. *Digital control engineering*. New Age International, 1988.

15 Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 225–230. IEEE Press, 2011.

16 Zain AH Hammadeh, Sophie Quinton, and Rolf Ernst. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *Proceedings of the 14th International Conference on Embedded Software*, page 10. ACM, 2014.

17 Thomas A Henzinger, Benjamin Horowitz, and Christoph M Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

18 Pavel Krcal and Wang Yi. Decidable and undecidable problems in schedulability analysis using timed automata. In *proceedings of 10th International Conference, TACAS'04*. IEEE, 2004.

19 Steffen Linsenmayer and Frank Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 4765–4770. IEEE, 2017.

20 M. M Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. In *IEEE Transactions on Computers*, 1995.

21 Tsutomu Mita. Optimal digital feedback control systems counting computation time of control laws. In *Decision and Control, 1984. The 23rd IEEE Conference on*, volume 23, pages 520–525. IEEE, 1984.

22 Marieke Posthumus-cloosterman, Cover Design, Gerda Cloosterman, and Paul Verspaget. *Control over communication networks: Modelling, analysis and synthesis*. PhD thesis, University of Eindhoven, 2008.

23 Sophie Quinton, Matthias Hanke, and Rolf Ernst. Formal analysis of sporadic overload in real-time systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 515–520. EDA Consortium, 2012.

24 Parameswaran Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, 1999.

25 Kang G Shin and Hagbae Kim. Derivation and application of hard deadlines for real-time control systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1403–1413, 1992.

26 Damoon Soudbakhsh, Linh TX Phan, Anuradha M Annaswamy, and Oleg Sokolsky. Co-design of arbitrated network control systems with overrun strategies. *IEEE Transactions on Control of Network Systems*, 2016.

27 Damoon Soudbakhsh, Linh TX Phan, Oleg Sokolsky, Insup Lee, and Anuradha Annaswamy. Co-design of control and platform with dropped signals. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*, pages 129–140, 2013.

28 Martin Törngren. Fundamentals of implementing real-time control applications in distributed computer systems. In *Real-Time Systems In Mechatronic Applications*, pages 3–35. Springer, 1998.

29 Eelco P van Horssen, AR Baghban Behrouzian, Dip Goswami, Duarte Antunes, Twan Basten, and WPMH Heemels. Performance analysis and controller improvement for linear systems with (m, k)-firm data losses. In *Control Conference (ECC), 2016 European*, pages 2571–2577. IEEE, 2016.

30 Wenbo Xu, Zain AH Hammadeh, Alexander Kröller, Rolf Ernst, and Sophie Quinton. Improved deadline miss models for real-time systems using typical worst-case analysis. In *Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on*, pages 247–256. IEEE, 2015.

31 Yang Xu, Karl-Erik Årzén, Enrico Bini, and Anton Cervin. Response time driven design of control systems. *IFAC Proceedings Volumes*, 47(3):6098–6104, 2014.

32 Yang Xu, Karl-Erik Årzén, Anton Cervin, Enrico Bini, and Bogdan Tanasa. Exploiting job response-time information in the co-design of real-time control systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on*, pages 247–256, 2015.

33 Tatsuya Yoshimoto and Toshimitsu Ushio. Optimal arbitration of control tasks by job skipping in cyber-physical systems. In *Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, pages 55–64. IEEE Computer Society, 2011.

34 S. Youcheng and M. Di Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. In *EMSOFT Conference, Seoul, Korea*. IEEE, October, 13-17, 2017.

35 Qiang Yu and Xudong Zhao. Stability analysis of discrete-time switched linear systems with unstable subsystems. *Applied Mathematics and Computation*, 273:718–725, 2016.