

Deterministic and Stochastic QoS Provision for Real-Time Control Systems

Daniele Fontanelli, Luigi Palopoli
Dipartimento di Scienza e Ingegneria dell'Informazione
University of Trento
Trento, Italy
{fontanelli,palopoli}@disi.unitn.it

Luca Greco
LSS - Supélec,
3, rue Joliot-Curie,
91192 Gif sur Yvette, France
lgreco@ieee.org

Abstract—In this paper, we propose two adaptive scheduling approaches to support real-time control applications with highly varying computation times. The use of a resource reservation scheduler enables the construction of a dynamic model describing the evolution of the computing delays, which can be incorporated in the system closed loop dynamics. The two approaches differ for the assumptions on the sequence of computation time. In the first approach, we have only an aggregate information (best case and worst case computation time) and design an adaptive scheduler that maintains the delay within the maximum bound compatible with the asymptotic stability of the system. In the second case, we assume a deeper knowledge on the distribution of the computation time and design an adaptive scheduler that ensures second moment stability of the system. The two approaches are evaluated on a case study exposing the different trade-offs between bandwidth and performance.

Keywords-Real-Time Control, Feedback Scheduling, Quality of Service.

I. INTRODUCTION

The development of real-time control applications is traditionally based on three assumptions: 1) the computation time of each application is almost constant, 2) the computation cost of sensing is fixed (and typically negligible), 3) the computing platform hosts a very static set of applications. Under these assumptions, it is possible to set a deadline for each activation of a task and apply the standard results of hard real-time theory (codified in several research paper from the seminal work of Liu and Layland [1] and in complete textbooks [2]) to ensure that such deadlines will be met. A deadline is essentially an upper bound for the delay introduced in the computation. The adoption of a time-triggered model of computation [3] constrains the I/O operations to take place exactly on the deadline. This way the delay is fixed and it can easily be compensated for by using the standard techniques of digital control [4].

The recent industrial developments are steadily changing the landscape and the three assumptions described above are doomed to a gradual obsolescence. First of all, the constant computation time was an assumption deeply grounded in classical digital control, where control algorithm were organised as a sequence of mathematical operations without any significant control flow. Modern control applications are

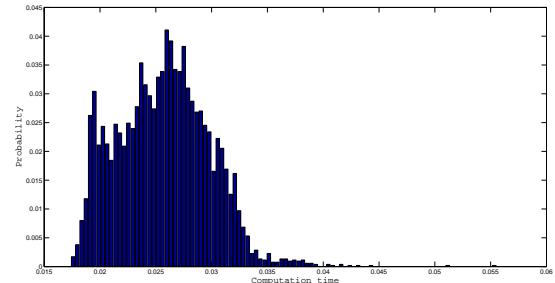


Figure 1: Probability distributions of the computation time of a visual tracking application (courtesy Ruslan Asaula)

often called to a tight interaction with the user and with the environment. Hence, mode changes are very frequent with a remarkable impact on the computation time. Second, the class of sensing devices offered by the modern technology is far richer than it used to be only a few years ago. For instance, visual based control systems are growing to remarkable level of popularity. The acquisition of data from this type of device is known to introduce large variations. As an example, in Figure 1, we report the distribution of the computation time of an application that tracks an object inside an MPEG stream. The computation time oscillates between 15 ms and 55 ms and its distribution can be approximated by a beta distribution [5]. As a third consideration, evident cost reasons and the simplicity of system engineering push toward a massive sharing of computation resources. In this context, hard real-time scheduling theory induces an over-design of the system reducing the number of applications that can be executed simultaneously. This is hardly acceptable since the dogmatic requirement of meeting every single deadline has been proved futile. Several studies [6], [7] have shown that there can be a sensible performance gain in aggressive choices for the scheduling parameters (e.g., the sampling period) even in the face of a moderate number of deadline misses.

This new generation of applications calls for innovative scheduling solutions, combining performance guarantees with a more efficient resource utilisation. Different authors have investigated on *how to make the design robust* against

an irregular timing behaviour of the software implementation, focusing on such effects as packet dropout [8], [9], jitter in computation [10], [11] and time varying delays [12]. Another important thread of work has aimed at modifying the scheduling behaviour in overload conditions to accommodate the needs of the control tasks. Very significant in this class is the work of Marti [13], who proposes to re-modulate the task periods in response to an overload condition. In the same direction goes the work of Lemmon and co-workers [14], who develop a Markov Chain model to describe the possible activations of the control task that can be skipped without compromising stability. This model is built upon to develop a heuristic scheduling strategy.

We argue that a more effective analysis of the impact of delays on the controlled system performance requires a scheduling algorithm that allows modelling the evolution of the delays resulting from a given choice of scheduling parameters. Very interesting in this respect is a class of algorithm known as the resource reservations [15], [16]. Such algorithms have been developed with a primary focus on multimedia applications. Their most relevant features are: 1) the possibility of allocating each task a fraction (bandwidth) of the processor, 2) the possibility of deciding the granularity of the allocation for each task (by a parameter known as server period). The use of this scheduling algorithm enables the construction of a dynamic model describing the temporal evolution of a reservation in which the scheduling parameters play the role of actuators, the computation time plays the role of an external disturbance term. A state variable, the scheduling error, accounts for the delay introduced by the task computation. A feedback loop (adaptive reservation) can be established that uses the measured error and the information on the sequence of computation times to obtain desired properties for the delay [17]. In our previous work [18], we have shown how to construct a jump linear system [19] that relates the closed loop evolution of the system to the one of the scheduling error. This way, it is possible to set up a Markov Decision Process (MDP) to design the adaptive reservation. The property required to the closed loop system was almost sure stability [20] (i.e., the convergence to the equilibrium of almost all the trajectories), which cannot avoid large overshoots in the evolution of the control system. In the paper, the analysis was restricted to uniform distributions for the computation time. In this paper, we push forth this investigation exploring two alternative avenues. First, we show how it is possible to improve the result of our previous work by using the notion of second moment stability (the variance of the state has to be asymptotically bounded), which is close to the idea of minimising a quadratic cost in the state. Moreover, we consider general distributions for the computation time. In our second contribution, we consider the case in which no prior information is available on the distributions. In this case, the adaptive reservation

can be designed considering the worst case evolution of the delays, as it results from all possible variations of the computation time of the task in a set. Both approaches are tested on a common case study, displaying the trade-offs between computation time and performance.

The paper is organised as follows. In Section II, we describe the different elements of the problem, setting the ground for the description of the two approaches. In Section III, we describe the deterministic approach. In Section IV, we describe the stochastic approach putting the stress on the new stability criterion used for this paper. In Section V, we offer our experimental evaluation. Finally, in Section VI we draw our conclusion and announce the future working directions.

II. PROBLEM DESCRIPTION

The problem addressed in this paper can be shortly described as follows. A real-time task τ is used to implement a control loop. The task has a stochastically variable computation time (with known distributions) and shares a CPU with other tasks. Our goal is to identify a scheduling solution that strikes a good compromise between control performance and efficiency. Our strategy is based on an adaptive scheduling mechanism whereby the scheduling parameters are changed for each execution of the task (job) depending on the accumulated delay. In this section, we set the theoretical ground for the two adaptive scheduling approaches showing a formal statement of the problem based on a precise scheduling choice.

A. Platform model

We consider a set of real-time tasks $\{\tau_i\}$ sharing a *processing unit* (CPU). A real-time task τ_i consists of a stream of jobs $J_{i,j}$. Each job $J_{i,j}$ arrives (becomes executable) at time $r_{i,j}$, and finishes at time $f_{i,j}$ after executing for a time $c_{i,j}$. Job $J_{i,j}$ is also characterised by a deadline $d_{i,j}$, that is respected if $f_{i,j} \leq d_{i,j}$, and is missed if $f_{i,j} > d_{i,j}$.

We focus on *periodic* tasks, where arrival times are spaced out by a *task period* T_i , i.e., $r_{i,j+1} = r_{i,j} + T_i$, and each activation time is also the deadline of the previous job $d_{i,j} = r_{i,j} + T_i = r_{i,j+1}$. The ration $\frac{c_{i,j}}{T_j}$ is henceforth referred to as *utilisation*.

1) *The scheduling algorithm:* As a scheduling mechanism, we advocate the use of *resource reservations* [15]. Each task τ_i is associated with a reservation (Q_i, R_i) , meaning that τ_i is allowed to execute for Q_i (*budget*) time units in every interval of length R_i (*reservation period*). The budget Q_i ranges in the interval $\{0, \dots, R_i\}$. The bandwidth allocated to the task is $B_i = Q_i/R_i$ and it can be thought of as the fraction of CPU time allocated to the task. For our purposes, it is convenient to choose a reservation period that is an integer sub-multiple of the task period: $T_i = N_i R_i$, $N_i \in \mathbb{N}$. The particular implementation of the Resource Reservations approach that we consider is the

Constant Bandwidth Server (CBS) [16]. In CBS, reservations are implemented by means of an Earliest Deadline First (EDF) scheduler which schedules tasks $\{\tau_i\}$ based on their *scheduling deadlines* $\{d_i^s\}$, dynamically managed by the CBS algorithm. The algorithm can be shortly described as follows. When a new job $J_{i,j}$ arrives, the server checks whether it can be scheduled using the last assigned deadline, otherwise the request is assigned an initial deadline equal to $r_{i,j} + R_i$. Each time the job executes for Q_i time units (i.e., its budget is depleted), its scheduling deadline is postponed by R_i . This way, the task is prevented from executing for more than Q_i units with the same deadline, resulting in the guarantee that a minimum bandwidth B_{m_i} is reserved to τ_i regardless of the behaviour of the other tasks.

2) *Dynamic model for the scheduler:* As discussed in previous work [17], the advantage of the CBS scheduling algorithm is in the possibility of modelling each reservation as a discrete-event dynamic system whose evolution is observed at the termination of each job $J_{i,j}$. The budget Q_i can be changed dynamically and used as an input to control the temporal evolution of the task. Therefore, by $Q_{i,j}$ we will denote the budget allocated to the j^{th} job of the i^{th} task. The reservation period R_i is held constant and it can be used to decide the granularity of the CPU allocation.

In this setting, we define the *scheduling error* as the difference between the server scheduling deadline $d_{i,j}^s$ (evaluated at the finishing-time of each job) and the deadline of the task:

$$\epsilon_{i,j} \triangleq d_{i,j}^s - d_{i,j} = d_{i,j}^s - r_{i,j} - T_i. \quad (1)$$

The server deadline is aligned with the end of the last server period used for the execution of job $J_{i,j}$. Therefore a positive value for $\epsilon_{i,j}$ means that $J_{i,j}$ received a smaller bandwidth than it needed and it finished in a reservation period beyond the deadline. Conversely, a negative value means that it finished in a reservation period preceding its deadline (the assigned bandwidth was greater than the task needed). An approximation for the dynamic evolution of the scheduling error is given by [17]:

$$\epsilon_{i,j+1} = S(\epsilon_{i,j}) + \left\lceil \frac{c_{i,j+1}}{Q_{i,j+1}} \right\rceil R_i - T_i. \quad (2)$$

where the function $S(\cdot)$ defined as:

$$S(x) \triangleq \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

expresses the scheduling error due to the previous jobs that affect the present job.

3) *Model of Computation for Embedded Controllers:* In the following discussion, we refer to a single control task τ managed by the CBS, hence we drop the i subscript in all the quantities related to τ . As a customary choice in classical digital control, in our setting the embedded controller is implemented as a real-time periodic task activated with a

period $T = NR$. Every job J_j reads an input y_j from the plant and produces a control value u_j according to the following rules: 1) the input and output operations take place only at the start/termination of a server period, 2) although the activation of the jobs is periodic, the input y_j is sampled at the very start of job J_j (so if J_{i-1} is delayed the sampling does not take place upon the activation period), 3) if the job finishes before the deadline (negative scheduling error) the release of the output u_j is deferred to the end of the period, 4) the output release can be delayed by an amount $\Delta_j = D_j R$ (an integer multiple of the server period), but if this delay becomes greater than a maximum value $\bar{\Delta}$ (usually a multiple of the period T) the job J_j is dropped and a new job is activated. The formulation of these rules is partly due to the need for controlling the jitter (if τ finishes all of its jobs before its deadline, the input and output operations are exactly spaced by N server periods). Another reason for rules 3 and 4 is to simplify the formulation of the model. Indeed, with these rules, the integer delay D_j can change for each job but it always ranges in the finite set $\{0, \dots, \frac{\bar{\Delta}}{R}\}$. Recalling equation (2) and considering the drop event, we can provide the following expression for D_j :

$$D_{j+1} \triangleq \max\left\{\frac{\epsilon_{j+1}}{R}, N\right\} - N = S^*(D_j) + \left\lceil \frac{c_{j+1}}{Q_{j+1}} \right\rceil - N, \quad (3)$$

where

$$S^*(x) \triangleq \begin{cases} \frac{\bar{\Delta}}{R} & \text{if } x \geq \frac{\bar{\Delta}}{R} \\ x & \text{if } \frac{\bar{\Delta}}{R} > x > 0 \\ 0 & \text{otherwise} \end{cases}.$$

B. The control problem

In our setting a real-time task is used to implement a controller for a linear time invariant plant. It is convenient to observe the evolution of the plant at the end of each server period. This way, we get the following discrete-time system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}, \quad (4)$$

The task implements a linear controller described by the following state equations:

$$\begin{aligned} z_{j+1} &= A_c z_j + B_c y_j \\ u_j &= C_c z_j + D_c y_j, \end{aligned} \quad (5)$$

where j is the index of the j^{th} job. Such a controller is designed assuming an ideal periodic implementation with period $T = NR$.

Let us assume in the following $\frac{\bar{\Delta}}{R} = N$ and denote by $F_j \in \{0, \dots, 2N\}$ the integer variable describing the number of server periods during which the control value u_{j-1} is held constant. In ideal conditions, where the control task has no delay, the control value is held constant for a whole period: $F_j = N$ (due to the unit delay between the output y_j sampling and the control u_j availability).

If, instead, the $(j - 1)^{th}$ job finished later with a delay $D_{j-1} > 0$, the control u_{j-1} (computed by the $(j - 1)^{th}$ job) is held constant for $N - D_{j-1}$ server periods in the present task period ($N - D_{j-1}$ is the number of server periods remaining till the task period expires). If the j^{th} job has a delay $D_j > 0$, then u_{j-1} is held constant also for D_j server periods in the next task period. Therefore, $F_j = N - D_{j-1} + D_j$.

According to this definition, we can write the controlled system dynamics between the release of the output of job J_{j-1} and of job J_j as follows:

$$x_{j+1} = A^{F_j} x_j + \sum_{t=0}^{F_j-1} A^{F_j-t-1} B u_{j-1} \quad (6)$$

$$y_j = C x_j.$$

In case of maximum delay ($D_j = N$), if the input u_{j-1} is available just at the end of the task period, then both the dynamics of the system (6) and of the controller (5) do not change. If, instead, the delay is greater than N , then a drop event takes place limiting the delay to N . The drop event can be managed either holding the previous control value (drop and hold), or zeroing it (drop and zero). In both cases we consider the controller state z_j to be held ($z_j = z_{j-1}$). Therefore, in the drop case, the dynamics of the controller (5) has to be modified, while the dynamics (6) is unchanged.

Because the delays and hence the number F_j can change on any activation, the system switches between different closed loop dynamics. Therefore, the closed loop system is represented by the following switching system:

$$w_{j+1} = \tilde{A}_{\phi_j} w_j \quad (7)$$

$$y_j = \tilde{C} w_j$$

where $\tilde{C} = [C, 0, 0]$ and the state space $w_j = [x_j^T, z_j^T, \xi_j]^T$ includes the state variable of the plant (x_j), of the controller (z_j) and an additional state variable ξ_j to manage the drop conditions. The expression for the different matrices can be found in our previous work [18]. The piecewise constant function $\phi : \mathbb{Z}_{\geq 0} \rightarrow \{0, \dots, 3N + 1\}$ rules the switching among the different subsystems according to the delay evolution (3) and the drop policy. A rigorous description of such a function will be provided in the next sections.

C. Problem Formulation

The discussion above is well summarised by the scheme in Figure 2. The variable computation time c_j of the control task τ and the scheduling budget Q_j assigned for each job, induce different values of the scheduling error ϵ_j and, hence, of the delay experienced by the j^{th} activation of the feedback loop. The evolution of the delay makes the closed loop to switch between different dynamics (it actually determines the value of ϕ_j).

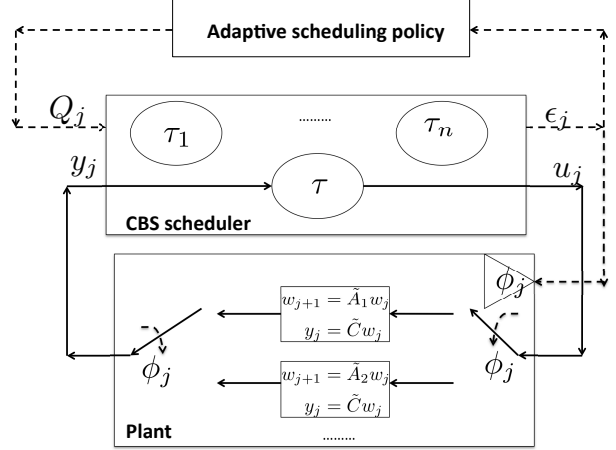


Figure 2: Block scheme of the system considered in the paper

In this setting, the problem we want to solve is to design an *adaptive scheduling policy* that, for each measurement of the scheduling error ϵ_j , chooses the budget Q_j fulfilling the following design goals:

- 1) the closed loop system respects the desired Quality of Control (QoC) specification,
- 2) the computation resources (i.e., the bandwidth) devoted to the control task are minimised.

These two goals are in evident conflict. Indeed, if all available bandwidth is devoted to the task, the delays in the loop are minimised. For instance if the bandwidth exceeds the maximum utilisation of the task, all jobs complete within their deadline and the controlled system always evolves with its “nominal dynamics” ($F_j = N$ for all jobs J_j) where the QoC specifications are respected by design. Conversely, if we choose a small bandwidth for τ , the jobs are often dropped and the system ends up executing with its “open loop dynamics” $F_j = 2N$ most of the times (invalidating the presence of a controller).

The design of a controller that strikes an acceptable trade-off between the two goals requires: 1) a clear identification of the QoC metric of interest for the problem, 2) a criterion that maps the QoC requirements onto an acceptable evolution for the delays, 3) a feedback law for the budget adaptation that makes the evolution of the delays compliant with the requirements with minimal expenditure of bandwidth. As shown next, this problem can be tackled both in a deterministic and in a stochastic framework. The key point difference between the two approaches is on the prior knowledge of the designer on the time-varying sequence c_j of the computation time.

III. DETERMINISTIC APPROACH

The ideal evolution of the scheduling error would be to stay constantly equal to 0. In this case the controlled system would always evolve with its nominal dynamics \tilde{A}_N . If the

sequence c_j would be known a-priori, we could simply set $Q_j = c_j$ and achieve the result (see Equation 2), which obviously corresponds to the minimal allocation of CPU that meets the goal.

In fact we have no such prior knowledge, nor do we assume (in the deterministic approach) to know the probability distributions of c_j . The only information that we have on the sequence of computation times is that it ranges in an interval $c_j \in [\underline{c}, \bar{c}]$ where \underline{c} and \bar{c} are respectively the best case computation time (BCET) and the worst case computation time (WCET).

In this case, it is impossible to force the scheduling error to 0 but we can reduce its evolution in an interval $\mathcal{I} = [-eR, ER]$ with $e, E \in \mathbb{N}$. This set is said a *robustly controlled invariant set* (RCIS), because if at some point the scheduling error is inside the interval then the controller constrains its evolution in the set for all future jobs, countering the fluctuations of the computation time. This is guaranteed by the following:

Theorem 1 ([17]): Consider the system (2) and assume that $c_j \in [\underline{c}, \bar{c}]$ and $Q_j \in [0, \bar{Q}]$. Consider the interval $\mathcal{I} = [-eR, ER]$ and let $\alpha = \frac{\underline{c}}{\bar{c}}$. Then \mathcal{I} is a RCIS if and only if:

$$e + \alpha E > (1 - \alpha)N - 1 \wedge \bar{Q} > R \frac{\bar{c}}{N}. \quad (8)$$

The family of controllers ensuring controlled invariance of \mathcal{I} are those for which:

$$Q_j \in \left[\frac{\bar{c}}{N + E - \frac{S(\epsilon_j)}{R}}, \min \left\{ \bar{Q}, \frac{\underline{c}}{N - 1 - e - \frac{S(\epsilon_j)}{R}} \right\} \right]. \quad (9)$$

The rationale of this result is as follows. Equation (8) poses a boundary for the minimum size of the RCIS \mathcal{I} : the larger the variations of the computation time expressed by the parameter α the larger is the minimum size of \mathcal{I} . Another constraint is on the maximum budget that the controller is required to use, which has to be larger than the maximum utilization of the task. This is easily understood, since if the scheduling error is at the right boundary of the RCIS, the controller has to anticipate the maximum computation request to restrain the next scheduling error inside \mathcal{I} . Equation (9) identifies a family of possible control laws that ensure robust controlled invariance of \mathcal{I} . Taking the left extremal point of the interval corresponds to the most conservative choice in terms of bandwidth, whereas the right extremal point corresponds to the most robust choice (w.r.t. an underestimated WCET). In this paper, for the sake of simplicity, we will choose the left extremal point:

$$Q_j = \min \left\{ \frac{\bar{c}}{N + E - \frac{S(\epsilon_j)}{R}}, \bar{Q} = \frac{N}{T} \right\}. \quad (10)$$

In Figure 3, we report the plot of the control law for different values of the scheduling error and for two different choices of the right boundary of \mathcal{I} . As it is possible to see, allowing

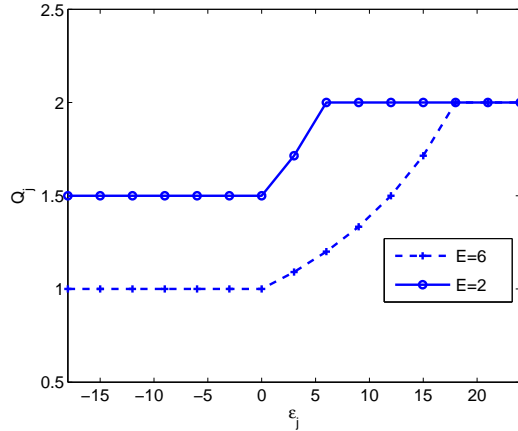


Figure 3: Deterministic feedback law (10) for a task with parameters $T = 18$, $R = 3$, $\bar{c} = 12$, for different values of E .

for a larger delay leads to significant savings in terms of bandwidth (the plot related to $E = 6$ is well below the one related to $E = 2$).

A. QoC metric

The Quality of Control metric that we choose in this context is the most relevant in control theory: asymptotic stability.

Definition 1: Let $x_{k+1} = f(x_k)$ a system such that 0 is an equilibrium point ($f(0) = 0$). The equilibrium is asymptotically stable if for all $\epsilon > 0$ there exists a $\delta > 0$ such that if $\|x_0\| \leq \delta$ then $\|x_k\| \leq \epsilon$ for all k and $\lim_{k \rightarrow \infty} x_k = 0$.

In essence, the definition above corresponds to requiring that 1) the trajectories remain close to the equilibrium when sufficiently small perturbations are applied to the equilibrium (stability), 2) they asymptotically converge to the equilibrium (convergence). In our case the system is the switching system described by Equation (7), where the number of dynamics the system can switch to is related to the maximum delay (the right extreme E of the RCIS). Because we do not assume any knowledge on the distribution of c_j , the evolution of ϵ_j inside the RCIS can be arbitrary, and so the sequence of delays. Thereby, asymptotic stability has to be guaranteed for all possible sequence of switching ϕ_j .

B. Control Design

The robustly controlled invariance of the set \mathcal{I} is obtained applying Equation (10), which determine at each job the reserved bandwidth for the control task. The objectives of the control design in the deterministic approach are: 1) to derive the largest set \mathcal{I} to ensure the most significant savings in terms of bandwidth; 2) to satisfy the QoC requirements, i.e., the closed stability of the system.

The solution here proposed is to iteratively estimate \mathcal{I} using an inner approximation. The starting point of this

procedure is to consider the system with no delay, i.e., $e = E = 0$. This corresponds to have a set of feasible closed loop matrices given by $\mathcal{A} = \{\tilde{A}_0\}$, i.e., the set with only the nominal closed loop system matrix with no delay. At each iteration, \mathcal{I} is enlarged by increasing (decreasing) the value of E (e) by one. As a consequence, the set of closed loop matrices is updated accordingly, i.e., $\mathcal{A} = \{\tilde{A}_{-e}, \dots, \tilde{A}_0, \dots, \tilde{A}_E\}$. To ensure the QoC requirements, a stability test on the set \mathcal{A} is performed at each iteration, applying the following Theorem.

Theorem 2 ([21]): If there exist $p = \#\mathcal{A}$ symmetric matrices G_1, \dots, G_p and P_1, \dots, P_p positive definite, symmetric matrices satisfying

$$\begin{bmatrix} G_i & \tilde{A}_i^T P_j \\ P_j \tilde{A}_i & G_j \end{bmatrix} > 0, \quad \forall (i, j),$$

then the system is asymptotic stability for any given sequence of switchings.

The necessary condition for the application of Theorem 2 is the asymptotic stability of each matrix in the set \mathcal{A} . Unfortunately, as long as the set \mathcal{I} increases, this property is no longer met. However, the execution of two jobs with, for instance, delay of a and b server periods, corresponds to the matrix $\tilde{A}_{b,a} = \tilde{A}_b \tilde{A}_a$, which can be stable even if one of the two matrices is unstable. Theorem 2 can then be applied to the *sequences* of matrices. Notice that the set of the feasible sequences is a subset of all the possible sequences since the delays are bounded by the RCIS property. To summarise, at each iteration, two tests of stability are performed: 1) a first test on sequences of elements of \mathcal{A} , with increasing length; 2) the test of Theorem 2. If both tests are verified, the closed loop system is asymptotically stable for any switching in the set \mathcal{I} . Albeit the described analysis is performed off-line, we point out that the problem is computationally feasible if a maximum matrix sequence length is fixed.

IV. STOCHASTIC APPROACH

In our setting, the sequence of the computation times of the control task τ is aptly described by a stochastic process denoted as $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$, where j is the job index. The discrete random variable c_j takes values in the set $L_c = \{\underline{c}, \dots, \bar{c}\}$, where \bar{c} is the worst case execution time and \underline{c} is the best case execution time. Even though c_j takes value in a discrete set given by the multiples of the system clock, we can reasonably assume that the granularity of the clock is much smaller than the one we use to observe the system evolution (i.e., the server period). Thereby, for all practical purposes we can consider $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$ as a real-valued process, associated with a probability density function $f(c_j)$.

From Equation 3, we can infer that the delay $\{D_j\}_{j \in \mathbb{Z}_{\geq 0}}$ is a stochastic process in its turn whose evolution is conditioned by the process $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$, the budget Q_j and the value of the previous delay D_{j-1} . Our goal is to design a

time invariant reservation policy $\mathcal{P}(Q_j)$, in which the choice of the budget Q_{j+1} for the $j+1$ -th job is *uniquely based on the delay D_j experienced* in the j -th job. The resulting system is a finite-state homogeneous discrete-time Markov Chain (FSH MC). We will denote by σ this FSH MV, $\sigma(j)$ being its state at the j -th job. This MC takes values in the set $L_\sigma = \{0, \dots, N+1\}$. The meaning of the states of this MC is easily described. If $\sigma(j) = q$ with $q \in \{0, \dots, N\}$ then the delay experienced in the j -th job was q server periods: $D_j = q$. If $\sigma(j) = N+1$ then $D_j = N$ and a drop event takes place. The stochastic evolution of $\sigma(j)$ is driven by the transition probability matrix $P = (p_{q,g})_{(N+2) \times (N+2)}$. The element $p_{q,g}$ of the matrix denotes the probability of for the state at step $j+1$ to become g *conditioned* by the state at step j being q : $p_{q,g} \triangleq \Pr\{\sigma(j+1) = g \mid \sigma(j) = q\}$. The evolution of the MC starts from an initial probability measure $\pi_\sigma(0) \in S^{N+1}$, where

$$S^{N+1} \triangleq \left\{ s = [s_1, \dots, s_{N+2}] \in [0, 1]^{N+2} \mid \sum_{i=1}^{N+2} s_i = 1 \right\}$$

is the $(N+1)$ -dimensional canonical stochastic simplex. The evolution of the probability distribution $\pi_\sigma(j)$ of the MC σ is then given by $\pi_\sigma(j+1) = \pi_\sigma(j)P$.

As discussed in our previous work [18], for every $q, g \in L_\sigma$, the probability $p_{q,g}$ is given by $p_{q,g} = \Pr\{c_{j+1} \in V(q, g)\}$, where the sets $V(q, g)$ represent the integration domains of the continuous-time pdf, i.e.

$$p_{q,g} = \int_{V(q,g)} f(x) dx.$$

The integration domain $V(q, g)$ takes the form described below. For $V(q, 0)$, we have $V(q, 0) \triangleq (0, \min(Q\{q\}(N-q), \bar{n})]$ for $0 \leq q \leq N$, $V(q, g) \triangleq (\min(Q\{q\}(g-q+N-1), \bar{c}), \min(Q\{q\}(g-q+N), \bar{c})]$ for $0 \leq q \leq N$ and $1 \leq g \leq N$, $V(q, N+1) \triangleq (\min(Q\{q\}(2N-q), \bar{c}), \bar{c}]$ for $0 \leq q \leq N$. The sets $V(N+1, g)$ for $0 \leq g \leq N+1$ are formally equal to the corresponding sets $V(N, g)$, with the only difference given by the values of the Q if they are state dependent (i.e. $Q\{N+1\} \neq Q\{N\}$).

From the theory of Markov Chains [22], we know that if the FSH MC σ is irreducible and aperiodic, then there exists a unique invariant probability distribution (i.p.d.) $\bar{\pi}_\sigma$ corresponding to the steady-state probability distribution of the MC (i.e. $\lim_{h \rightarrow \infty} \pi_\sigma(h) = \bar{\pi}_\sigma$ for any $\pi_\sigma(0)$). A MC enjoying this property is commonly denoted as FSHIA MC.

It can be easily verified that for σ to be a FSHIA MC it is sufficient to have non-zero entries on the diagonal, on the first subdiagonal and on the first superdiagonal of the transition matrix P . Given the expression of $p_{q,g}$, this

property can be equivalently refo Or equivalently

$$\begin{aligned} 0 < Q\{q\} < \left\lfloor \frac{\bar{c}}{N} \right\rfloor, \quad \forall q \in \{0, \dots, N\} \\ 0 < Q\{N+1\} < \left\lfloor \frac{\bar{c}}{N-1} \right\rfloor. \end{aligned} \quad (11)$$

Example 1: Consider an exponential pdf

$$f(x) = \lambda e^{-\lambda x},$$

with $\lambda \in \mathbb{R}_{>0}$ and continuous support $x \in [0, +\infty) \subset \mathbb{R}$. Since we assume to have a finite WCET \bar{c} and a non-zero BCET \underline{c} , we can define a modified exponential pdf for $c(j)$ as

$$f(c(j)) \triangleq \begin{cases} 0 & \text{for } c(j) < \underline{c} \\ \frac{\lambda e^{-\lambda(c(j)-\underline{c})}}{1 - e^{-\lambda(\bar{c}-\underline{c})}} & \text{for } \underline{c} \leq c(j) \leq \bar{c} \\ 0 & \text{for } c(j) > \bar{c}. \end{cases}$$

Assuming that $\alpha R < \underline{c} \leq (\alpha + 1)R$, with $\alpha \in \{0, \dots, N-1\}$, the transition probability matrix can be obtained noticing that $p_{q,g} = 0$ if $0 < q - g \leq \alpha$. Integrating the pdf over the aforementioned domains $V(q, g)$

and defining $F(a, b) \triangleq \frac{e^{\lambda a}}{1 - e^{-\lambda(\bar{c}-a)}} (e^{-\lambda a} - e^{-\lambda b})$, we have: $p_{q,0} \triangleq F(\underline{c}, Q\{q\}(N-q))$, $p_{q,g} \triangleq F(Q\{q\}(g-q+N-1), Q\{q\}(g-q+N))$ for $1 \leq g \leq N$, $p_{q,N+1} \triangleq F(Q\{q\}(2N-q), \bar{c})$, in the case of $0 \leq q \leq N$ such that $q - g > \alpha$ or $q - g \leq 0$. In the case of $q = N+1$ such that $N - g > \alpha$ we have: $p_{N+1,0} \triangleq 0$, $p_{N+1,g} \triangleq F(Q\{N+1\}(g-1), Q\{N+1\}g)$ for $1 \leq g \leq N$, and $p_{N+1,N+1} \triangleq F(Q\{N+1\}N, \bar{c})$. The mean value of $c(j)$ in the case of exponential pdf is given by $E\{c(j)\} = \frac{e^{\lambda \underline{c}}}{1 - e^{-\lambda(\bar{c}-\underline{c})}} \frac{e^{-\lambda \underline{c}}(1 + \lambda \underline{c}) - e^{-\lambda \bar{c}}(1 + \lambda \bar{c})}{\lambda}$.

Example 2: As suggested in the literature ([5]), in many cases of interest the distribution of $c(j)$ can be approximated by the pdf of the continuous Beta distribution

$$f(c(j)) = \frac{c(j)^{a-1}(1-c(j))^{b-1}}{\int_0^1 x^{a-1}(1-x)^{b-1} dx}$$

taking values in the continuous set $c(j) \in [0, \bar{c}] \subset \mathbb{R}$. The parameters $a, b \in \mathbb{R}_{>0}$ are used to shape of the pdf $f(c(j))$. The mean value of $c(j)$ in the case of the pdf of the Beta distribution is given by $E\{c(j)\} = \frac{a}{a+b}$. The computation of the transition probabilities can be made an in the previous example.

From the stochastic description of the process describing the delays D_j , it is possible to derive a description of the process $\{\phi_j\}_{j \in \mathbb{Z}_{>0}}$ ruling the switchings of the closed loop system (7), which is thus a Stochastic Jump Linear System. The process $\{\phi_j\}_{j \in \mathbb{Z}_{>0}}$ takes values in the set $L_\phi = \{0, \dots, 3N+1\}$. The evolution of the state ϕ_j is related to that of $\sigma(j)$ as follows:

$$1) \quad \phi_j = N - \sigma(j-1) + \sigma(j) \text{ for } \sigma(j-1), \sigma(j) < N+1;$$

- 2) $\phi_j = 2N - \sigma(j-1)$ for $\sigma(j-1) < N+1$ and $\sigma(j) = N+1$;
- 3) $\phi_j = 2N+1 + \sigma(j)$ for $\sigma(j-1) = N+1$ and $\sigma(j) < N+1$;
- 4) $\phi_j = 3N+1$ for $\sigma(j-1), \sigma(j) = N+1$.

Such a process is not directly a FSH MC, but it has been shown in [18] that its distribution is linearly related to the distribution of the FSH MC $\hat{\sigma}(j) \triangleq (\sigma(j), \sigma(j-1))$ taking values in the set $L_{\hat{\sigma}} = L_\sigma \times L_\sigma$ and describing the evolution of two consecutive steps of the MC σ . The evolution of the probability distribution of the MC $\hat{\sigma}(j)$ is given by $\hat{\pi}(j) = \hat{\pi}(j-1)\hat{P}$, with $\hat{\pi} = [\hat{\pi}_{00}, \hat{\pi}_{01}, \dots, \hat{\pi}_{N+1,N}, \hat{\pi}_{N+1,N+1}]$, $\hat{P} = [v_0 p_{00}, v_0 p_{01}, \dots, v_{N+1} p_{N+1,N+1}]$ and v_a a -th column of the matrix $V \in \{0, 1\}^{(N+2)^2 \times (N+2)}$ given by $V = [I_{N+2}, \dots, I_{N+2}]^T$. Therefore, we can write $\pi_{\phi_j} = \hat{\pi}_j W$ for a suitable (stochastic) matrix $W \in \{0, 1\}^{(N+2)^2 \times (3N+2)}$. In [18] we have proved that if σ is a FSHIA MC, then the MC $\hat{\sigma}$ admits a unique i.p.d., which we denote with $\hat{\pi}$. Hence, even the process $\{\phi_j\}_{j \in \mathbb{Z}_{>0}}$ has a steady state distribution given by

$$\bar{\pi}_\phi = \hat{\pi} W. \quad (12)$$

A. QoC Metric

As for the deterministic approach, we consider here as QoC metric a stochastic version of the stability of closed loop system.

Definition 2 (SM stability): [23] The jump linear system

$$x_{t+1} = H_{\gamma(t)} x_t \quad (13)$$

with $x \in \mathbb{R}^M$, $H_i \in \mathbb{R}^{M \times M}$, $i \in \{1, \dots, m\}$ driven by the stochastic process $\{\gamma(t)\}_{t \in \mathbb{Z}_{>0}}$ is said asymptotically second moment stable if, for any $x_0 \in \mathbb{R}^M$ and any initial distribution $\pi_\gamma(0)$ of $\gamma(t)$, the following holds

$$\lim_{t \rightarrow \infty} E \{ \|x_t(x_0)\|^2 \} = 0$$

with $x_t(x_0)$ a sample solution of (13) having initial condition x_0 .

It is worth noting that, even if the SM stability is more restrictive than the Almost Sure stability adopted in [18], it can ensure in general a better behaviour of the closed loop system. If the process $\{\gamma(t)\}$ is a FSH MC, then the previous definition is equivalent to the existence of m positive definite matrices verifying a set of linear matrix inequalities (see for instance [24]). The process $\{\phi_j\}_{j \in \mathbb{Z}_{>0}}$, governing the switching of the system (7), is not a FSH MC. However, in view of the linear relation $\pi_{\phi_j} = \hat{\pi}_j W$ (with $W = (\omega_{lh})_{(N+2)^2 \times (3N+2)}$) between the distribution of ϕ_j and the distribution of the FSH MC $\hat{\sigma}(j)$, it is possible to prove the following sufficient condition [25].

Theorem 3: The jump linear system (13) is SM stable if there exist $(N + 2)^2$ matrices $G_l = G_l^T > 0$ such that

$$\sum_{h=0}^{3N+1} \omega_{lh} \tilde{A}_h^T \hat{G}_l \tilde{A}_h - G_l < 0, \quad \forall l \in L_{\hat{\sigma}} \quad (14)$$

with $\hat{G}_l \triangleq \sum_{i=0}^{(N+2)^2-1} \hat{p}_{li} G_i$ and $\hat{P} = (\hat{p}_{li})_{(N+2)^2 \times (N+2)^2}$ transition probability matrix of the FSH MC $\hat{\sigma}(j)$.

B. Optimal Reservation Policy

We are now in condition to formulate an optimal synthesis problem to find a reservation policy $\mathcal{P}(Q)$ ensuring the SM stability of the closed loop system (7) and minimising a cost index related to the bandwidth B .

A suitable cost index, which provides a stochastic average of the budget provided by the scheduler, is the long-run expected value of the budget itself. Let us define the stochastic process $\{q(j)\}_{j \in \mathbb{Z}_{\geq 0}}$ as the sequence of budgets assigned to the control task τ by the scheduler. Such a process takes values in the set $\{Q\{0\}, \dots, Q\{N+1\}\}$ representing all the possible state dependent budget values. Collecting all these values in a vector $Q = [Q\{0\}, \dots, Q\{N+1\}]^T$ and assuming σ to be a FSHIA MC, the index cost can be written as $\lim_{j \rightarrow \infty} E\{q(j)\} = \bar{\pi}_\sigma Q$. By definition $\bar{\pi}_\sigma$ is a function of the budget Q , hence the previous one is a nonlinear cost index. Even \hat{P} is a function of the budget Q by means of P .

In order to complete the optimal problem formulation we must add some constraints on the values of the budget. Assuming that a fraction of each server period R has to be reserved for the execution of other tasks than τ , then each $Q\{q\}$ must be less than a pre-specified Q_{\max} . Moreover, in order σ to be a FSHIA MC we must add the constraints (11). Therefore, defining the vectors $\underline{Q} = [0, \dots, 0]^T$ and $\bar{Q} = [a, \dots, a, b]^T$, where $a = \max\left(\lfloor \frac{\bar{c}}{N} \rfloor, Q_{\max}\right)$ and $b = \max\left(\lfloor \frac{\bar{c}}{N-1} \rfloor, Q_{\max}\right)$, we can write the constraints on the budget as $\underline{Q} < Q < \bar{Q}$. Summarising we have the following mixed integer nonlinear optimisation program, the *Optimal Reservation Policy - Second Moment Stability* problem (ORP-SMS Problem) in the vector Q and the matrices G_l :

ORP-SMS Problem: $\min_Q \bar{\pi}_\sigma(Q) Q$ s.t.

$$\underline{Q} < Q < \bar{Q} \\ \text{inequalities (14) .}$$

V. SIMULATION RESULTS

The mechanical system chosen for the simulation results is a Furuta pendulum with zero offset [26] (see Figure 4). The continuous time transfer function $G(s)$ between the input torque τ and the output angle α , and the discrete controller $C(z)$ are reported in Table I. The controller $C(z)$ is obtained using a systematic Linear Quadratic Gaussian design assuming a constant unit delay between the controller

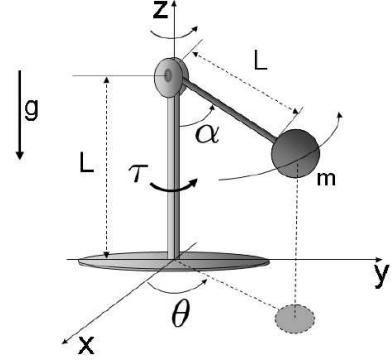


Figure 4: The Furuta pendulum adopted for the simulations.

$$G(s) \left| \begin{array}{l} \frac{7.435}{s(s^2 + 34.63)} \\ C(z) \left| \frac{-35.7517z(z^2 - 1.99z + 0.9919)}{(z - 6 \cdot 10^{-5})(z - 0.8106)(z^2 - 1.85z + 0.8811)} \end{array} \right.$$

Table I: Continuous transfer function of the Furuta pendulum with the relative discrete time controller.

and the system. The system is sampled with a period equal to $T = 25$ ms, which is four times the server period. The server period R is divided into 20 time slices, hence $T = 80$ time slices and the maximum bandwidth value is 20. For the computation time of the control task, we considered three different distributions (reported in Fig. 5): a uniform distribution, an exponential distribution and a beta distribution. The three distributions are truncated in the range $[20, 80]$. Therefore the worst case utilisation of the task is 100%. We have compared four different scheduling solutions. The base line is the classic hard real-time solution: a static budget equal to the worst case utilisation of the task. The second solution is still a static allocation with a choice of the budget corresponding to slightly more the average utilisation. In this case, when a job is delayed for more than one period, it is dropped. The third solution corresponds to the deterministic approach presented in Section III, with $E = 2$ server periods. Finally, the fourth solution has been derived using the stochastic approach.

The simulation results have been obtained using `Matlab`. The simulation procedure comprises four steps: 1) the stochastic process governing the computation time (see Fig. 5) is simulated in a given time horizon; 2) the process realization is used to simulate the real-time system, i.e., to generate the events (i.e., task finished, deadline missed, task delayed) and the relative event time according to each policy; 3) the sequences of events are used to rule the `Simulink` experiments. In all the experiments reported below, the system output angle α was required to track a reference signal given by a square wave with amplitude 1 rad, period 10s and duty-cycle equal to 50%.

A. Qualitative Comparison

In this section, we make a qualitative comparison between the different scheduling solutions referring to the exponential

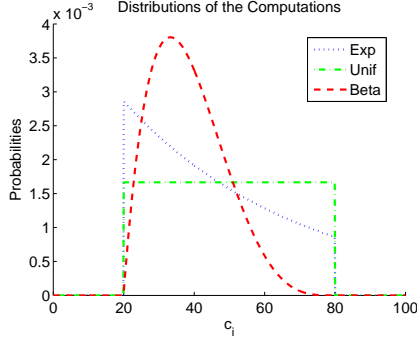


Figure 5: Distributions of the computation time

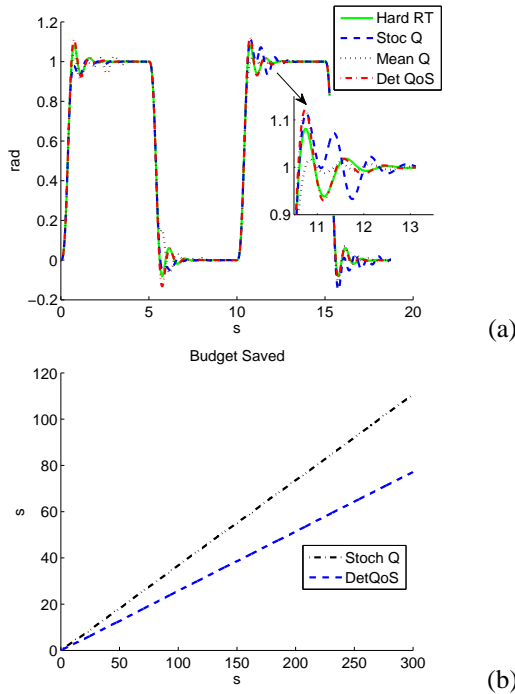


Figure 6: Comparison of a) the output of the system for the four different solutions, b) Accumulated bandwidth saving w.r.t. the hard real time solution of the deterministic and of the stochastic adaptive solution.

distribution in Figure 5. In Figure 6.(a) we report the output response of the system in a time interval for each of the four scheduling solutions. As highlighted by the closeup, the hard real-time solution and the deterministic adaptive scheduler have a very similar performance. The stochastic adaptive scheduler shows a more “nervous” behaviour with a higher number of peaks. The solution with a fixed allocation close to the mean value has smaller peaks but it converges more slowly. The assessment of the two approaches is completed looking at the plot in Figure 6, which shows the integral of the bandwidth saved with respect to the hard real-time allocation. Not surprisingly, the more nervous behaviour of the stochastic approach is paid off by a more substantial saving of computation resources.

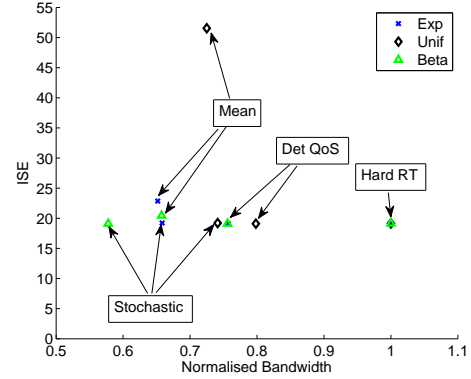


Figure 7: Quantitative comparison between the different solutions

B. Quantitative Comparison

In order to come up with a more accurate comparison of the four solutions, we have introduced a cost function given by the Integral of Squared Error (ISE), defined as $\int_0^\infty \|y(t) - y_{ref}\|^2 dt$. This index quantifies the difference between the output of the system and its reference value. For each experiment, the value of the ISE (truncated to 300s of simulation) was averaged through 20 runs. Likewise, we computed the average of the mean value of the bandwidth used in the different cases. The results are summarised in Figure 7, which reports the bandwidth on the x axis and the mean of the ISE on the y axis. The hard real-time solution is clearly the one with the highest occupation of bandwidth. The ISE performance is independent of the distribution (since the hard real-time solution guarantees that the deadline is invariably met). The deterministic adaptive solution (Det QoS) gains an ISE result very close to the hard real-time allocation, but with a remarkable saving in terms of average bandwidth. This saving is more substantial in the case of the exponential and of the beta distribution. The static allocation of bandwidth close to the average utilisation is clearly more convenient from the point of view of the used resources, but its ISE performance is clearly degraded. The solution based on the stochastic adaptive scheduler is the one that combines a reduced use of resources with an acceptable ISE performance. To make a fair comparison, though, we should observe that the deterministic solution is much easier to implement.

VI. CONCLUSIONS

In this paper, we have considered a real-time task used to implement a digital controller with highly varying computation time. Our purpose is to guarantee an assigned level of Quality of Control (QoC) to the control task with a minimal allocation of computation time. We made the point that an effective way to achieve the result is by an adaptive scheduling policy (adaptive reservation) in which the budget assigned to a CBS server is adapted based on

the experienced delay. The availability of a dynamic model for the delays allows us to write the closed loop dynamics of the controlled system including the dynamics of the scheduler. In this way, we can design the adaptive reservation requiring the desired QoC properties as a constraint for the closed loop evolution of the scheduler. The problem can be tackled in a deterministic framework (if we only have aggregate information on the computation time) or in a stochastic framework (if we have a deeper knowledge on the probability distributions). The two approaches have been confronted on a common case study.

This work has disclosed important opportunities for our future work. The most interesting direction is considering several control loops at once, setting up a scheduling solution that identifies a good trade-off between their performance. Another important direction is studying the robustness of our stochastic approach w.r.t. errors in the distributions. Finally, we plan to investigate stability criteria for the deterministic approach that are more manageable than the one presented in Section III.

REFERENCES

- [1] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, 1973.
- [2] J. Liu, *Real-time systems*. Prentice Hall, 2000.
- [3] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.
- [4] K. Astrom and B. Wittenmark, *Computer-controlled systems: theory and design*. Prentice Hall New York, 1996.
- [5] A. Gupta and S. Nadarajah, *Handbook of beta distribution and its applications*. CRC, 2004.
- [6] L. Palopoli, L. Abeni, G. Buttazzo, F. Conticelli, and M. Di Natale, "Real-time control system analysis: an integrated approach," in *Proc. IEEE Real-Time Systems Symposium*, Orlando, FL, USA, Nov. 2000, pp. 131–140.
- [7] 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 Systems Magazine*, vol. 23, no. 3, pp. 16–30, June 2003.
- [8] J. Nilsson and B. Bernhardsson, "Analysis of real-time control systems with time delays," in *Proc. IEEE Conf. on Decision and Control*, vol. 3, Dec. 1996, pp. 3173–3178.
- [9] Q. Ling and M. Lemmon, "Robust performance of soft real-time networked control systems with data dropouts," in *Proc. IEEE Conf. on Decision and Control*, vol. 2, Dec. 2002, pp. 1225–1230.
- [10] P. Marti, J. Fuertes, G. Fohler, and K. Ramamritham, "Jitter compensation for real-time control systems," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 2001, pp. 39–48.
- [11] B. Lincoln and A. Cervin, "JITTERBUG: a tool for analysis of real-time control performance," in *Proc. IEEE Conf. on Decision and Control*, Dec. 2002, pp. 1319–1324.
- [12] C.-Y. Kao and A. Rantzer, "Stability analysis of systems with uncertain time-varying delays," *Automatica*, vol. 43, no. 6, pp. 959–970, June 2007.
- [13] G. Buttazzo, M. Velasco, and P. Marti, "Quality-of-Control Management in Overloaded Real-Time Systems," *IEEE Trans. on Computers*, vol. 56, no. 2, pp. 253–266, Feb. 2007.
- [14] T. Chantem, X. S. Hu, and M. Lemmon, "Generalized Elastic Scheduling for Real-Time Tasks," *IEEE Trans. on Computers*, vol. 58, no. 4, pp. 480–495, April 2009.
- [15] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia systems," in *Proc. of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.
- [16] L. Abeni and G. Buttazzo, "Integrating Multimedia Applications in Hard Real-Time Systems," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 1998, pp. 4–13.
- [17] L. Palopoli, T. Cucinotta, L. Marzario, and G. Lipari, "Aquosa - adaptive quality of service architecture," *Software: Practice and Experience*, vol. 39, no. 1, pp. 1–31, 2009.
- [18] D. Fontanelli, L. Greco, and L. Palopoli, "Adaptive reservations for feedback control," in *Proc. IEEE Int. Conf. on Decision and Control*, Atlanta, GE, USA, December 2010, to Appear.
- [19] M. Mariton, *Jump linear systems in automatic control*. CRC, 1990.
- [20] P. Bolzern, P. Colaneri, and G. D. Nicolao, "On almost sure stability of discrete-time Markov jump linear systems," in *Proc. 43rd IEEE Conf. On Decision and Control*, vol. 3, 2004, pp. 3204–3208.
- [21] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach," *Automatic Control, IEEE Transactions on*, vol. 47, no. 11, pp. 1883–1887, 2002.
- [22] C. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2008.
- [23] Y. Fang and K. Loparo, "Stabilization of continuous-time jump linear systems," *Automatic Control, IEEE Transactions on*, vol. 47, no. 10, pp. 1590–1603, 2002.
- [24] T. Morozan, "Stabilization of some stochastic discrete-time control systems," *Stoch. Anal. Appl.*, vol. 1, no. 1, pp. 89–116, 1983.
- [25] L. Greco, E. Panteley, and A. Chaillet, "Note on stochastic iss of discrete-time systems," LSS, Tech. Rep., 2010.
- [26] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers. Pt. I. Journal of Systems and Control Engineering*, vol. 206, no. 14, pp. 263–269, 1992.