Overload Management Through Selective Data Dropping

Flavia Felicioni, Ning Jia, Françoise Simonot-Lion, Ye-Qiong Song

Chapter 5

Overload Management Through Selective Data Dropping

5.1. Introduction

During system and network overload periods, excessive delay or even data loss may occur. To maintain the quality of control of an NCS, the implementation system (including both computer and network) overload must be correctly handled. As we can see in the previous chapters, a common approach to dealing with this overload problem is to dynamically change the sampling period of the control loops. In this chapter, as an alternative to the explicit sampling period adjustment, we present an indirect sampling period adjustment approach which is based on selective sampling data dropping according to the (m, k)-firm model [HAM 94]. The interest of this alternative is its easy implementation despite having less adjustment quality, since only the multiples of the basic sampling period can be exploited. Upon overload detection, the basic idea is to selectively drop some samples according to the (m, k)-firm model to avoid long consecutive data drops. The consequence is that the shared network and processor will be less loaded. However, the control stability and performance must still be maintained to an acceptable level. This can be achieved by keeping either the total control tasks on a same processor or the messages sharing a same network bandwidth schedulable under the (m, k)-firm constraint.

In this chapter, we first give a sufficient condition for scheduling a set of control tasks under (m, k)-firm constraint (section 5.2). Then, through several examples, we

Chapter written by Flavia FELICIONI, Ning JIA, Françoise SIMONOT-LION and Ye-Qiong SONG.

present the methods for determining the value of k for a given control loop which will still maintain control loop stability (section 5.3), and the optimal values of m and the control gains which minimize a total cost function either in the presence of the control task configuration changes (section 5.4) or by further coping with the plant state changes (section 5.5). The problem of control loop stability with on-line parameter switching is also discussed in section 5.5.1.

To better illustrate this approach, let us first present the generic system architecture. Then, we identify the problem to solve and provide several notations.

5.1.1. *System architecture*

We consider a global control architecture that integrates a set of plants to control, each one being controlled by a discrete controller. We are interested in the deployment of the control application onto limited resources; for example, all the numerical controllers share the same processors or all the plant states sampled by sensors are transmitted to the controller through the same communication architecture. Moreover, we suppose that according to the global state of the plant, a supervisor chooses the current working mode of the global system. In particular, it can stop the control of a plant, start the control of a plant or modify the control strategy of a plant (control law, sampling period, etc.). The consequence of the transition between working modes is the modification of the set of active tasks/messages that can bring about

– some of them are stopped,

– new tasks are activated or new messages are transmitted,

– the characteristics of tasks may be modified, for example, changing their Worst Case Execution Time (WCET); the characteristics of messages can be modified, for example, their size or the sampling period can be transformed by using a new control strategy.

This means that the scheduling of the messages on a network or the scheduling of the tasks on a processor have to be redefined each time the supervisor modifies the global control mode. The schedulability analysis of a set of tasks or messages subject to hard real-time constraints (all the instances have to meet their deadline) can lead to over provisioning of the resources and this oversizing can be worse in the case of an architecture that implements several working modes as already mentioned before. Moreover, the relationship between the performance of the control and the scheduling of the activities is not well known quantitatively; therefore, the identification of the scheduling parameters relies generally on experiments and/or simulations [SIM 05] that are not exhaustive and so, cannot be generalized. So, a feasible scheduling provided by applying the relaxation of timed constraints as proposed in the classical solution does not lead systematically to the optimal performance of the control application.

Therefore, we propose a new scheduling architecture for handling such configurations as well as an adaptive technique that makes adjustments on-line for, on the one hand, the (m, k)-firm constraints of activities (data transmission and/or task implementing the control law) and, on the other hand, the parameters of the control laws. By doing so, the global performance of the application is fixed at an optimized level and the schedulability under (m, k)-firm constraints is guaranteed.

In the case study that will illustrate the adaptive approach, we consider plants that correspond to harmonic oscillators (as, for example, cart systems, a pendulum or an inverted pendulum). Furthermore, the solution is detailed for processor sharing and therefore, we will deal with the scheduling of tasks instead of messages.

The scheduling architecture is illustrated in figure 5.1. The system to control is composed of a set of plants (Plant 1, Plant 2, ..., Plant n) that are assumed to be independent. Each plant is controlled by a controller (Controller 1, Controller 2, ..., Controller n). In this example, the controllers are deployed as a set of n tasks (τ_1, τ_2, .., τ_n) on the same processor. A task handler is dedicated to the identification of the optimal configuration of (m_i, k_i)-firm constraints for each task τ_i running on the processor. The system is observed by a supervisor, assumed implemented on another processor. The role of the supervisor is to observe the state of the plants and to decide at each instant which plants have to be controlled and which control laws have to be applied. Each time the supervisor modifies the configuration of the system, it sends the task handler a list of active controllers as well as their new characteristics, in

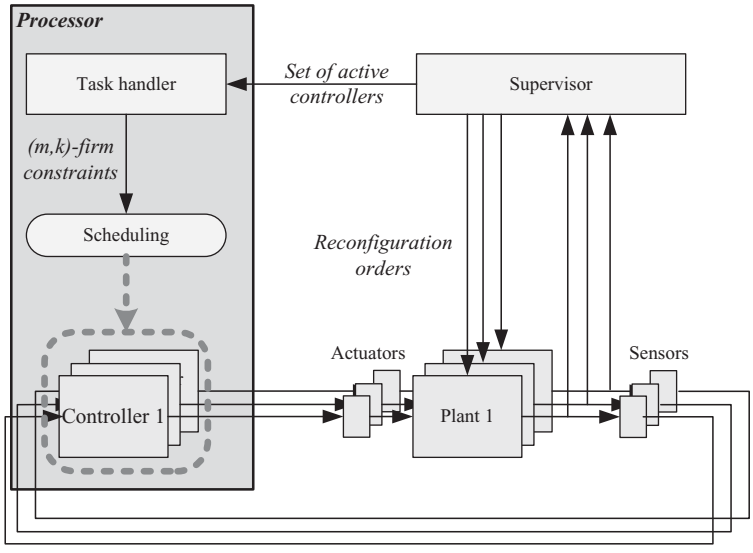

Figure 5.1. *The global scheduling architecture.*

short, the activation period and the worst case execution time of the control law. Of course, the schedulability of this task set must be guaranteed in sense of (m_i, k_i)-firm.

5.1.2. *Problem statement*

The global problem of finding, at each instant an optimized scheduling of control tasks can be divided into several sub-problems.

– Firstly, we have to ensure, at least the stability of each controlled system. This will be achieved by the specification of the highest value of k_i in the constraint (m_i, k_i) of each task τ_i sharing the processor that ensures the stability under a $(1, k_i)$-firm constraint.

– A second problem concerns the optimization issue. Two main considerations have to be taken into account in this context : what is the cost function? what are the criteria? For the first question, we will use the (Linear Quadratic Regulator LQR) cost function that is classically applied in the control community. The determination of the value of the parameter m_i in the (m_i, k_i) constraint of each task τ_i will answer the second question.

Solutions to these different problems will be presented in the remaining part of this chapter. But, first of all, let us give more details in the following section on how to schedule tasks under (m, k)-firm constraint.

5.2. Scheduling under (m, k)-firm constraint

The (m, k)-firm model was first proposed by Hamdaoui and Ramanathan in order to precisely characterize the timing constraints required by certain kinds of applications [HAM 94]. It allows the specification of the guarantee level required by real-time applications tolerating deadline miss of certain instances of tasks or messages. More specifically, in this chapter a task or a message τ_i is said to have (m_i, k_i)-firm deadlines if at least m_i out of any k_i consecutive instances must meet their deadlines and if the schedulability analysis of a set of tasks or messages subject to such (m, k)-firm constraints must furnish a deterministic guarantee. Note that, $m_i = k_i = 1$ means that each instance of τ_i is constrained by a firm deadline. Initially, this technique was introduced to deal with overloading in real-time message stream handling, but it was soon seen that it could be used in real-time control applications [RAM 99]. Two problems must be solved for such constraint specifications :

– scheduling policy : how to schedule a set of tasks or messages subject to (m, k)-firm;

– schedulability analysis : for a given scheduling policy, how to guarantee that the constraints are satisfied.

The main solutions to these problems are briefly described in the following.

Two classes of scheduling policies that were developed to take into account the (m, k)-firm constraints were studied : dynamic scheduling and static scheduling.

In the following, we consider a set of tasks or message streams τ_i, each of them being defined by

– C_i, the largest time needed by the instances of τ_i to complete the task instance execution or the message instance transmission;

– D_i, the relative deadline of τ_i, supposed to be the same for each instance;

– and, when τ_i is periodic or sporadic, its period or the minimum inter-arrival h_i;

– m_i and k_i, the parameters of the (m, k)-firm constraint imposed to τ_i;

– the (m_i, k_i)-pattern, Π_i defined as the sequence $\Pi_i(0)$, $\Pi_i(1)$, .., $\Pi_i(k-1)$ where $\Pi_i(j) = 1$ indicates that the $(j + n.k)^{th}$ $(n \geq 0)$) instance of τ_i has to meet its deadline, $\Pi_i(j) = 0$ indicates that it is not mandatory that the $(j + n.k)^{th}$ instance of τ_i meets its deadline, and consequently, $\sum_{j=n}^{n+k_i-1} \Pi_i(j) = m_i$ for $n > 0$.

5.2.1. *Dynamic scheduling policy under* (m, k)*-firm constraints*

In [HAM 95], the (Distance Based Priority DBP) algorithm is proposed for non-pre-emptive tasks. It implements a dynamic scheduling that is based, at each scheduling point (e.g., arrival instant or departure instant of a task instance), on the distance to a failure state of each task or message. The failure of τ_i is defined as the situation where more than $k_i - m_i$ instances among the k_i last ones have missed their deadline. The highest priority is given to the task that has the lowest distance to its failure. A FIFO strategy is applied when the distance to the failure is the same for several tasks. The schedulability analysis, based on Markov chain, proposed in [HAM 95] provides a probabilistic guarantee. Further studies have been proposed for improving this approach. In particular, in [LI 06], a sufficient condition is specified for a set of periodic or sporadic messages, scheduled using DBP and (Earliest Deadline First EDF).

5.2.2. *Static scheduling policy under* (m, k)*-firm constraints and schedulability issue*

If these dynamic policies provide a good quality of service for a set of streams, they appear quite costly in the context of control applications sharing processors or network resources. To deal with such applications, [RAM 99] considered a system composed of several control applications. The controllers are implemented as pre-emptive tasks running on two processors. When a failure occurs on one processor, all the tasks that were allocated there migrate to the other one, leading to a possible

overload situation. In this case, a static scheduling based on (m, k)-firm constraints is proposed. The mandatory instances of each task are scheduled according to the fixed priority of the task, while the lowest priority is given to the optional instances. Given a couple (m_i, k_i) for each τ_i, the instance number a ($a \geq 0$) is mandatory if $a = \left\lfloor \left\lceil \frac{a.m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor$. This classification has been improved in [QUA 00] by a global approach that determines on-line the mandatory instances of all the tasks τ_i subject to a (m_i, k_i)-firm constraint. As this problem has been proved NP-hard, a heuristic was proposed. Furthermore, [QUA 00] provides, on the one hand, an algorithm for evaluating of an upper bound of each τ_i response time for a "deeply-red" pattern (a "deeply-red" pattern is such that $\Pi_i(a) = 1$ if $0 \leq a < m_i$ and $\Pi_i(a) = 0$ in the other case) and, on the other hand, it demonstrated that if for each task τ_i, the response time is less than the deadline for a (m_i, k_i) "deeply-red" pattern, each τ_i is schedulable for any (m_i, k_i) pattern.

5.2.3. *Static scheduling under* (m, k)*-constraints and mechanical words*

In [JIA 05], a new method using the properties of the mechanical words is developed for the schedulability analysis of a set of non-pre-emptive tasks under (m_i, k_i)-firm constraints. First, it proved that the static patterns defined in the literature can be characterized in the form of the mechanical words leading to a largely simplified schedulability assessment. Then, by identifying the defaults of these patterns, it proposed a new way, based on a cellular line, to determine the (m_i, k_i) pattern of each task τ_i. Through intensive simulations, this technique has been demonstrated to achieve an improvement in the schedulable region. Considering $\alpha = \frac{m_i}{k_i}$, if α is rationale, then the mechanical word whose slope is α is (k_i)-periodic; the classification of the instances as mandatory or optional is therefore based on formula (5.1) :

$$\Pi_i(a) = \lceil (a+1).\alpha \rceil - \lceil a.\alpha \rceil, \forall 0 \leq a < k_i. \tag{5.1}$$

The upper bound of the response time of a non-pre-emptive task τ_i subject to a (m_i, k_i) constraint is obtained, assuming that the sequence is convergent, by the limit of R_i^q when $q \to \infty$. R_i^q is given by the classical recurrent equation (5.2):

$$\begin{aligned} R_i^0 &= C_i \\ R_i^q &= \max_{j>i}(C_j) + \sum_{j<i} \left\lceil \frac{m_j}{k_j} \left\lceil \frac{R_i^{q-1}}{h_j} \right\rceil \right\rceil C_j, \end{aligned} \tag{5.2}$$

where $i > j$ means that the priority of the task τ_i is lower than the priority of task τ_j.

If, for each mandatory instance of each task τ_i, $R_i^q + C_i < D_i$, then the system is proved to be schedulable.

Let us consider now a set of pre-emptive tasks scheduled using a fixed priority strategy defined by the rate monotonic algorithm (the larger the period is, the smaller

the priority is); in [RAM 99] and [JIA 05], it is proved that it is possible to limit the length of the interference interval of a task τ_i due to a task of higher priority τ_j to the basic period h_i (the basic period is the interval between two instances of a task subject to a hard real-time constraint, meaning a (k,k)-firm one). In this case, we first consider a set of intervals associated to the task τ_i and defined in (5.3). h_j is the basic period of a task τ_j. $i > j$ means $h_j < h_i$ (i.e. the priority of the task τ_i is lower than the priority of task τ_j):

$$\begin{aligned} S_{i,j} &= \left\{ \left\lfloor l\frac{k_j}{m_j} \right\rfloor h_j,\ \forall l \epsilon \mathbb{N} \mid \left\lfloor l\frac{k_j}{m_j} \right\rfloor h_j < h_i \right\} \\ S_i &= \bigcup_{j=1}^{i-1} S_{i,j}. \end{aligned} \tag{5.3}$$

Then, the response time of a task may be evaluated by equation (5.4):

$$\begin{aligned} n_j(t) &= \left\lceil \frac{m_j}{k_j} \left\lceil \frac{t}{h_j} \right\rceil \right\rceil \\ W_i(t) &= C_i + \sum_{j=1}^{i-1} n_j(t) C_j. \end{aligned} \tag{5.4}$$

If, for each task τ_i, $\min_{r \epsilon S_i}(\frac{W_i(r)}{r}) \leq 1$, then the (m,k)-firm constraint can be met by all the tasks.

This condition is sufficient in general cases. It is necessary and sufficient if the tasks are synchronous (i.e., the first release time of all the tasks starts at the same time, often at $t = 0$).

5.2.4. *Sufficient condition for schedulability assessment under (m,k)-pattern defined by a mechanical word*

The computation of the sequence W_i for each task τ_i, as it is presented in the recurrence relation (5.4), is non-deterministic and it is difficult, if not impossible, to apply it on-line. Therefore, below, we propose a sufficient condition that ensures the schedulability of a set of tasks under (m,k)-firm constraints. We consider a task set $(\tau_1, \tau_2, ..., \tau_n)$; these tasks are periodic and their periods, named "basic period" in the following are, respectively, $h_1, h_2, ..., h_n$. We assume here that the tasks are scheduled according to a pre-emptive fixed priority policy based on the rate-monotonic algorithm

(the larger the basic period is, the lower the priority is) and that the (m, k)-pattern of each task is defined by a mechanical word through equation (5.1):

The (m_i, k_i)-firm constraint of each task τ_i is satisfied if

$$C_i + \sum_{j=1}^{i-1} n_{i,j} C_j < h_i, \forall 1 \leq i \leq n, \tag{5.5}$$

where $n_{i,j} = \left\lceil \frac{m_j}{k_j} \left\lceil \frac{h_i}{h_j} \right\rceil \right\rceil$.

In fact, $\sum_{j=1}^{i-1} n_{i,j} C_j$ gives the workload generated by all the tasks whose priority is higher than τ_i before instant h_i (remember that we are dealing with the special case where $h_1 < h_2 \cdots < h_n$). So it is clear that the deadline is met for τ_i if the total workload $C_i + \sum_{j=1}^{i-1} n_{i,j} C_j$ can be finished within $[0, h_i]$.

Note that the above schedulability condition is sufficient and necessary when the basic period of a task is a multiple of the basic period of all the tasks with higher priority. In the other cases, it degenerates to a sufficient condition.

5.2.5. *Systematic dropping policy in control applications*

This chapter introduces an approach based on the (m, k)-firm model in order to schedule a set of tasks or messages sharing a common resource. The proposed scheduling principles can be seen as a particular case of the period adjustment technique. More specifically we consider that the period of activities sharing a resource (tasks implementing the control law or samples transmitted by a sensor on a network) can be chosen among a limited number of multiples of the basic sampling period. For example, if the basic period for sampling the plant state is h_i and if the (m_i, k_i)-pattern, Π_i of an activity τ_i (task or message) is $\Pi_i = [10100110]$, then, the actual period will be h_i or $2.h_i$ or $3.h_i$. In fact, in this chapter, we will consider that each optional instance of τ_i is dropped systematically.

As introduced in [SET 96] and improved in [EKE 00], the determination of (m_i, k_i)-pattern, Π_i for each activity τ_i is relevant to an optimization problem for which the cost function is an indicator of the system performance. In the following, we propose a co-design approach dealing with both points : the scheduling parameters and the control parameters. In short, the technique that will be presented in the following sections aims to determine an optimal configuration, or more exactly one that is sub-optimal in practice, of m_i, k_i, Π_i for each activity τ_i and of the gain γ_i of the controller implemented by the task.

5.3. Stability analysis of a multidimensional system

The purpose of this section is to present how to guarantee at least the stability condition of control systems. In fact, we are interested by systems whose structure is presented in figure 5.2.

5.3.1. *Generic model*

The plant is defined as a linear continuous-time system whose evolution is modeled by equation (5.6):

$$dx = Axdt + Budt + dv_c, \tag{5.6}$$

where x is the state vector of the system and u is the output of the controller $x \in \mathbb{R}^p$, $u \in \mathbb{R}^q$. The parameters A and B are two matrices whose dimensions are, respectively, (p, p) and (p, q). v_c is a white noise whose covariance is $R_c dt$. The dimension of the constant matrix R_c is (p, p).

The plant state is sampled periodically; the sampling period is noted h. The jth sampled plant state vector transmitted at times jh is consumed by the linear discrete controller defined in equation (5.7); it is noted x_j in the following:

$$u_j = -Lx_j \quad i = 0, 1, 2.... \tag{5.7}$$

We assume that this controller is implemented as a task. As the purpose is to share the processor among several controller tasks and therefore to decrease the processor bandwidth consumed by this task, in case of an overload situation, several of its instances are rejected according to a (m, k)-firm model and under the constraint that the stability of the system has to be ensured.

Each time an instance is executed, it produces a command u_j. This command is maintained until a new command is produced. The use of the (Zero-Order Holder

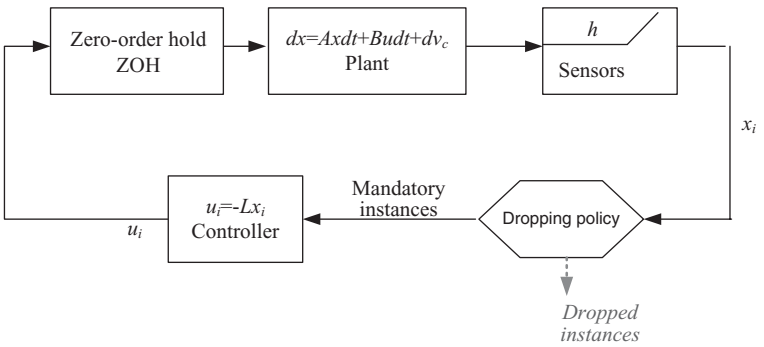

Figure 5.2. *Control system architecture.*

ZOH) and the periodicity of the systematic instances dropping defined by the (m,k)-pattern provide a discrete time behavior of the system modeled by equation (5.8) :

$$x_{j+1} = \Phi_j x_j + \Gamma_j u_j + v_j,\ j = 0, 1, 2..., \tag{5.8}$$

where Φ_j and Γ_j are given by

$$\begin{aligned} \Phi_j &= e^{Af_j} \\ \Gamma_j &= \int_0^{f_j h} e^{As} ds B. \end{aligned}$$

The value of f_j is the distance, in terms of the number of basic sampling periods, between two updates of the command. For example, if, for a given controller task, the (m,k)-pattern is $\Pi = [1001000]$, we get $f_0 = 3$ and $f_1 = 4$.

v_j is a discrete white noise with a zero mean and

$$E v_j v_j^T = R_1 (f_j h) = \int_0^{f_j h} e^{As} R_c e^{A^T s} ds. \tag{5.9}$$

As formerly written, the task instances are periodically rejected according to a given (m,k)-pattern, $f_j + f_{j+1} + f_{j+m-1} = k$ and $f_j = f_{j+m}, \forall j > 0$, the period of the system defined in (5.8) is m, meaning also $\Phi_j = \Phi_{j+m}, \Gamma_j = \Gamma_{j+m}$.

5.3.2. *Example of multidimensional system*

As an example, we propose to study the control of a cart that can move in one direction guided by a rail. The purpose of the control strategy is to drive the cart to a given position. We suppose that the friction parameters are negligible.

An initial reference position is defined and the position of the cart, d is measured with regard to this reference. $\dot{d}$ notes the speed of the cart. Therefore, the state vector of the plant is $x^T = [d, \dot{d}]$. The parameters p and q of the generic model are, respectively, equal to 2 and 1. The simplified model of the plant is given by the following equation (5.10):

$$\ddot{d} = -k_1 \dot{d} + k_2 u. \tag{5.10}$$

An identification of the system furnished the value of the parameters k_1 and k_2 :

$$k_1 = 12.6559 \text{ and } k_2 = 1.9243$$

The continuous model of the system according to the state space representation is given by the generic equation (5.6). In this example, $A = \begin{bmatrix} 0 & 1 \\ 0 & -k_1 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ k_2 \end{bmatrix}$ and $v_c = 0$. The output of the cart is periodically sampled (period h). The closed-loop model is given by equation (5.11) :

$$x_{j+1} = \Phi(h)\, x_j + \Gamma(h)\, u_j, \tag{5.11}$$

with $\Phi(h) = \begin{bmatrix} 1 & \frac{1-e^{-k_1 h}}{k_1} \\ 0 & e^{-k_1 h} \end{bmatrix}$ and $\Gamma(h) = \frac{k_2}{k_1} \begin{bmatrix} h - \frac{1-e^{-k_1 h}}{k_1} \\ 1 - e^{-k_1 h} \end{bmatrix}$.

The command elaborated by the controller is

$$u_j = L(x_{\text{ref}} - x_j), \tag{5.12}$$

where L is the gain $L = \begin{bmatrix} k_c & k_d \end{bmatrix}$ and x_{ref} is the targeted reference defined as $x_{\text{ref}} = \begin{bmatrix} d_{\text{ref}} & 0 \end{bmatrix}^T$ (d_{ref} is the reference of the position on the rail where the cart has to stop).

5.3.2.1. *Sampling period definition*

The basic sampling period h_{basic} is determined according to the empirical "*rule of thumb*" formulated in [ÅST 97] : $0.2 < \omega_0 < 0.6$, where ω_0 is the natural frequency of the system ($\omega_0 = 20$, for the cart considered in this example). The period h_{basic} has to be chosen in $[0.01, 0.03]$. A study of this system shows that its performance in terms of rise time and overshot is optimal for $h_{\text{basic}} = 0.01$s. So we will use this value as the basic sampling period.

5.3.2.2. *Controller parameters*

We note $k_{c,0.01}$ and $k_{d,0.01}$, the parameters of the controller obtained for a basic sampling period $h_{\text{basic}} = 0.01$s. Their values are evaluated by solving the Linear Quadratic Regulator (LQR) problem proposed in [ÅST 97]. The final results are $k_{c,0.01} = 121$ and $k_{d,0.01} = 6.5$.

5.3.3. *Stability condition*

The configuration of a dropping policy based on the (m,k)-firm model needs to identify the values of k, m, and the (m,k)-pattern. The first problem to solve concerns the stability of the system, and the question is which parameter of the constraint is critical for ensuring this property. This identification relies on the intuitive idea that is : if a system is stable for a given $(1,k)$-firm policy, it will be stable for any dropping policy based on a (m,k)-firm. Therefore, we propose to evaluate, for each task, $k_{\max}$, the greatest value of k, so that the system is guaranteed to be stable for each constraint (m,k) with $k \leq k_{\max}$ and $m \leq k$.

The analysis of equation (5.8), shows that on the one hand, a system subject to a (m, k)-firm constraint can be considered as a system with sampling periods varying according to a regular form specified by the (m, k)-pattern and, on the other hand, a system subject to a $(1, k)$-firm constraint is equivalent to a system controlled under a sampling period equal to k times the basic period. Therefore, the determination of $k_{\max}$ is equivalent to the determination of the maximal value of h that ensures the system stability.

In particular, if we study the example proposed in section 5.3.2, let us note $\Psi(h) = \Phi(h) - \Gamma(h) L$, given by

$$\Psi(h) = \begin{bmatrix} 1 - k_c \frac{k_2}{k_1} \left(h - \frac{1-e^{-k_1 h}}{k_1} \right) & \frac{1-e^{-k_1 h}}{k_1} - k_d \frac{k_2}{k_1} \left(h_c - \frac{1-e^{-k_1 h}}{h} \right) \\ k_c \frac{k_2}{k_1} \left(1 - e^{-k_1 h} \right) & e^{-k_1 h} - k_d \frac{k_2}{k_1} \left(1 - e^{-k_1 h} \right) \end{bmatrix}.$$

By applying the Jury criteria [FRE 63], the following three conditions provide a necessary and sufficient condition for the system stability :

$$\begin{aligned} 1.\ a_2 &< 1, \\ 2.\ a_2 &> a_1 - 1, \\ 3.\ a_2 &> -a_1 - 1, \end{aligned}$$

where $a_1 = \Psi_{1,1}\Psi_{2,2} - \Psi_{1,2}\Psi_{2,1}$, $a_2 = -\Psi_{1,1} - \Psi_{2,2}$; $\Psi_{i,j}$ notes the element placed in line i, column j of the matrix Ψ; a_1 and a_2 are expressed according to the controller parameters (k_c, k_d) and the sampling period h. This defines a domain of admissible t-uple (k_c, k_d, h).

In practice, the determination of the greatest admissible value of the sampling period starts by fixing $k_d = k_{d,0.01}$, which is the value obtained for k_d when $h = h_{\text{basic}} = 0.01\text{s}$ (the basic period). Then we study k_c according to h under the following constraints deduced from Jury's conditions.

So, we need to analyze the function that expresses the maximal value of h for each value of k_c, with k_d being fixed at the value $k_{d,0.01}$. In fact, we study the function $k_{cMax}(h)$, which is evaluated by

$$k_{cMax}(h) = \begin{cases} k_{c1}(h) & \text{if } k_{c1}(h) < k_{c\text{Lim}}(h) \\ k_{c2}(h) & \text{if } k_{c1}(h) \geq k_{c\text{Lim}}(h), \end{cases}$$

where

$$\begin{aligned} k_{cLim}\left(h\right) &= \frac{4k_1 e^{k_1 h}}{hk_2\left(e^{k_1 h}-1\right)} \\ k_{c1}\left(h\right) &= \frac{2k_1\left(k_1+k_1 e^{k_1 h}+k_2 k_d-k_2 k_d e^{k_1 h}\right)}{k_2\left(2-2e^{k_1 h}+hk_1+hk_1 e^{k_1 h}\right)} \\ k_{c2}\left(h\right) &= \frac{k_1\left(k_1+k_2 k_d\right)\left(e^{k_1 h}-1\right)}{k_2\left(-1-hk_1+e^{k_1 h}\right)}, \end{aligned}$$

and we obtain the stability region in a space (h, k_c). This region, for the example proposed in 5.3.2, is identified by the gray color in figure 5.3. It represents, for each point k_c, all the admissible values of h. For example, for $k_c = k_{c,0.01}$, the maximal value of h, $h_{\max}$, ensuring the stability is given by the abscissa of the point P and h can be chosen in the interval $[0.01, h_{\max}]$.

As mentioned before, the period adjustment based on a (m,k)-firm model is equivalent to a regular sequence of time intervals between to consecutive samples, specified by the (m,k)-pattern; each time interval is a multiple of the basic period. Therefore, for the cart system, $k_{\max}$ is determined by $\left\lfloor \frac{h_{\max}}{h_{\text{basic}}} \right\rfloor$ and the maximal sampling period, ensuring the stability and corresponding to a $(1, k_{\max})$ constraint, is equal to $k_{\max} h_{\text{basic}}$.

5.4. Optimized control and scheduling co-design

Once the stability of the system ensured, there is a further step needed to deal with the optimization issue. So, to do this, we first define a cost function used for determining an optimal control (section 5.4.1) and, then, we identify the global optimization problem for a set of closed loops where the algorithms implementing the control laws share one processor (section 5.4.2). Finally, the proposed approach is illustrated by a case study in section 5.4.3.

The optimization approach relies on two phases :

– The first is done off-line. For each task, τ_i, a value of k_i ensuring the stability of the system is fixed according to the method described in 5.3.3. For each value of $m_{i,j}$ such that $1 \leq m_{i,j} \leq k_i$, a $(m_{i,j}, k_i)$-pattern $\Pi_{i,j}$ is defined based on the mechanical words technique (see section 5.2, page 192). Then, for each pattern $\Pi_{i,j}$, the cost function of the system is evaluated. Such a function is proposed in section 5.4.1.

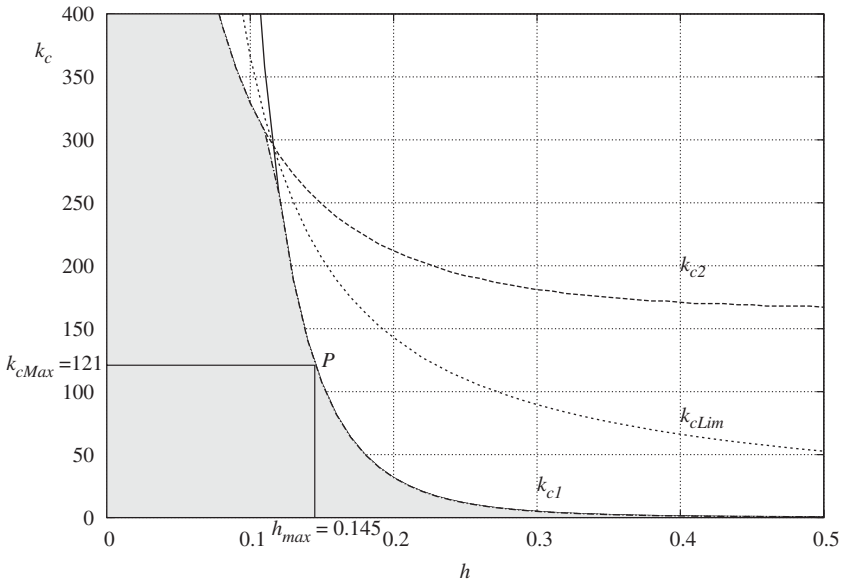

Figure 5.3. *Stability region evaluated on the case study presented in section 5.3.2.* k_{c1}, k_{c2}, k_{cLim} *and* k_{cMax} *are functions of the sampling period* h. *The stability region for the cart system is identified by the gray color.*

– The second phase is done by the task handler as illustrated in figure 5.1. It concerns the global optimization of the system. For each working mode, the task handler has to configure the (m, k)-firm constraints and the scheduling parameters of each task implementing the control law of each system and sharing the same processor.

5.4.1. *Optimal control and individual cost function*

An indicator of the closed-loop performance can be given, among others, by the Least Quadratic (LQ) cost function , which provides a form of "cumulative cost" for an infinite horizon. Applied to a system described by equation (5.6), it is defined by the following formula :

$$J^{\infty} = \lim_{N \to \infty} \frac{1}{N} E(\int_0^N x^T(t) Q x(t) + u^T(t) R u(t) dt),$$

where Q and R are two matrices that respectively weigh the state and the input of the plant.

For a task τ_i that implements the ith controller, given k_i such that the stability is ensured, J_i^{∞} is the cost function to minimize. We consider all the numbers of mandatory instances m_i $(1 \leq m_i \leq k_i)$, assuming that the corresponding patterns

Π_i are defined using the mechanical words approach; for each possible value of m_i and Π_i we determine the optimal control law, in fact the gain of the controller, which minimizes the cost function. As the intervals between two instances are time varying, according to the pattern, the optimal control law is described by a sequence of gain, $L_{i,p}$ for $p = 1, 2, ..., m_i$. For example, let us consider $k_i = 10$, for $m_i = 3$; the (m_i, k_i)-pattern Π_i is equal to $[1001001000]$, and we need to determine three values of the gain : $L_{i,1}$, $L_{i,2}$, $L_{i,3}$ to apply using the control law when producing the command to the actuator, respectively, at the first, second, and third mandatory instances.

We consider the discrete time-varying model of the plant presented in (5.8). We denote by h_i the basic period of the ith plant controller.

The following discrete form of J_i^{∞} is evaluated for each possible value of m_i $(1 \leq m_{i,j} \leq k_i)$ and therefore for the corresponding sequences $(f_{i,0}, f_{i,1}, ..., f_{i,m_{i,j}-1})$:

$$\begin{aligned} J_i(m_{i,j}) &= \frac{1}{m_{i,j}\frac{H_i}{k_i}} \sum_{p=0}^{m_{i,j}\frac{H_i}{k_i h_i}} \left(x_{i,p}^T Q_{i,p}^{'} x_{i,p} + 2 x_{i,p}^T M_{i,p} u_{i,p} + u_{i,p}^T R_{i,p}^{'} u_{i,p} \right) \\ &+ \frac{1}{m_{i,j}\frac{H_i}{k_i}} \left(E\left(x_{m_i \frac{H_i}{k_i h_i}}^T Q_{i0} x_{m_{i,j}\frac{H_i}{k_i h_i}} \right) + \sum_{p=0}^{m_{i,j}\frac{H_i}{k_i h_i}} \overline{J}_{i,p} \right), \quad (5.13) \end{aligned}$$

where H_i is the time horizon of the ith plant, under the condition $\frac{m_{i,j} H_i}{k_i h_i} \epsilon \mathbb{N}^*$, $x_{i,p}$ is the plant state measured at the pth sample and $u_{i,p}$ the corresponding command sent to the actuator, $Q_{i,p}^{'} = \int_0^{f_{i,p} h_i} \Phi_i^T(t) Q \Phi_i(t) dt$, $M_{i,p} = \int_0^{f_{i,p} h_i} \Phi_i^T(t) Q \Gamma_i(t) dt$, $R_{i,p} = \int_0^{f_{i,p} h_i} (\Gamma_i^T(t) Q \Gamma_i(t) + R_i) dt$, $\overline{J}_{i,p} = tr(Q \int_0^{f_{i,p} h_i} R_{ic}(t) dt$ with R_{ic}, the covariance of v_i, and $\Phi_i(t) = e^{A_i t}$ and $\Gamma_i(t) = \int_0^t e^{A_i t} dt B_i$. The optimal control law that minimizes the cost function 5.13 is given by [ÅST 97] as

$$u_{i,p} = -L_{i,p} x_{i,p} \ , \ p = 0, 1, ..., \quad (5.14)$$

where

$$L_{i,p} = \left(\Gamma_{i,p}^T S_{i,p+1} \Gamma_{i,p} + R_{i,p}^{'} \right)^{-1} \left(\Gamma_{i,p}^T S_{i,p+1} \Phi_{i,p} + M_{i,p}^T \right). \quad (5.15)$$

$S_{i,p}$ is obtained from the recurrent equation

$$\begin{aligned} S_{i,m\frac{H_i}{k_i}} &= Q_0^{'} \\ S_{i,l} &= \Phi_{i,l}^T S_{i,l+1} \Phi_{i,l} + Q_{i,l} - \left(\Gamma_{i,l}^T S_{i,l+1} \Phi_{i,l} + M_{i,l}^T \right)^T \\ &\quad \left(\Gamma_{i,l}^T S_{i,l+1} \Gamma_{i,l} + R_{i,l}^{'} \right)^{-1} \left(\Gamma_{i,l}^T S_{i,l+1} \Phi_{i,l} + M_{i,l} \right) \\ &\quad \text{for } 0 \leq l \leq m_{i,j} \frac{H_{i,j}}{k_i h_i}. \quad (5.16) \end{aligned}$$

Taking into account the periodicity of the pattern, $\Phi_{i,p} = \Phi_{i,p+m_{i,j}}$ and $\Gamma_{i,p} = \Gamma_{i,p+m_{i,j}}$. Consequently, the solution of equation (5.16) is also periodic with a period $m_{i,j}$ when calculated on a sufficiently long time horizon [BIT 91] i.e., $S_{i,l} = S_{i,l+m_{i,j}}$. Then, the gain of the controller $L_{i,p}$ is designed using the steady-state solution of the Ricatti equation (5.15), and its solution is also periodic :

$$L_{i,p} = l_{i,p+m_{i,j}}.$$

With this computation, it is possible, for a given $(m_{i,j}, k_i)$-pattern, to determine the best sequence of $L_{i,p}$ that allows us to partially compensate the task instance dropouts.

When time goes to infinity, ($\lim H_i \rightarrow \infty$), the influence from the initial condition decreases and because $S_{i,l} = S_{i,l+m_{i,j}}$, equation (5.13) may be written as

$$J_i^{\infty}(m_{i,j}) = \frac{1}{m_{i,j}h_i}\left(\sum_{p=0}^{m_{i,j}-1} trS_{i,p+1}R_{i,j} + \sum_{p=0}^{m_{i,j}-1} \overline{J}_{i,p}\right). \qquad (5.17)$$

5.4.2. *Global optimization*

Let us now consider the problem introduced at the beginning of section 5.4. As mentioned before, the problem is the global optimization of a set of controllers deployed as a set of tasks sharing the same processor. In the last section (5.4.1), we demonstrated, for a given (m, k)-firm constraint and a given (m, k)-pattern, how to determine the sequence of controller gains that compensate for the task instance dropout between two mandatory instances; this set of gains is identified in order to minimize a cost function that represents a cumulative cost and is derived from the LQ function. Using this evaluation, realized off-line, each task τ_i that may be activated in one of the possible working modes, is characterized by several attributes :

– its basic period h_i and its priority P_i,

– the execution time of the task C_i,

– the parameter k_i that ensures the stability of the system under a period $k_i h_i$,

– the number n_i of values $m_{i,j}$ such that a systematic dropping of the task instances can be done following the $(m_{i,j}, k_i)$-firm constraint :

– for each value $m_{i,j}$:

- the value of $m_{i,j}$,

- the $(m_{i,j}, k_i)$ pattern, $\Pi_{i,j}$; we recall that it is defined thanks to the mechanical words,

- the list of gains to apply at each instance of the task : $L_{i,1}$, $L_{i,2}$, ...,$L_{i,m_{i,j}}$
- $J_{i,j}$ the value of the cost function obtained by using these gains repeatedly on each interval $[pk_i, (p+1)\,k_i]$ for $p \geq 0$.

This information is used on-line for the resolution of the global optimization problem. As soon as a new working mode is defined, the task handler knows which plant needs to be controlled and by which controller; at this point, it has to choose, for each active controller, and therefore for each corresponding task τ_i , what the value of the parameter m_i is that has to be applied in order to minimize a global cost function. We denote the number of tasks activated in one working mode by n.

Let us consider a set of tasks τ_i , $1 \leq i \leq n$, described by the parameters given above; the global optimization problem consists in determining $(s_{i,1}, s_{i,2}, .., s_{i,n_i})$ for each task τ_i that minimizes

$$\sum_{i=1}^{n} \sum_{j=1}^{n_i} (-s_{i,j} J_{i,j}), \tag{5.18}$$

with $s_{i,j} \epsilon \{0,1\}$, $\sum_{j=1}^{n_i} s_{i,j} = 1$, $i = 1, .., n$.

Under the schedulability constraint that has to be met by all tasks τ_i, $i = 1, .., n$, this condition is equivalent to the one formulated in equation (5.5) :

$$C_i + \sum_{j=1}^{i-1} \left\lceil \frac{\sum_{p=1}^{n_j} s_{j,p} m_{j,p}}{k_j} \left\lceil \frac{h_i}{h_j} \right\rceil \right\rceil C_j < h_i. \tag{5.19}$$

This optimization problem can be seen as a Multiple-Choice, Multiple Dimension Knapsack (MMKP) problem [MAR 90] that has been proved to be NP-hard. Therefore, solving this problem on-line requires developing an heuristic algorithm ensuring that it can provide a tight sub-optimal solution. In the following, we apply a slightly modified version of the computationally cheaper algorithm HEU proposed in [KHA 02]. For our optimization problem, the proposed algorithm is

5.4.3. *Case study*

In this section, we apply the method presented above to a case study. Let us consider four cart systems, cart_1, cart_2, cart_3, and cart_4, similar to the one presented in section 5.3.2. The control of these cart systems can be active or not depending on the

Algorithm 5.1: Modified computationally cheaper heuristic

1) to find a feasible solution first, that is to say, select $m_{i,j}$ for each τ_i while satisfying the constraints given in (5.19);
for this purpose, the algorithm HEU is modified by always setting the value of $m_{i,j}$ of each task τ_i to be equal to 1 (if the solution is infeasible in this case, no other solution will be feasible);

2) and then to iteratively improve the solution by replacing, for eachτ_i, the current value of $m_{i,j}$ by another value corresponding to a better performance while keeping the constraints (5.19) satisfied;
if no such solution can be found, the algorithm tries an iterative improvement of the solution which;
a) first replaces $m_{i,j}$ for one task τ_i, which is not schedulable with the current value of $m_{i,j}$;
b) and then replace the value of $m_{i',j}$ for all tasks $\tau_{i'}$ ($i' \neq i$) by a value providing a worse performance;
the original algorithm HEU tries to find, after the first step, a better solution requiring less resource consumption which, however, does not exist in our model, therefore, this property also help us to delete an unprofitable search procedure in HEU;

3) The iteration finishes when no other feasible solution can be found;

working mode chosen by the supervisor (see figure 5.1). The tasks implementing each controller share the same processor.

A Matlab/Simulink model of the system is specified. The scheduling policy is implemented using the toolbox TrueTime [CER 03]. The system is then analyzed by tracing an indicator of the control performance during the simulation of this model running on a given scenario. This indicator is given, for each controlled cart system, by function (5.20), which is evaluated at each simulation step:

$$J_i(t) = \int_0^t \left(x_i^T(s) Q_i x_i(s) + u_i^T(s) R_i u_i(s) \right) ds, \tag{5.20}$$

where $Q_i = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $R_i = 0.00006$ for each cart_i, $i = 1, 2, 3, 4$.

Furthermore, we can observe the state of each task instance during the simulation (running, pre-empted, not activated).

5.4.3.1. *Plants and controllers*

The generic continuous model of the cart system, cart_i for $1 \leq i \leq 4$ (see equation 5.6) is given by

$$dx = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-11.4662}{M_i} \end{bmatrix} xdt + \begin{bmatrix} 0 \\ \frac{1.7434}{M_i} \end{bmatrix} udt + dv_c, \tag{5.21}$$

where M_i is the mass of cart_i:

$$M_1 = 1.5,\ M_2 = 1.2,\ M_3 = 0.9,\ M_4 = 0.6.$$

The controller of each cart_i is noted as $controller_i$, the task which implements the $controller_i$ is τ_i, and its basic period is h_i :

$$h_1 = 0.007,\ h_2 = 0.0085,\ h_3 = 0.01,\ h_4 = 0.0115.$$

5.4.3.2. *Scheduling parameters*

The tasks τ_i are scheduled according to a fixed pre-emptive priority policy under an implicit deadline constraint for their mandatory instances ($D_i = h_i$). The priorities of the tasks are defined thanks to their basic period (the larger the period is, the lower the priority is). So, in any working mode, there is the following relation between the task priorities :

$$\mathrm{priority}(\tau_1) > \mathrm{priority}(\tau_2) > \mathrm{priority}(\tau_3) > \mathrm{priority}(\tau_4)$$

Let us now fix the parameters of the constraint (m, k)-firm for each task. The value of the parameter k_i is defined in order to ensure the stability of the system under a period equal to $k_i h_i$. In this case study, we identified the following value of k_i :

$$k_1 = 5,\ k_2 = 8,\ k_3 = 10,\ k_4 = 1.$$

We consider, in the considered experiments, that the four tasks have the same execution time :

$$C_1 = C_2 = C_3 = C_4 = 3\ \mathrm{ms}$$

and that the execution time of the task handler is $C_{th} = 2,5\ \mathrm{ms}$. Furthermore, its priority is higher in the system.

5.4.3.3. *Optimal controller*

The controller of cart_i is defined by $u_i = L_i x_i$, where the gain L_i is evaluated for each interval between two consecutive mandatory instances according to the (m_i, k_i)-firm strategy used for this plant. As detailed in section 5.4.2, the value of the gain is calculated in order to optimize the control performance during this interval. The cost function, to minimize, in this case study is the discrete form of

$$J_i = \lim_{N \to \infty} \int_0^N \left(x^T(t) Q_i x(t) + u^T(t) R_i u(t) \right) dt, \tag{5.22}$$

where $Q_i = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ and $R_i = 0.00006$ for each cart_i, $i = 1, 2, 3, 4$.

5.4.3.4. *Simulation scenario*

The system is observed along a scenario that introduces three working modes and two types of working mode switchings:

– at time $t = 0s$, the first two cart systems (cart_1 and cart_2) are to be controlled; therefore, only two tasks are running : τ_1 and τ_2 implementing, respectively, the $\mathrm{Controller}_1$ and $\mathrm{Controller}_2$; the set of tasks to be scheduled is $\{\tau_1, \tau_2\}$;

– at time $t = 1s$, the fourth cart system ($cart_4$) has to be controlled; so, τ_4 implementing the $Controller_4$ is activated; the set of tasks to be scheduled is $\{\tau_1, \tau_2, \tau_4\}$;

– at time $t = 2s$, the third cart system ($cart_3$) has to be controlled; so, τ_3 implementing the $Controller_3$is activated; the set of tasks to be scheduled is $\{\tau_1, \tau_2, \tau_3, \tau_4\}$.

Two pre-emptive-fixed priority scheduling policies were modeled:

– *Hard real-time constraints*. We consider that all the task instances are mandatory; in this case, the gain of each controller is constant and determined in order to optimize the cost function (5.22) for the basic sampling period of each system.

– *Adaptive system*. In this case, we implement a systematic dropout of the non-mandatory instances according to the (m, k)-firm constraints specified for each task; the gain of the controller is adapted to each inter-sample interval length in order to optimize the performance of each system; the value of k is constant for a given system in each working mode, while the value of m is evaluated for each system at the beginning of each working mode in order to optimize the global cost function proposed in (5.18) under the schedulability condition (5.19).

5.4.3.5. *Simulation results for hard real-time constraints*

The control performance of each system as well as the evolution of the task state are illustrated in figures 5.4–5.6.

– In the interval $[0, 1[$, two tasks are periodically activated: τ_1, with the period $h_1 = 0.007$ s, and τ_2, with the period $h_2 = 0.0085$ s; all the instances of both tasks meet their deadline.

– At time $t_1 = 1$ s, the supervisor decides to include the control of $cart_4$ leading to the activation of the task τ_4; its period is $h_4 = 0.0115$ s; so, it will be activated successively at 1 s, 1.0015 s, 1.013 s, 1.0245 s, etc.; we can observe in figure 5.4 that no instance of τ_4 meets its deadline; the starting time of several instances is delayed and the completion of all the instances are after their deadline; as the priorities of τ_1 and τ_2 are higher than that of τ_4, the activation of τ_4 has no impact on the scheduling of the two other tasks that meet always their deadlines;

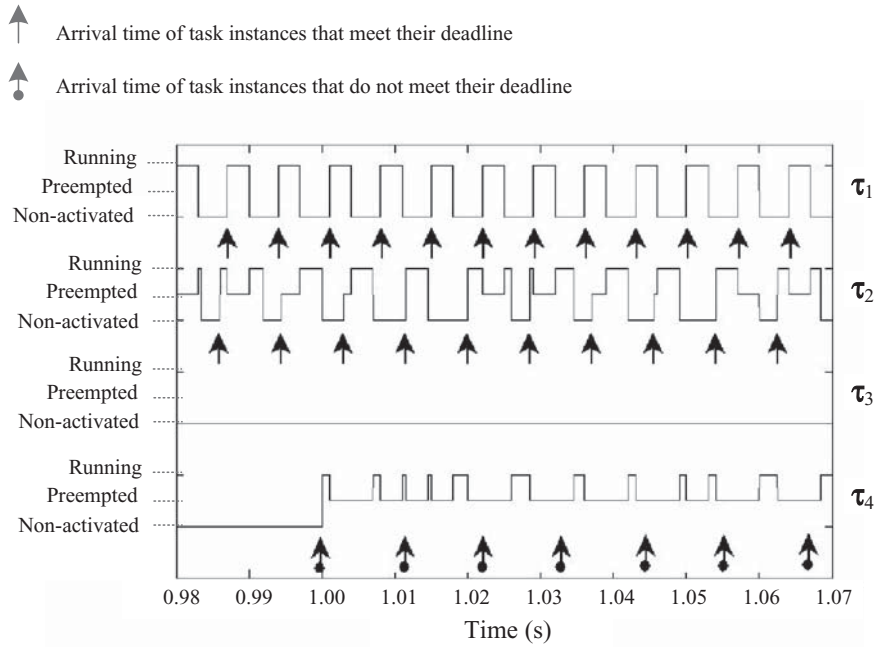

Figure 5.4. *Task evolution under hard real-time constraints strategy. The detailed time interval is* $[0.98, 1.07]$ *and it includes a working mode switch at time* $t_1 = 1\,\mathrm{s}$*: before* t_1*, only two tasks are activated (*τ_1 *and* τ_2*); at time* t_1*, the plant* $cart_4$ *has to be controlled leading to the activation of the task* τ_4*.*

– At time $t_2 = 2$ s, the supervisor includes the control of $cart_3$ leading to the activation of the task τ_3 at time 2 s, 2.0115 s, 2.023 s, etc.; we can observe in figure 5.5, that no instance of this new task meets its deadline; furthermore, the interference of the three tasks of higher priority, τ_1, τ_2, and τ_3, makes the processor unavailable for the task τ_4, leading all the instances of this task to fail to run.

An analysis of figure 5.6 shows how the performance of the system varies, as evaluated by function (5.20). We can note that for $cart_4$, the performance is acceptable between $t = 1$ s, up to $t = 2$ s, which is before the activation of task τ_3. Then the performance of $cart_4$ diverges. This is due to the interference of tasks τ_1, τ_2, and τ_3, whose priorities are higher than that of τ_4. As mentioned before, this task will never run. Finally, the performance of $cart_3$ is always acceptable despite the instances of the corresponding control task τ_3 never meeting their deadline.

5.4.3.6. *Simulation results for* (m, k)*-firm constraints*

The control performance of each system as well as the evolution of the task state are illustrated in the figures 5.7–5.9.

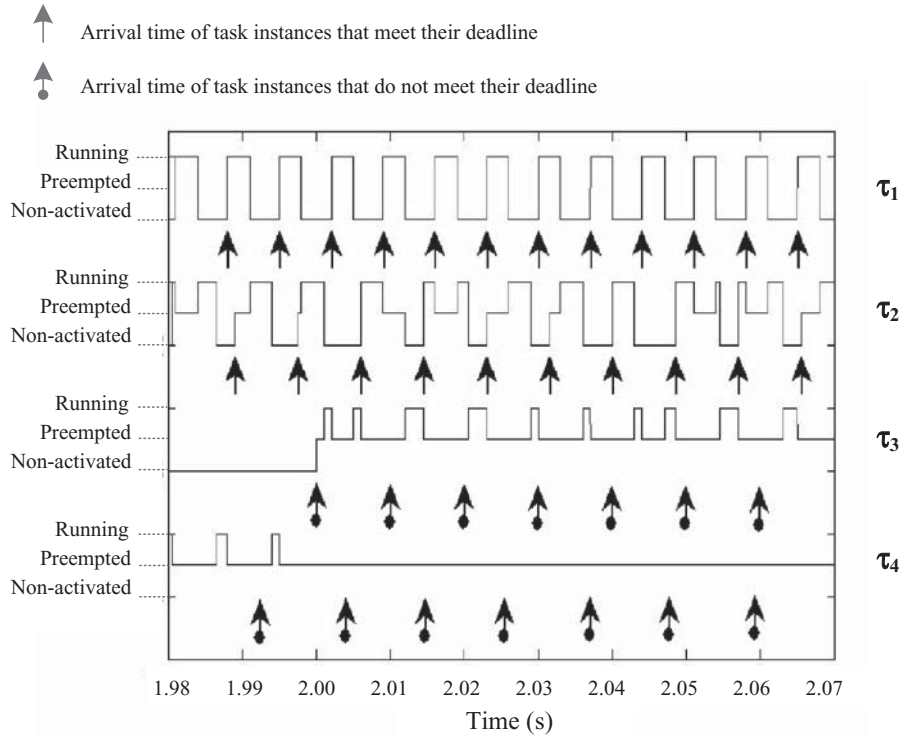

Figure 5.5. *Task evolution under hard real-time constraints strategy. The detailed time interval is* $[1.98, 2.07]$ *and includes a working mode switch at time* $t_2 = 2\,\mathrm{s}$*: before* t_2*, three tasks are activated (*τ_1*,* τ_2 *and* τ_4*); at time* t_2*,* $cart_3$ *has to be controlled leading to the activation of the task* τ_3*.*

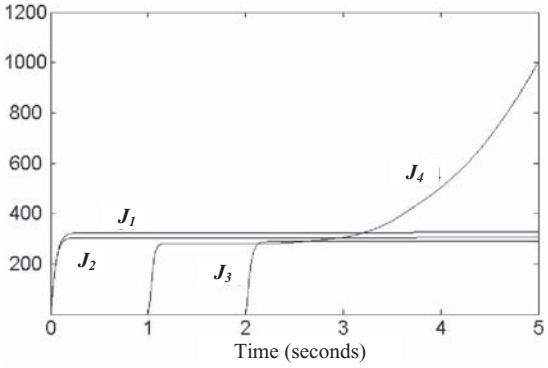

Figure 5.6. *Control performance under hard real time constraints strategy.*

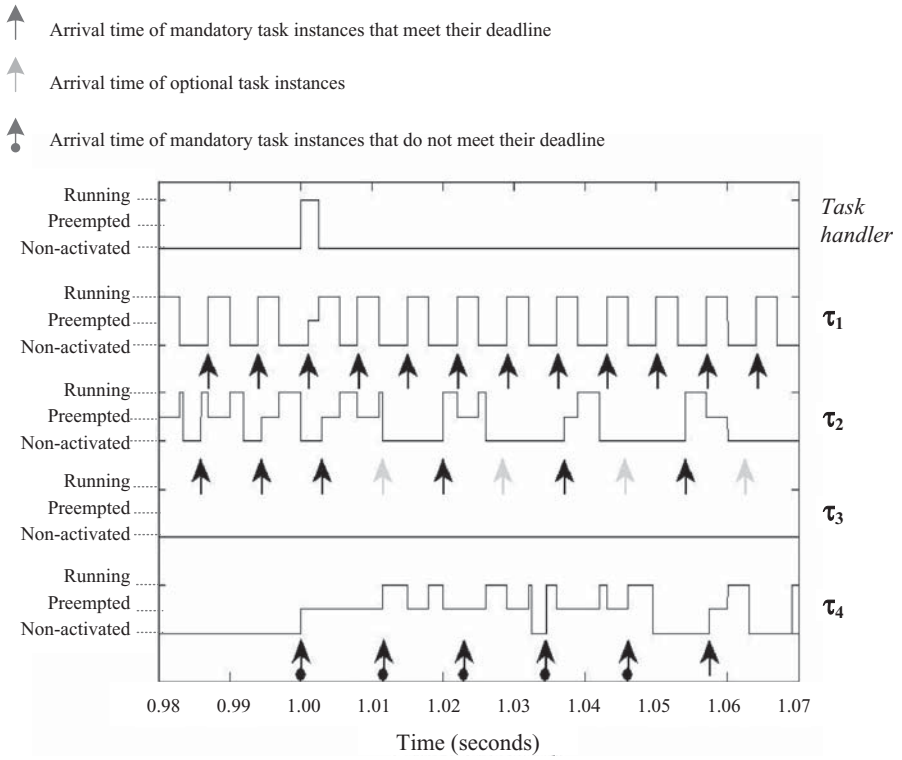

Figure 5.7. *Task evolution under (m, k)-firm strategy. The detailed time interval is $[0.98, 1.07]$ and includes a working mode switch at time $t_1 = 1$ s: before t_1, only two tasks are activated (τ_1 and τ_2); at time t_1, $cart_4$ has to be controlled, leading to the activation of task τ_4.*

– In the interval $[0, 1[$, two tasks are activated periodically: τ_1, with the period $h_1 = 0.007$ s, and τ_2, with the period $h_2 = 0.0085$; these tasks are schedulable under hard real-time constraints as seen in section 5.4.3.5; moreover, in this working mode, the global cost function is optimized for $m_1 = k_1$ and $m_2 = k_2$, so all the instances of τ_1 and τ_2 are mandatory.

– At time $t_1 = 1$ s, the supervisor decides to include the control of $cart_4$ leading to the activation of task τ_4; in figure 5.7, at this time, the task handler is activated; as its priority is higher in the system, its completion is at time $t = t_1 + C_{th} = 1.0.0025$ s; at time t_1, the optimization of the global cost function (5.18), realized by the task handler, furnishes the value of $m_1 = 5$ and $m_2 = 4$ providing the rules for the dropping policy: $(5, 5)$-firm for τ_1 and $(4, 8)$-firm for τ_2 under the (m_2, k_2)-pattern $\Pi_2 = [10101010]$; as $k_4 = 1$, all the instances of τ_4 are mandatory; figure 5.7 shows that all the mandatory instances of τ_1 and τ_2 meet their deadline, while no instance of

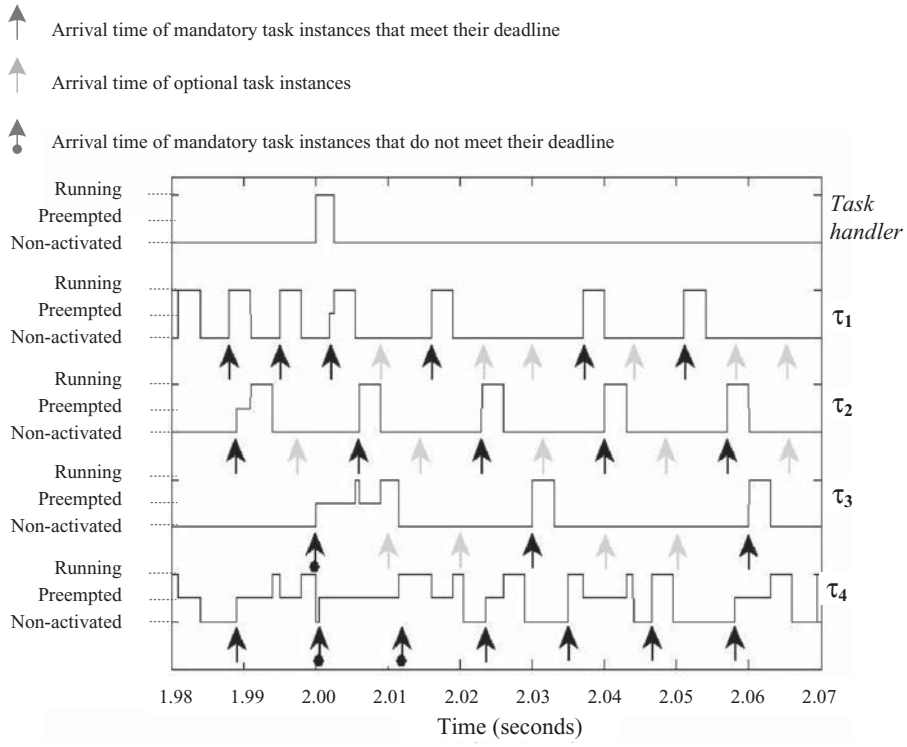

Figure 5.8. *Task evolution under (m, k)-firm strategy. The detailed time interval is $[1.98, 2.07]$ and includes a working mode switch at time $t_2 = 2$ s: before t_2, three tasks are activated (τ_1, τ_2 and τ_4); at time t_2, $cart_3$ has to be controlled, leading to the activation of the task τ_3.*

τ_4 does; nevertheless, all its instances run to completion, under a delayed starting time and delayed completion time.

– At time $t_2 = 2$ s, the supervisor includes the control of $cart_3$, so a new working mode that integrates the former tasks τ_1, τ_2, and τ_4 and the new task τ_3 is started; the task handler has to redefine the optimal configuration of the task activation rules; in this case, function (5.18) is minimized for the following (m_i, k_i)-firm constraints:

- $(m_1, k_1) = (2, 5)$ and $\Pi_1 = [10100]$,
- $(m_2, k_2) = (4, 8)$ and $\Pi_1 = [10101010]$,
- $(m_3, k_3) = (3, 10)$ and $\Pi_1 = [1001001000]$,
- $(m_4, k_4) = (1, 1)$.

We can observe, in figure 5.8, that the two first instances of τ_4 and the first instance of τ_3 , activated at the beginning of this new working mode, do not meet their deadline because of a transient overload due to the execution of the task handler; after that point, all the mandatory instances of the four tasks meet their deadline.

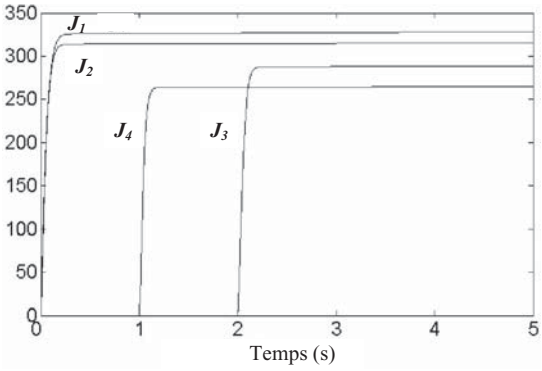

Figure 5.9. *Control performance under* (m, k)*-firm strategy.*

Figure 5.9 illustrates the evolution of the cost function, provided by formula (5.20). For each cart system, the performance becomes quickly stable and remains constant in each working mode.

The optimization algorithm of the global cost function (5.18) under the schedulability constraints (5.19) for a set of control tasks sharing the same processor allows for applying a best dropout policy of several well-identified instances for each tasks; the consequence is a wider availability of the processor and the possibility for each task to run its mandatory instances to completion before the deadline (the relative deadline is the basic period of each task). Furthermore, the controller itself is optimized and an adaptive gain suited to the (m_i, k_i)-pattern is defined.

5.5. Plant-state-triggered control and scheduling adaptation and optimization

Let us consider in this section, more precisely, the different situations of the states of the plants that may be controlled :

– the plant is not controlled; this situation occurs when the plant does not exist for the overall system (plant controlled only during certain time interval, plant deactivated because its output is out of a given domain);

– it is controlled and its state is a steady state;

– it is controlled, but when this control is just activated, its state is a transient state.

We have to take into account the last two cases and, in particular, we must adapt the cost functions in order to deal with both situations. For this purpose, we propose taking into consideration, depending on the situation of the plant, an infinite-horizon or a finite-horizon cost function in order to find the optimal configuration of each m_i and of each controller gain L_i.

5.5.1. *Closed-loop stability of switching systems*

A change in the value of m_i, for a given k_i, produces a sampling period variation, and then we consider a Discrete Time Switched System (DTSS) description. To adapt the control law parameters to this variation, we use the design process presented in section 5.4.1. But, as was shown in [SCH 02], controllers designed with optimal-LQ techniques may suffer from instability under certain switching sequences, i.e., when the sampling period changes. Because of this undesirable result, [SCH 02] adopts a linear matrix inequalities (LMI) framework to design stable optimal controllers. We propose using a LMI framework to find a common quadratic Lyapunov function (CQLF); then the asymptotic stability is guaranteed for any (m_i, k_i)-firm sequence for each plant, proving the stability of the control.

Firstly, for a fixed control task τ_i, k_i, we consider each possible value of the number of mandatory instances, m_i. For each of these, we compute the corresponding m_i controller parameters $\mathcal{L}_{i,m_i} = \left\{L^0_{i,m_i}, L^1_{i,m_i}, ..., L^{m_i-1}_{i,m_i}\right\}$ by using equation (5.15).

Secondly, we consider, for the control task τ_i, the set of open-loop discrete time models (5.6), $\Theta_i = \left\{(\Phi_{i,1}), (\Phi_{i,2}), ..., (\Phi_{i,k_i})\right\}$ and evaluate the k_i periods, taking into account the possible interruptions in a planned sequence at any time.

By using the elements in both sets $\mathcal{L}_{i,m_i}$ and Θ_i, we can establish a new set of $m_i.k_i$ closed-loop models (5.7) without noise, $A^l_{i,p} = \Phi_{i,l} + \Gamma_{i,l} L^p_{m_i}$, where l varies between 1 and k_i $((\Phi_{i,l}, \Gamma_{i,l})\, \epsilon \Theta_i)$ and p between 0 and $m_i - 1$ $((L^p_i \epsilon \mathcal{L}_{i,m_i}))$.

In order to prove the stability of the DTTS [LIB 99], for k_i and each possible value of m_i, we need to find a CQLF for the set of matrices $A^i_{n,p}$, where $n = 1, ..., k_i$ and $p = 0, ..., m_{i-1}$. Then, we can define a set of inequalities :

$$\left(A^i_{n,p}\right)^T P A^i_{n,p} - P < 0,\ \forall n = 1, ..., k_i,\ \forall p = 0, ..., m_{i-1}. \tag{5.23}$$

Note that the identification of a CQLF is a sufficient condition. In order to solve the problem and find the matrix $P = P^T > 0$, we use the LMI toolbox from the Matlab tool.

5.5.2. *On-line plant state detection*

As mentioned in the previous section, three plant states are identified : non-activated plant, steady plant state (or near), and transient plant state. Reaching or leaving the first situation for a plant modifies the value of the number of control tasks n. The dead-band approach presented in [OTA 02] is used to distinguish the steady and the transient states of a plant. Each controlled plant has a state, which asymptotically tracks the reference r, and is supervised by the supervision task. Let y_i be the state of

the plant i. y_i can be a subset of the plant state vector x_i defined in section 5.3.1. For instance, taking the previously studied cart system with $x = [position\ speed]'$, if we want the position to follow the reference position r, the variable $position$ is then the parameter of the interest, and we can define $y = [1\ 0] * x$. The following condition on two successive samples (n^{Te} and $(n+1)^{NH}$ ones) is set up :

$$|y_i\left(h_i + Nd_i\right) - y_i\left(nh_i\right)| < \min\left\{\delta\left|y\left(nh_i\right)\right|, \mathrm{th}\right\},$$

where th is a threshold to prevent false identifications due to noises, δ is a parameter to adjust for each plant, h_i is the detection period implemented by the supervision task for plant i. If the condition is verified, then the plant is considered to be in steady state; otherwise, it is in a transient state. The advantage of this plant state detection mechanism is that it depends on the actual evolution and it detects, in the same way, reference changes or/and non-modeled perturbations.

5.5.3. *Global optimization of control tasks taking into account the plant state*

As in section 5.4, we deal with several working modes where, in each mode, a number of control tasks have to share one processor. In this section, we take into account that the working modes are identified, on the one hand, by a number of plants to be controlled, meaning, a number of given control tasks and, on the other hand, by the situation in which the plant is found. This situation corresponds to the two above-mentioned cases: the plant is in a transient state (due to a new reference or noises) or is in steady state. Intuitively, the constraints, in terms of performances, that have to be applied to the control are not the same for these two situations. Therefore, we have to consider a more complex cost function than the one presented in section 5.4.

We suppose that the value k_i of each τ_i has been carefully chosen and is constant during the execution of application. The value of m_i has to be chosen in $[1..k_i]$ on-line by the task handler. We note n the number of tasks activated in one working mode. For each control task τ_i, each possible value of m_i is associated with two values G_{i,m_i} and G'_{i,m_i} corresponding to the control performance, respectively, in a transient situation and in a steady one. Supposing that a lower value of G_{i,m_i} or G'_{i,m_i} represents a better control performance, in the event of a situation change of the plant states, the aim of the task handler is to find, for each τ_i, a value so that the sum of G_{i,m_i} or G'_{i,m_i} (according to the situation into which the plants fall) for $j\epsilon\left[1\ldots k_i\right]$ and $i\epsilon\left[1\ldots n\right]$ is minimized the subject to the task schedulability condition 5.19. This is formally formulated as the following optimization problem.

Considering in a set of tasks τ_i , $1 \leq i \leq n$, described by h_i, their basic period (considered as their relative deadline, i.e., $D_i = h_i$), C_i, their worst case execution time, k_i the parameter of their (m,k)-constraint, n_i, the number, and the list of possible values for the parameter m in the (m,k) constraint; the global optimization

problem consists in determining $(s_{i,1}, s_{i,2}, \ldots, s_{i,n_i})$ for each task τ_i that minimizes

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i}(s_{i,,j}G_{i,,j}I + s_{i,,j}G'_{i,,j}F),$$

with $s_{i,,j}\epsilon\{0,1\}$, $\sum_{j=1}^{n_i} s_{i,,j} = 1$, $i = 1, \ldots, n$ and such that condition (5.19) is verified.

F and I indicate the current situation: in transient state, $F = 0$ and $I = 1$ while in the steady state $F = 1$ and $I = 0$.

The values of $G_{i,,j}$ and $G'_{i,,j}$ have to reflect the performance offered by each solution. We identified two ways for defining the performance indicator.

– In the first case, $G_{i,,j}$ is defined on a finite horizon by formula (5.13) while $G'_{i,,j}$ is evaluated on an infinite horizon using (5.17):

$$\begin{aligned} G_{i,,j} &= J_i(m_{i,j}) \\ G'_{i,,j} &= J_i^{\infty}(m_{i,j}). \end{aligned} \tag{5.24}$$

We have to note that the optimization problem is to minimize the overall cost of the application. However, the sub-systems with lower costs may suffer from greater control performance degradation due to a low value of m_i. That is to say, the task handler maintains the value of each m_i as high as possible for the sub-systems with greater costs by reducing the value of m_i for the sub-systems with lower costs.

– The second solution is concerned by a cost that represents performance degradation :

$$\begin{aligned} G_{i,j} &= \frac{J_i(m_{i,j}) - J_i(k_i)}{J_i(k_i)} \\ G'_{i,,j} &= \frac{J_i^{\infty}(m_{i,j}) - J_i^{\infty}(k_i)}{J_i^{\infty}(k_i)}, \end{aligned} \tag{5.25}$$

where $J_i(x)$ and $J_i^{\infty}(x)$ are defined, respectively, by functions (5.13) and (5.17). In this case, the control performance criteria avoid the problem given for the first solution presented. The control performance degradation of each sub-system is treated equally. On the other hand, the overall cost of application may not be optimal. So, the choice of control performance representation should be identified according to the application requirements.

In both cases, the time horizon H_i for the finite-horizon cost function is an important design parameter, which directly affects the overall control performance, and needs to

	A_i	B_i	$R_{i,c}$ (incremental covariance of $v_{i,c}$)
Plant_1	$\begin{bmatrix} 0 & 1 \\ -18 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 516 \end{bmatrix}$	$\begin{bmatrix} 0.0025 & -0.005 \\ -0.005 & 0.001 \end{bmatrix}$
Plant_2	$\begin{bmatrix} 0 & 1 \\ 0 & -12.6558 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1.9243 \end{bmatrix}$	$\begin{bmatrix} 0.005625 & -0.075 \\ -0.075 & 1 \end{bmatrix}$
Plant_3	$\begin{bmatrix} 0 & 1 \\ -22.206 & -0.9424 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0.48036 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 22.2066 \end{bmatrix}$
Plant_4	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -14 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 28 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

Table 5.1. *Parameters of the plants whose controllers share the same processor and scheduled according to their transient and steady state.*

be carefully chosen. We propose opting for H_i as the settling time (defined approximately as three times the rise time). Moreover, the off-line computation of (5.25) is done by considering the typical values of the reference values and by neglecting noise.

As in the solution presented in section 5.4.2, the optimization problem results from the MMKP. To solve this problem on-line, we also propose using the heuristic algorithm (HEU) presented in [KHA 02].

5.5.4. *Case study*

In this section, we illustrate the scheduling approach presented above by studying the control of four plants. $Plant_1$ (resp. $Plant_2$, $Plant_3$, $Plant_4$) corresponds to a harmonic oscillator system, (resp. to a cart system, a pendulum and an inverted pendulum). The controller of $Plant_i$ is denoted by $Controller_i$. Each plant is modeled by the differential equation (5.6) whose parameters are given in the table 5.1. The controller of the $Plant_i$ is defined by equation (5.7).

The rise time specifications of each plant are, respectively, 0.2, 0.2, 0.3, and 0.5. Then, the basic sampling periods are related to rise time specifications, i.e., $h_1 = 0.02s$ for $Plant_1$, $h_2 = 0.02s$ for $Plant_2$, $h_3 = 0.03s$ for $Plant_3$, and $h_4 = 0.05s$ for $Plant_4$.

We suppose that the first state variable in vector x of each plant is the variable supervised by the supervision component. In other words, the controller tries to keep it tracking the plant state reference asymptotically. The step response target for the cart

	Q_i	R_i
$Controller_1$	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 25 \end{bmatrix}$	200
$Controller_2$	$\begin{bmatrix} 1.25 & 0 \\ 0 & 0.0085 \end{bmatrix}$	0.0001
$Controller_3$	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$	0.00001
$Controller_4$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	0.001

Table 5.2. *Weights used for each performance evaluation of each controller sharing the same processor and scheduled according to the transient or steady state of the controlled plant.*

($Plant_2$) is an over-damped response, while for the others they are under-damped, being the damping coefficient greater than 0.6 ($\text{overshoot} < 10\%$). The gain of the controllers is computed for each possible value of $m_{i,j}$, according to the approach proposed in 5.4.1. The design weights, which allow the satisfaction of the above-mentioned rise time and overshoot, are presented in table 5.2.

Using the LMI control toolbox of Matlab/Simulink tool, the set of inequalities (for the overall set of discrete plants and controllers parameters), has a QCLF, guarantying stability. To allow for fast changes between different $(m_{i,j}, k_i)$-firm constraints at an $m_{i,j}$ adjustment, the controller parameters are calculated off-line and stored in a table.

As applied in section 5.4, to the control task τ_i which implements the controller $Controller_i$ is assigned the rate-monotonic priority, the task with the largest period has the lowest priority; its execution has no influence on the other tasks. Therefore, no task instance classification will be applied to τ_4, or in other words, it is executed under (k, k)-firm constraint. Using the approach proposed in section 5.3.3, the value of k_i is set to, respectively, $k_1 = 6$, $k_2 = 5$, $k_3 = 5$ and $k_4 = 1$. The value of m_i for the plant τ_i may vary within $[1..k_i]$. The worst case execution time of each task is $C_1 = C_2 = C_3 = C_4 = 9$ ms.

The optimal costs as well as the sequence of corresponding gains, associated with the different possible (m_i, k_i)-firm constraints of each task τ_i are evaluated off-line and stored in order to be used on-line by the task handler. Then the Matlab/Simulink model is simulated. The deployment characteristics of the global system, for short, the specific scheduling policy, are done by using TrueTime toolbox [CER 03].

5.5.4.1. *Simulation scenario*

The evolution of the scenario is illustrated in figure 5.10.

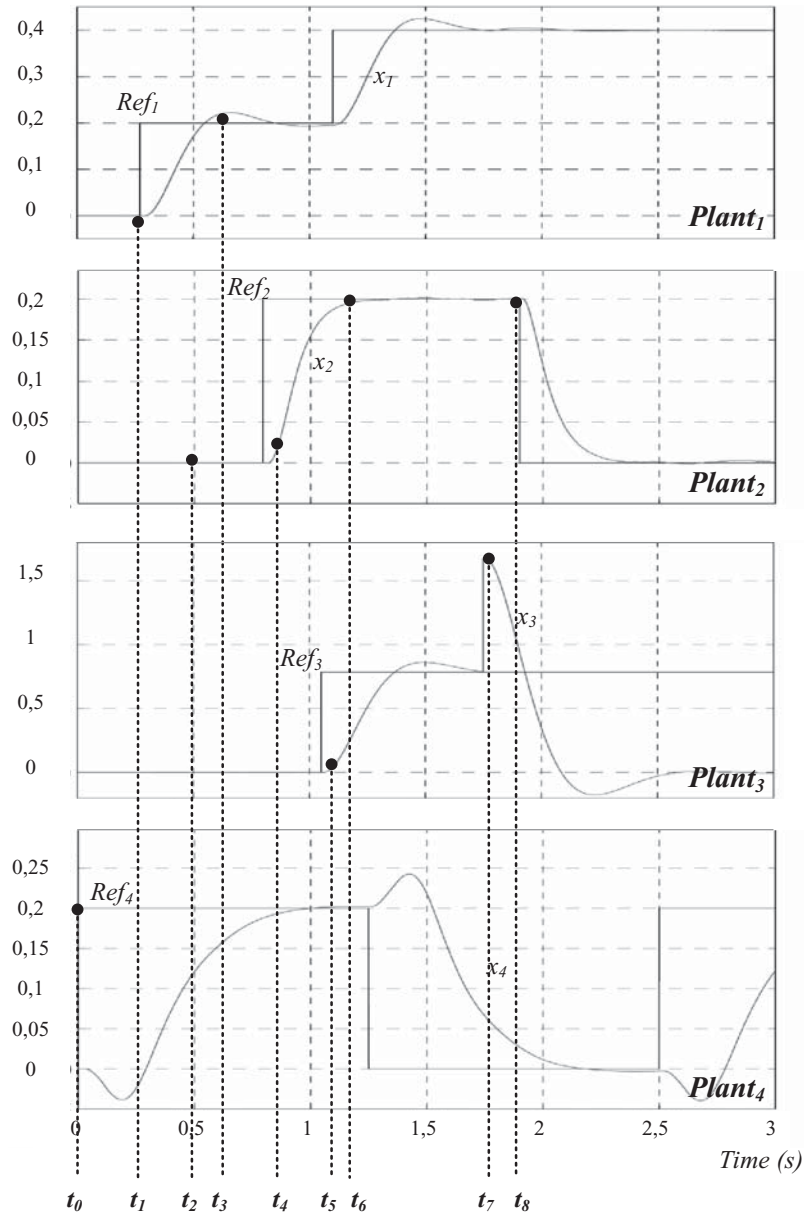

Figure 5.10. *Activation, reference (Ref_i for $Plant_i$) and output (x_i for $Plant_i$) evolution of the four plants whose controllers share the same processor.*

	$Plant_1$ ($k_1 = 6$)		$Plant_2$ ($k_2 = 5$)		$Plant_3$ ($k_3 = 5$)		$Plant_4$ ($k_4 = 1$)	
	State	m_1	State	m_2	State	m_3	State	m_4
$t < t_0$	steady	6	-	-	steady	5	-	-
$[t_0 t_1[$	steady	4	-	-	steady	5	steady	1
$[t_1 t_2[$	transient	6	-	-	steady	2	transient	1
$[t_2 t_3[$	transient	3	steady	1	steady	2	transient	1
$[t_3 t_4[$	steady	2	steady	1	steady	5	transient	1
$[t_4 t_5[$	steady	2	transient	2	steady	2	transient	1
$[t_5 t_6[$	steady	2	transient	1	transient	5	steady	1
$[t_6 t_7[$	steady	2	steady	1	transient	5	steady	1
$[t_7 t_8[$	steady	6	steady	5	-	-	transient	1
$t \geq t_8$	steady	2	-	-	-	-	transient	1

Table 5.3. *Values of m_i as they are selected by the task handler at each working mode switch; these values are evaluated for each task activated.*

It shows, for each $Plant_i$, if it is controlled or not and, in the first case, the reference that is applied and the output of the plant, respectively, noted in figure Ref_i and x_i for $Plant_i$. Table 5.3 provides, at each significant instant, the value of m_i that is selected by the task handler for each current task. An instant is significant if it leads to a switch between two working modes: a new plant has to be controlled, one plant is no longer activated, one plant goes from steady state to transient state, or a plant reaches its steady state.

These instants are identified on-line by the supervisor.

The sequence of significant instants in the proposed scenario is described below.

– Just before the observation of the system, we suppose that only $Plant_1$ and $Plant_3$ are controlled, while the other plants are not activated and not controlled. These two plants are in a steady state.
In this case, the system is schedulable under (k_1, k_1) and (k_3, k_3) constraints for τ_1 and τ_3.

– At time t_0, $Plant_4$ is activated and, therefore, the set of tasks to schedule is $\{\tau_1, \tau_3, \tau_4\}$; no reference is applied to this new plant. The three plants are in a steady state.
The task handler has to find the optimal configuration of m_i for this set of tasks and to deduce the corresponding sequence of values for L_i, the gain of each controller. For this purpose, it uses the infinite-horizon cost for each plant. The result is $m_1 = 4$ and $m_3 = 5$ (we have to note that m_4 has to be always equal to 1.)

– A transient state is detected by the supervisor at time t_1 for $Plant_1$ (occurrence of a new reference) and $Plant_4$
The task handler looks for an optimal configuration of the (m_i, k_i) constraints by

taking into account the finite-horizon costs for $Plant_1$ and $Plant_4$ and the infinite-horizon cost for $Plant_3$. This results in $m_1 = 6$ and $m_3 = 2$.

– $Plant_2$ is activated at time t_2 and the supervisor identifies it is in a steady state while $Plant_1$ and $Plant_4$ are still in a transient state.
The values of m_i defined by the task handler at this time are, respectively, $m_1 = 3$, $m_2 = 1$ and $m_3 = 2$.

– The supervisor detects a steady state for the $Plant_1$ at time t_3.
The new values evaluated for m_i are $m_1 = 2$, $m_2 = 1$ and $m_3 = 5$.

– At time t_4, the values of m_i have to be modified as the supervisor detects a transient state for $Plant_2$.
Therefore, $m_1 = 2$, $m_2 = 1$ and $m_3 = 5$.

– $Plant_3$ enters in a transient state at t_5.
The adjustment of the values of m_i results in $m_1 = 2$, $m_2 = 1$ and $m_3 = 5$.

– $Plant_2$ goes from transient to steady state at t_6.
This leads the task handler to readjust the m_i parameters, using the infinite-horizon cost for τ_1 and τ_2 and to the finite-horizon cost for τ_3 : $m_1 = 2$, $m_2 = 1$, $m_3 = 5$.

– At t_7, an inadmissible perturbation enters in $Plant_3$ whose output reaches $\frac{\pi}{2}$ and consequently, this plant is deactivated, thus reducing the number of tasks to 3. At the same time, $Plant_4$ enters a transient state.
The task handler can therefore increase the values of m_i for the tasks τ_1 and τ_2 : $m_1 = 6$ and $m_2 = 1$.

– At the end of the scenario, at time t_8, a transient state is detected by the supervisor for $Plant_2$.
The last values for the m_i parameters that are chosen by the task handler are $m_1 = 2$ and $m_2 = 5$. We recall that during the simulation, parameter m_4 is always equal to 1 (hard real-time constraint).

5.5.4.2. *Observed performance*

In order to analyze the control of the performance degradation due to the (m, k)-firm policy, we evaluate the LQ cost, given by formula (5.20), for each system during the simulation time. Let us note J_i^{adaptive} as this value. On the other hand, during the same simulation time and under the same simulation setup, we calculated the nominal performance of each plant provided by the same formula when each system is controlled by a control task on a separate processor; in this case, the constraint applied is a (k, k) one and the sampling period as well as the activation period of the control task are equal to the basic period of each controller. We note J_i^{nominal}, the value obtained for $Plant_i$. Table 5.4 presents an example of the performance degradation.

Of course, these results depend on the simulation setup, and they are only exposed here to show that using the proposed technique, the degradation of the performance should be kept as small as possible in each situation subject to the task schedulability.

	$J_i^{adaptive}$	$J_i^{nominal}$	$\frac{\lvert J_i^{adaptive} - J_i^{nominal} \rvert}{J_i^{nominal}}$
$Plant_1$	395	359	10%
$Plan_2$	52,5	40,04	$31,1\%$
$Plant_3$	199,4	154,6	$29,1\%$
$Plant_4$	26,347	25,31	$0,05\%$

Table 5.4. *Performance degradation of the four plants.*

$Plant_4$ suffered the lower cost degradation due to the fact that this system has to respect a hard real-time constraint $m_4 = k_4 = 1$. $Plant_2$ suffered the maximum cost degradation, due to the $Plant_2$ performance indicator, which generates the reduction of m_2 as soon as the other plants require the use of the processor.

5.5.4.3. *Summary*

Through this case study we can see that the on-line adaptation mechanism we proposed can effectively control the degradation of the system performance during the plant state transient period and system overload. In fact, given the current states of the controlled plants, the proposed approach allows us to derive a (m, k)-firm constraint for each control task and the corresponding optimal control gain while still meeting the (m, k)-firm schedulability condition of the total control tasks. Moreover, table 5.4 shows that comparing to the idle case where the processor has infinite capacity, using our approach does not induce much performance degradation. Notice that, for the simulated scenario, if we do not allow sample data dropping (i.e., in case that all tasks are considered under hard real-time constraint), the processor will be in an overload at time $t = 0.5$ s. As the tasks τ_3 and τ_4 have lower priority, they are no longer executed by the processor and this leads to the instability of $Plant_3$ and $Plant_4$. Figure 5.11 shows that starting from $t = 0.5$ s where the higher priority task τ_2 is released, the processor can no longer execute task τ_3 correctly and could never execute task τ_4. As an example, figure 5.12 clearly shows that $Plant_3$ is not stable.

5.6. Conclusions

Computing and networking resource sharing is a common trend in NCS for achieving cost-effective solutions. However, an overload situation may occur, either by the dynamic application configuration changes or by the implementation system performance variations.

For dealing with system overload situations, this chapter proposed an approach based on selectively dropping some samples while still guaranteeing the (m, k)-firm schedulability of the control tasks sharing a same processor. By adjusting both the acceptable (m, k) values and the control gains, we have shown, through case studies, that the performance degradation is efficiently controlled. The key point is the use of

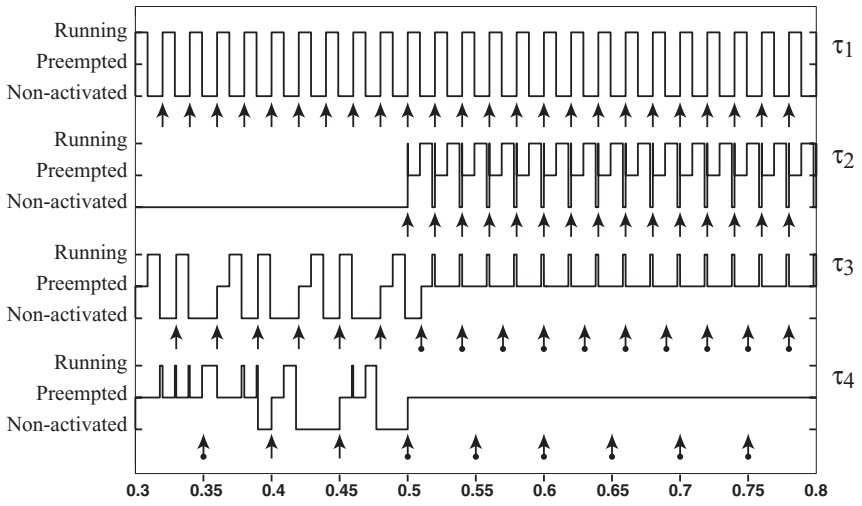

Figure 5.11. *Simulation trace of the tasks execution without selective sample dropping.*

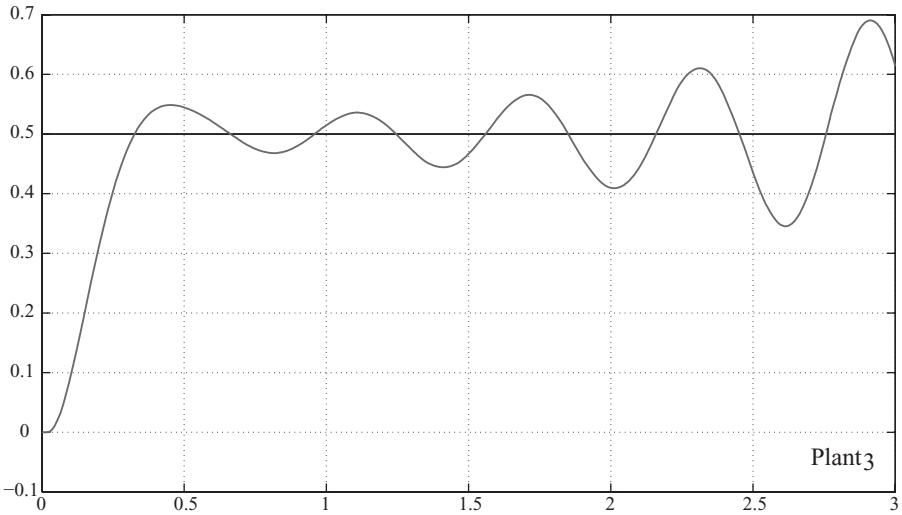

Figure 5.12. *Unstability of* $Plant_3$.

an algorithm that allows for finding on-line the optimal values of m and the control gains leading to minimizing a global control performance cost function.

This approach has been further extended to also take into account the transient plant state, where the plant needs more resources to be controlled than when it is in its steady state. The stability of this dynamic parameter switching closed-loop has been discussed, and we showed that the asymptotic stability is guaranteed if one can find a CQLF. This extension makes the resource utilization still more efficient. In fact, the idea behind this is that the shared resource is dynamically allocated to the plant that needs it (when it is in its transient phase), rather than being allocated to the plants that are in their steady-state and need less control.

As indicated in the introduction section of this chapter, the approach presented can be considered as an alternative to the direct sampling period adaptation for dealing with system overload situations. Using such an approach, the resulting system will be more robust as it can auto-adapt to fit with certain system overload situations.

5.7. Bibliography

[ÅST 97] ÅSTRÖM K., AND WITTENMARK B., *Computer-Controlled Systems*, Information and System Sciences Series, Prentice Hall, 3rd edition, 1997.

[BIT 91] BITTANTI S., COLANERI P., AND DE NICOLAO G., The periodic Riccati equation, *The Riccati Equation*, p. 127–162, Springer-Verlag, Berlin, 1991.

[CER 03] CERVIN A., HENRIKSSON D., LINCOLN B., EKER J., AND BERNHARDSSON B., ÅRZÉN K. E., How does control timing affect performance?, *IEEE Control Systems Magazine*, vol. 23, p. 16–30, June 2003.

[EKE 00] EKER J., HAGANDER P., AND ARZEN K.-E., A feedback scheduler for real-time controller tasks, *Control Engineering Practice*, vol. 8, p. 1369–1378, 2000.

[FRE 63] FREEMAN H., *Discret Time Systems*, John Wiley, New York, 1963.

[HAM 94] HAMDAOUI M., AND RAMANATHAN P., A service policy for real-time customers with (m,k)-firm deadlines, *Fault-Tolerant Computing Symposium*, Austin, USA, p. 196–205, April 1994.

[HAM 95] HAMDAOUI M., AND RAMANATHAN P., A dynamic priority assignment technique for streams with (m,k)-firm deadlines, *IEEE Transactions on Computers*, p. 1443–1451, December 1995.

[JIA 05] JIA N., HYON H., AND SONG Y., Ordonnancement sous contraintes (m,k)-firm et combinatoire des mots, *13th International Conference on Real-Time Systems*, Paris, France, April 2005.

[KHA 02] KHAN P., LI K., MANNING E., AND AKBAR M., Solving the knapsack problem for adaptive multimedia system, *Studia Informatica*, vol. 2, p. 161–182, 2002.

[LI 06] LI J., SONG Y.-Q., AND SIMONOT LION F., Providing real-time applications with graceful degradation of QoS and fault tolerance according to (m,k)-firm Model, *IEEE Transactions on Industrial Informatics*, vol. 2, p. 112–119, 2006.

[LIB 99] LIBERZON D., AND MORSE S., Basic problems in stability and design of switched systems, *Control Systems Magazine*, vol. 19, p. 59–70, October 1999.

[MAR 90] MARTELLO S., AND TOTH P., *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley, New York, 1990.

[OTA 02] OTANEZ P., MOYNE J., AND TILBURY D., Using deadbands to reduce communication in networked control systems, *American Control Conference*, Anchorage, USA, May 2002.

[QUA 00] QUAN G., AND HU X., Enhanced fixed-priority scheduling with (m,k)-firm guarantee, *21st IEEE Real-Time Systems Symposium*, Orlando, Florida, USA, p. 79–88, November 2000.

[RAM 99] RAMANATHAN P., Overload management in real-time control applications using (m,k)-firm guarantee, *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, p. 549–559, June 1999.

[SCH 02] SCHINCKLE M., CHEN W.-H., AND A. RANTZER, Optimal control for systems with varying samplin rate, *American Control Conference*, Anchorage, USA, May 2002.

[SET 96] SETO D., LEHOCZKY J. P., SHA L., AND SHIN K. G., On task schedulability in real-time control system, *17th IEEE Real Time Systems Symposium*, Washington, DC, USA, p. 13–21, December 1996.

[SIM 05] SIMON D., AND BENATTAR F., Design of real-time periodic control systems through synchronisation and fixed priorities, *International Journal of Systems Science*, vol. 36, p. 57–76, 2005.