On real-time feedback control systems : requirements, achievements and perspectives.

Daniel Simon
INRIA Grenoble Rhône-Alpes
Inovallée Montbonnot
38334 St Ismier Cedex, France
daniel.simon@inria.fr

Alexandre Seuret
GIPSA-lab - Control Systems Dpt
BP 46
38402 Saint Martin d'Hères Cedex, France
alexandre.seuret@gipsa-lab.fr

Olivier Sename
GIPSA-lab - Control Systems Dpt
$ENSE^3$ BP 46
38402 Saint Martin d'Hères Cedex, France
olivier.sename@gipsa-lab.fr

***Abstract*—The present article is a position paper reviewing the main requirements and existing achievements of co-design approaches for real-time control and computing. This problem arises with the increasing complexity of modern computers which require more integrated methodologies, specifically suited to critical embedded systems. The general problem to be solved is the achievement of multi-objective goals (i.e. mixing stability, performance and dependability requirements) under constraints of limited execution resources (combining hardware and software components in CPUs and networks). It is expected that co-design approaches, handling the constraints arising from the control and real-time computing domains at early design time, can improve the overall effectiveness of distributed real-time controllers.**

I. Framework and objectives

The design and implementation of real-time control systems, including embedded control components distributed over execution resources, combines the domains of control systems, communication networks and real-time computing [1], [2], [3], [4], [5].

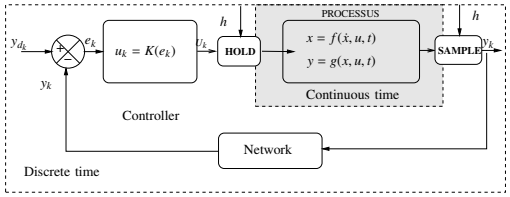

Figure 1. Control under execution resources constraints

From a control point of view, the problem to be solved is the achievement of a control objective, mixing stability, performance and reliability requirements. The controlled plant is most often a physical device which evolves in continuous time, for example whose dynamics can be described by a state space model :

$$x = f(\dot{x}, u, t) \quad y = g(x, u, t) \tag{1}$$

where x, y and u are vectors of continuous variables which respectively represent the state, the output and the control inputs of the plant.

In a computer-controlled system, the control objective is achieved by a feedback controller K which cyclically computes at some instants t_k the control inputs u_k from the error signal $e(k) = y_d(k) - y(k)$ between the desired and the sampled outputs of the plant (Figure1). Sampling and delays introduce distortions w.r.t. to the original continuous time framework, especially when limited computing resources induce long calculations and sampling intervals as shown in Figure 2. Anyway the control objectives must be achieved despite the disturbances induced by the distribution of the control system over execution resources with limited bandwidth and computing power.

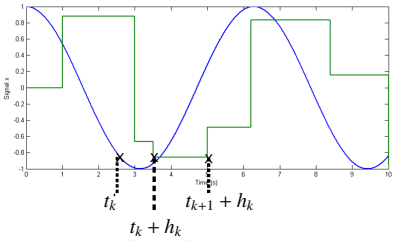

Figure 2. Deterioration of a continuous-time signal with delay and sampling

Historically, tools for the design and analysis of systems related to these disciplines have been designed and used with limited interaction. Feedback control is usually designed in continuous time, or in a basic sampled-data framework with equidistant sampling and constant delays, where the imperfections of real-life implementation constraints are not taken into account. On the other hand, real-time scheduling focus on purely computational aspects, e.g. schedulability of a task set knowing worst-case conditions, but fail to capture the properties and requirements of feedback controllers. It happens that a solution which is optimal in one domain is not well suited for the other, e.g. in [6] where the blind implementation of a control system under an optimal (Rate Monotonic) scheduling policy provides a poor control performance, while an *ad-hoc* implementation taking care of control-specific needs provides far better results. The poor interaction of control and computing at design time results both in poor control performance and in wasting execution resources.

Indeed, the increasing complexity of modern computer systems and the rapidly evolving technology of computer networks require more integrated methodologies, specifically suited to distributed control systems. Control is playing an increasing role in the design and run-time management of

large interconnected systems to enhance high performance in nominal modes and safe reconfiguration processes in the occurrence of faults and failures. In fact, it appears that the achievement of system-level requirements exceeds by far the achievable reliability of individual components [2]. The deep intrication of components and sub-systems from different technologies, and that are subject to various constraints, calls for a joint design to solve conflicting constraints early on.

Hence, a control/computing co-design approach is a promising direction for achieving the control of a real-time control system under execution resources constraints. This approach more or less tightly combines :

Computing-aware control where the control algorithms are specifically made robust and/or adaptive w.r.t. the execution induced constraints and disturbances;

Control-aware computing where the scheduling scheme and parameters are designed to adaptively capture the control system requirements and to improve the overall efficiency of a feedback controller;

Control of numerical objects is an appealing way to implement flexible and adaptive scheduling policies, where software objects are dynamically governed by feedback control loops.

This position papers reviews requirements to be solved in order to efficiently design and implement distributed control systems, and presents some achievements already done in this framework. The next section provides guidelines for control and computation co-design, from the analysis of requirements. Section III browses some computing-aware control designs and section IV examines how the real-time implementation can be made aware of control needs. Some perspectives are given in the conclusion.

II. Flexibility and robustness requirements

Control and real-time computing have been associated for a long time, for the control of industrial plants and in embedded systems, e.g. automotive and robotics. However both parts, control and computing, are often designed with poor interaction and mutual understanding. From the control design point of view, a constant and unique sampling period is usually assumed. Delays are supposed negligible or constant, and jitter is ignored. The implementation design then follows, trying to meet these assumptions.

Except in the case of failures due to hardware or software components, most processes usually run with nominal behavior : however, even in the nominal modes, neither the plant nor the execution resource parameters are perfectly known or modeled. A very conservative viewpoint consists in allocating system resources to satisfy the worst-case, which results in over-provisioned, and thus wasted, execution resources. From the control viewpoint, specific time-related deficiencies to be considered include poorly predictable jitter, delays and data-loss.

Real-time scheduling has mainly focused on how to dimension computing resources to meet deadlines, or equivalently, on the schedulability analysis for a given resource. Indeed the real-time community has usually considered that control tasks have fixed periods, hard deadlines and known worst-case execution times. This assumption has served the separation of control and scheduling designs, but has led to under utilization of CPU resources and inflexible design.

The hard and costly way consists in building a highly deterministic system, from the hardware, operating system and communication protocols sides, so that the actual timing parameters meet the ideal ones. However trying to nullify (even virtually) latencies and jitter generally leads to worst-case based resources proportioning and tends to needlessly over-constraint the system's design and implementation. Thus it deserves to carefully re-examine the real hardness of the real-time constraints related with feedback control systems.

Guidelines for co-design

A new co-design domain, more or less tightly combining control and real-time computing features, has been opened by both computer scientists [7] and control scientists [8]. New research directions and potential applications have arisen from this co-design paradigm :

- Computing devices can be controlled by feedback loops, thus they can take benefit of feedback properties such as adaptivity and robustness. In particular real-time systems can be controlled by flexible policies implemented through feedback schedulers;
- Feedback loops are *not* hard real-time : indeed, as they are robust w.r.t. the plant's parameter uncertainties, they also are robust w.r.t. some amount of jitter, deadlines misses and data loss. Therefore new degrees of freedom based on slackened real-time constraints can be exploited to implement feedback loops.

III. Control under computing constraints

A. Delays and data-loss in real-time control systems

Control systems are often cited as examples of "hard real-time systems" where jitter and deadline violations are strictly forbidden, e.g. [9]. In fact experiments show that this assumption may be false for closed-loop control. Any practical feedback system is designed to obtain some stability margin and robustness w.r.t. the plant parameters uncertainty. This also provides robustness w.r.t. timing uncertainties : closed-loop systems are able to tolerate some amount of sampling period and computing delays deviations, jitter and occasional data loss without loss of stability or integrity. For example in [6] the loss of control performance has been checked experimentally using an ope-loop unstable inverted pendulum, for which a Linear Quadratic (LQ) controller has been designed according to a nominal sampling period and null delay and jitter. It was checked that the control system keeps stable with moderate loss of performance even for quite large jitter, delays and period drift.

Delays appear naturally in the modeling of several physical processes, where delays often come from the transportation of materials or information. Stability analysis of time-delay

systems is thus an important topic in many disciplines of science and engineering [10], [11], [12]. Motivating applications are found in diverse areas such as biology, chemistry, telecommunication control engineering, economics, and population dynamics [13]. There has been an increased interest in the area of time-delay systems over the last two decades due to the emerging area of networked control systems (NCS), which are systems where sensor and actuator devices communicate with control nodes over a communication network [3]. In such systems, processing time and preemption in the network nodes, together with propagation delays in the inter-node communication, necessarily leads to time delays affecting the overall closed-loop control system. Various phenomena related to delays in networked controlled systems have recently been considered, e.g., packet losses [5], [14], stabilization w.r.t. varying delays [15] and robust sampling [16].

B. Robustness against delays and data loss

The problem of ensuring stability under delays and sampling has been widely addressed in the literature, [4], [5], and is often referred as a problem of networked control systems. There are mainly four approaches to solve this problem. The first one consists in the analysis of the discretized model of the system (1) using the discrete-time version of the Lyapunov Theorem [17], [18], [19]. The main advantages of this approach are that the resulting conditions are tight and less conservative than other methods. However, it suffers from the complexity of the conditions and the difficulty to include uncertainties in the original system. The second approach is generally referred as the input delay approach introduced in [16] and employed for instance in [20] among many others. This method allows for using the continuous-time Lyapunov-Krasovskii framework in order to take into account the the sampling as a particular type of delay. The main advantages of the method is the possibility to take into account time-varying uncertainties both in the sampling period and in the systems parameters. However the resulting conditions are generally more conservative than the previous method. [14] introduced an impulsive system approach which refines the previous input delay one. It was further refined in [21]. A last approach was introduced in [22], [23] which proposed an equivalence theorem between the discrete-time approach and the input delay or the impulsive systems approaches using a new class of functionals referred as "looped functionals". The main advantage of this last method comes from the relaxation of the constraints on the functionals in comparison to the two previous approaches.

C. Variable scheduling parameters

Managing the computing cost of a control task can be done in several ways:

- Increasing the sampling control interval h can be easily done in real-time, providing that the control interval remains high enough to preserve the system stability and that interval switching is also handled from the stability viewpoint;
- Reducing its execution time c, e.g. by using a simplified version of the control algorithm : this requires a control tasks switching, which may be complex to be safely performed in real-time;
- The Dynamic Voltage and Frequency Scaling (DVFS) capabilities of modern computing chips can be exploited to dynamically adapt the CPU computing speed and minimize the energy cost of computing.

Anyway, a key actuator to be used for CPU utilization or network bandwidth control is the control interval. Concerning co-design for on-line implementation, recent results deal with varying sampling rates in control loops, in the framework of linear systems. Unfortunately, most real-life systems are non-linear and the extrapolation of timing assignment through linearization often gives rough estimations of allowable periods and latencies or they can even be meaningless. In fact, the knowledge of the plant behavior is necessary to get an efficient control/scheduling co-design.

As mentioned before the key actuator to be used for CPU utilization or network bandwidth control is the control interval. A first idea is to design a bank of controllers, each of them being designed and tuned for a specific sampling frequency, and to switch between them according to the decisions of the feedback scheduler. However, it has been observed that switching without caution between such controllers may lead to instability although each controller in isolation is stable [24]. Let us cite some possible solutions to guarantee control stability during sampling rate variations :

1) Control under (m,k)-firm scheduling constraints: This sampling period adjustment approach is based on selective sampling data dropping according to the (m,k)-firm model, where it is ensured that at least m out of k consecutive instances of tasks (or messages) are executed ([25]). The interest of this method is an easy implementation since only the multiples of the basic sampling period are exploited (which is well suited for non-preemptive scheduling as in communication networks). 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 or processor will be less loaded. However, the control stability and performance must still be maintained to an acceptable level. A scheduling architecture for handling such configurations has been proposed to both design and tune a stable feedback controller, and to find optimal scheduling patterns compliant with the (m,k)-firm scheduling constraint. 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. Extensive presentations of this approach, i.e. co-design of control loops and computing resources management subject to (m,k)-firm scheduling, can be found, e.g. in [26]. The application of co-design between LQ control and (m,k)-firm scheduling for a quadrotor drone control and diagnosis under networked induced data loss has been done, e.g. in [27].

2) LPV modeling and variable sampling: Fine grain variations of the sampling intervals can be implemented, e.g. using the features of preemptive operating systems. To ensure the system stability for a measured variable input delay, [28] pro-

poses for example a gain scheduled approach based on time-varying observers and state feedback controllers, synthesized using linear matrix inequalities (LMI) and quadratic Lyapunov functions. Indeed, gain scheduling is a popular approach to control non-linear plants, allowing to some extent to re-use the well-known linear control theory and tools [29].

LPV (Linear Parameter Varying) approaches have been then developed to enforce the overall control system stability during switching. An important feature of LPV based design is that the stability of the control system is ensured whatever the variations of the parameters, provided that they stay inside the predefined bounds. In ([30]) the LPV polytopic approach is used to design a control law with adaptation of the sampling period to account for the available computing resources for an inverted pendulum. As the sampling interval can be changed arbitrarily fast between its allowed values, such a control law can be easily used as a building block of a feedback-scheduler in charge of the computing resource management.

However the main drawback of the polytopic method could be the large number of LMIs to solve as the number of varying parameters increases. This is not the case in the Linear Fractional Transform (LFT) method and, as emphasized in ([31]), it leads to a LMI problem whose solution can directly be implemented. Moreover, this approach allows to consider in the same structure varying parameters, uncertainties and some non-linearities.This latter approach merging varying sampling with plant non-linearities modeled as varying parameters has been successfully applied to the case of bottom-referenced control of an Autonomous Underwater Vehicle (AUV), where the acoustic altitude sensors are subject to asynchronous sampling ([32]).

IV. Control-aware computing

A. Flexible real-time scheduling

To implement a controller, the basic idea consists in running the whole set of control equations in a unique periodic real-time task, whose clock gives the controller sampling rate. In fact, all parts of the control algorithm do not have an equal weight and urgency w.r.t. the control performance. To minimize the I/O (sensor-to-actuator) latency, a control law can be basically implemented as two real-time blocks, the urgent one sends the control signal directly computed from the sampled measures, while updating the state estimation or parameters can be delayed or even more computed less frequently [33].

In fact, a complex system involves sub-systems with different dynamics which must be further coordinated [1]. Assigning different periods and priorities to different blocks according to their relative weight allows for a better control of critical latencies and for a more efficient use of the computing resource. However in such cases finding adequate periods for each block is out of the scope of current control theory and must be done through case studies, simulation and experiments, e.g. [34].

Latencies have several sources : the first one comes from the computation duration itself, whose values are distributed between best-case and worst-case bounds (Figure 3, even in dedicated embedded CPUs. In multi-tasking systems they come from preemption due to concurrent tasks with higher priority, from precedence constraints and from synchronization. Another source of delays is the communication medium and protocols when the control system is distributed on a network of inter-connected devices. In particular it has been observed that in synchronous multi-rate systems the value of sampling-induced delays show complex patterns and can be surprisingly long, e.g. [35].

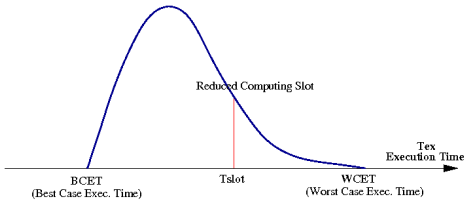

Figure 3. Typical execution time distribution

Besides feedback control considerations, flexible and control aware solutions have been also provided by the computer science side, e.g. through a Quality of Service (QoS) formalism. For example the “Elastic Tasks” paradigm [36] considers the sensitivity of the QoS relative to the execution period for every task (modeled by a “stiffness”), and takes account bounds in the allowed execution period. To make the tasks set schedulable, the tasks stack is “compressed” until the accumulated execution load fit with the allocated CPU capacity. Although this implementation is in open loop w.r.t. the actual QoS it allows for an improved adaptation against transient overloads.

As sharing the computing resource between controllers is a central issue, some variants of the CBS (Control Bandwidth Server) approach have been used to enforce protection between competing control activities, for example the Control-Server ([37]) where a fraction of the total CPU power is statically reserved to each control thread. Then the system behaves as if each controller was isolated using its own computation resource, in particular an overloaded controller does not disturb its neighbors. Inside each computing segment the individual controllers are organized to minimize their I/O latency and jitter. In case of transient overload the missing computing budget for one controller is postponed to its next reserved slice, with no impact on the others.

B. Weakened scheduling schemes

Robustness in control usually deals with the plant’s parameter uncertainties, but the insensitivity or adaptability w.r.t. timing deviations from the theoretical pattern, such as jitter or deadlines misses can be exploited. For SISO linear systems robustness can be quantified using phase margins, delay margins and module margins. It appears that a phase margin implies a delay margin (i.e. the maximum unmodeled constant extra delay that can be suffered before instability) and certainly a jitter margin, which is more difficult to quantify ([38]) but which can be experimentally shown ([6]). A feedback control system can be even robust enough to tolerate missed samples, e.g. as in the (m,k)-firm scheduling approach cited in section

III-C1. The interesting point is that a feedback control system which is robust w.r.t. the plants parameters uncertainties is also robust, to some extent, w.r.t. timing deviations. Hence a feedback control system is not as hard as it is often considered in the literature, but should be better considered as *weakly hard*, i.e. able to tolerate specified timing deviations without leaving its requested performance [39].

As already observed and reported in aforementioned references, it is likely that a robust feedback control systems can keep stability despite occasional data loss, at the price of some degradation in performance and robustness. To improve the average efficiency of embedded computers while preserving the control stability and performance, and relying on the robustness of feedback control laws, it has been proposed (e.g. [40]) to slacken the usual hard real-time constraints and worst-case based implementations.

Considering a typical distribution of execution times for a real-time time task on a given CPU (Figure 3), the time slot T_{slot} allocated to a periodic control task is chosen smaller than its worst case execution time (WCET). Hence, the deadline of the task can be occasionally missed, in that case the running task can be for example aborted (and the last control signal frozen for an extra period), or allowed to continue until completion and the next execution is skipped. But most of the time the task can be fully executed within the allocated time slot. From the control point of view, the result is a reduction of the average control latencies and periods, keeping control stability despite several successive sample loss, and even inducing an improvement of its robustness as formally proved in [41]. From the computation point of view, the needed computing power is reduced compared with a worst-case based implementation, thus saving weight and energy which are precious in embedded systems. Finally, such a co-design approach enlighten a new viewpoint about real-time control systems fault-tolerance, where jitter and deadline misses should be processed as specified disturbances rather than fatal failures.

C. Feedback scheduling

Robust and/or varying sampling control loops provide building blocks to be used in more complex control structures where outer loops, e.g. based on Quality of Service considerations, govern end-users level control objectives. In fact the underlying idea is that digital objects, e.g. real-time schedulers, can be also objects that can be controlled by feedback loops. This approach has been initiated both from the real-time computing side [42], [7] and from the control side [8], [43]. The idea consists in adding to the process controllers an outer sampled feedback loop ("scheduling controller") to control the scheduling parameters as a function of a QoC (Quality of Control) measure. For example such loops can be used to control a trade-off between conflicting objectives, like the control performance (e.g., measured by the rise-time and bandwidth) and the CPU or sensors' cost used to implement the controller. It is expected that an on-line adaption of the scheduling parameters of the controller may increase its overall efficiency w.r.t. timing uncertainties coming from the unknown controlled environment. Also it is known from control theory that closing the loop may increase performance and robustness against disturbances when properly designed and tuned (otherwise it may lead to instability).

In its basic form a feedback scheduler cyclically computes updated values of the execution parameters, e.g. tasks control intervals or CPU clock frequency, from measures taken on the execution resources, e.g. CPU or network's load. In that case, the dynamics of the controlled system, e.g. a stack of real-time tasks ready for execution, is quite simple. For example, fluid modeling provides an adequate level of abstraction for control purpose, where a digital system can be viewed as flows of incoming requests and executed services [44]. Tasks and message queues behave as integrators, and often the main source of dynamics to be considered for control design is the bandwidth of the measuring filters. Hence, feedback schedulers can be easily implemented, e.g. as an additional real-time task, using off-the-shelf hardware and software components as shown for example in [34] for a multirate robot controller or in [45] for an adaptive video decoder. Note that in that case, transient computing overloads are automatically recovered within a few samples.

More elaborated scheduling control loops are expected to govern the control tasks execution parameters as a function of the controlled plant performance itself. The design problem can be stated as control performance optimization under constraint of available computing resources. Early results come from [8] where a problem of optimal control under computation load constraints is theoretically solved by a feedback scheduler, but leads to a solution too complex to be implemented in real-time. Formalizing the problem is not obvious, as constraints from one domain (e.g. scheduling) must be translated into the other (e.g. feedback), and most often results in non-linear and hybrid models. The design of feedback controllers to solve such problems in real-time (with resources of moderate size) does not lead to both general and effective solutions, and control solving relies on case studies [46].

V. Summary and perspectives

Co-design approaches have already shown that they can provide both safe and cost-effective real-time control systems. In particular, the robustness provided by feedback allows for slackening the real-time scheduling constraints and for saving computing power and networking bandwidth. On the other hand, numerical objects can be controlled using feedback loops, therefore improving their adaptivity and robustness w.r.t. uncertain and variable operating conditions.

A better understanding and integration of control and computing is becoming necessary in systems of increasing complexity. For example, out-coming tiny-scale CMOS chips are subject to unavoidable characteristics dispersion, and integrating feedback control in the hardware is now essential. Also, in modern processors the execution time dispersion (Figure 3) is spreading over very large $\frac{worst-case}{best-case}$ ratios, and relying on robust control and weakly-hard real-time is a key to avoid unacceptable wastes in computing power and energy.

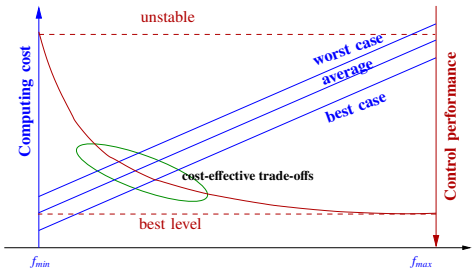

Figure 4. Control performance vs computing cost

It is expected that cost-effective solutions will emerge from trade-offs modeling and multi-criteria optimization between conflicting features of the different domains involved, as sketched in Figure 4.

References

[1] M. Törngren, "Fundamentals of implementing real-time control applications in distributed computer systems," *Real Time Systems*, vol. 14, no. 3, pp. 219–250, 1998.

[2] R. M. Murray, K. J. Åström, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE Control Systems Magazine*, vol. 23, no. 2, Apr 2003.

[3] J. Baillieul and P. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 9–28, 2007.

[4] S. Zampieri, "A survey of recent results in Networked Control Systems," in *Proc. of the* 17th *IFAC World Congress*, Seoul, Korea, July 2008.

[5] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[6] A. Cervin, "Integrated control and real-time scheduling," Ph.D. dissertation, Dpt of Automatic Control, Lund Inst. of Technology, Sweden, Apr. 2003.

[7] C. Lu, J.-A. Stankovic, G. Tao, and S.-H. Son, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real Time Systems*, vol. 23, no. 1, pp. 85–126, 2002.

[8] J. Eker, P. Hagander, and K.-E. Arzen, "A feedback scheduler for real-time controller tasks," *Control Engineering Practice*, vol. 8, no. 12, pp. pp 1369–1378, 2000.

[9] G. C. Buttazzo, *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications*. Springer, 2011.

[10] K. Gu, V.-L. Kharitonov, and J. Chen, *Stability of time-delay systems*. Birkhauser, 2003.

[11] S.-I. Niculescu, *Delay Effects on Stability. A Robust Control Approach*. Springer-Verlag, 2001.

[12] J.-P. Richard, "Time delay systems: an overview of some recent advances and open problems," *Automatica*, vol. 39, no. 10, 2003.

[13] V. Kolmanovskii and A. Myshkis, *Applied theory of functional differential equations*. Kluwer, 1999.

[14] P. Naghshtabrizi, J. Hespanha, and A. Teel, "Exponential stability of impulsive systems with application to uncertain sampled-data systems," *Systems and Control Letters*, vol. 57, no. 5, pp. 378–385, 2008.

[15] E. Witrant, C. Canudas-de Wit, D. Georges, and M. Alamir, "Remote stabilization via communication networks with a distributed control law," *Automatic Control, IEEE Transactions on*, vol. 52, no. 8, 2007.

[16] E. Fridman, A. Seuret, and J.-P. Richard, "Robust sampled-data stabilization of linear systems: An input delay approach," *Automatica*, vol. 40, no. 8, pp. 1141–1446, 2004.

[17] M. Cloosterman, N. van de Wouw, W. P. M. H. Heemels, and H. Nijmeijer, "Stability of networked control systems with uncertain time-varying delays," *IEEE Trans. on Automatic Control*, vol. 54, no. 7, 2009.

[18] Y. Oishi and H. Fujioka, "Stability and stabilization of aperiodic sampled-data control systems: An approach using robust linear matrix inequalities," in *Joint* 48th *IEEE CDC and* 28th *Chinese Control Conf.*, December 16-18 2009.

[19] L. Hetel, J. Daafouz, and C. Iung, "Stabilization of arbitrary switched linear systems with unknown time-varying delays," *IEEE Transactions on Automatic Control*, vol. 51, no. 10, pp. 1668 – 1674, 2006.

[20] H. Gao, T. T. Chen, and J. Lam, "A new delay system approach to network-based control," *Automatica*, vol. 44, no. 1, pp. 39–52, 2008.

[21] K. Liu and E. Fridman, "Wirtinger's inequality and lyapunov-based sampled-data stabilization," *Automatica*, vol. 48(1), pp. 102–108, 2012.

[22] A. Seuret, "A novel stability analysis of linear systems under asynchronous samplings," *Automatica*, no. 1, p. 177 182, 2012.

[23] ——, "Stability analysis of networked control systems with asynchronous sampling and input delay," in *IEEE American Control Conference*, 2009, pp. 533 – 538.

[24] M. Schinkel, W.-H. Chen, and A. Rantzer, "Optimal control for systems with varying sampling rate," in *American Control Conference ACC'02*, Anchorage, USA, May 2002.

[25] M. Hamdaoui and P. Ramanathan, "A service policy for real-time customers with (m,k)-firm deadlines," in *Fault-Tolerant Computing Symposium*, Austin, USA, April 1994, pp. 196–205.

[26] F. Felicioni, N. Jia, Y.-Q. Song, and F. Simonot Lion, "Optimal on-line (m,k)-firm constraint assignment for real-time control tasks based on plant state information," in *12th IEEE Conf. on Emerging Technologies and Factory Automation*, Hamburg, Germany, 2008.

[27] J.-F. Guerrero-Castellanos, C. Berbra, S. Gentil, and S. Lesecq, "Quadrotor attitude control through a network with (m-k)-firm policy," in *European Control Conference ECC09*, Budapest, Hungary, august 2009.

[28] A. Sala, "Computer control under time-varying sampling period: An lmi gridding approach," *Automatica*, vol. 41, no. 12, pp. 2077–2082, 2005.

[29] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *Int. Journal of Control*, vol. 73, no. 11, pp. 1001–1025, 2000.

[30] D. Robert, O. Sename, and D. Simon, "An H_∞ LPV design for sampling varying controllers : experimentation with a t inverted pendulum," *IEEE Trans. on Control Systems Technology*, vol. 18, no. 3, pp. 741–749, 2010.

[31] P. Apkarian and P. Gahinet, "A convex characterisation of gain-scheduled $\mathcal{H}_\infty$ controllers," in *IEEE Transaction on Automatic Control*, vol. 40, May 1995, pp. 853 – 864.

[32] E. Roche, O. Sename, and D. Simon, "An LFR approach to varying sampling control of LPV systems: application to auvs," in *submitted to the 1st Int. Conf. on Systems and Computer Science*, Lille, France, 2012.

[33] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed., ser. Information and System Sciences Series. Prentice Hall, 1997.

[34] D. Simon, D. Robert, and O. Sename, "Robust control / scheduling co-design: application to robot control," in *11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA, Mars 2005.

[35] B. Wittenmark, "Sample-induced delays in synchronous multirate systems," in *European Control Conference*, Porto, Portugal, sep 2001.

[36] G. Buttazzo and L. Abeni, "Adaptive rate control through elastic scheduling," in *39th CDC*, Sydney, Australie, dec 2000.

[37] A. Cervin and J. Eker, "The Control Server: A computational model for real-time control tasks," in *15th Euromicro Conference on Real-Time Systems*, Porto, Portugal, July 2003, pp. 113–120.

[38] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo, "The jitter margin and its application in the design of real-time control systems," in *10th Int. Conf. on Real-Time and Embedded Computing Systems and Applications*, Göteborg, Sweden, Aug. 2004.

[39] G. Bernat, A. Burns, and A. Llamosí, "Weakly hard real-time systems," *IEEE Trans. on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[40] P. J. Andrianiaina, D. Simon, A. Seuret, J.-M. Crayssac, and J.-C. Laperche, "Weakening Real-time Constraints for Embedded Control Systems," INRIA, Rapport de recherche RR-7831, Dec. 2011.

[41] P.-J. Andrianiaina, A. Seuret, and D. Simon, "Robust control under weakened real-time constraints," in *50th IEEE CDC/ECC*, Orlando, USA, December 2011.

[42] C. Lu, J. Stankovic, T. Abdelzaher, G. Tao, S. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," in *Real-Time Systems Symposium*, December 2000.

[43] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Arzen, "Feedback-feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, no. 1–2, pp. 25–53, July 2002.

[44] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, *Feedback Control of Computing Systems*. New York: Wiley-IEEE Press, 2004.

[45] A.-M. Alt and D. Simon, "Control strategies for h.264 video decoding under resources constraints," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 53–58, 2010.

[46] M. Ben Gaïd, D. Robert, O. Sename, and D. Simon, "Plant state based feedback scheduling," in *Co-design approaches for dependable networked control systems*, C. Aubrun, D. Simon, and Y.-Q. Song, Eds. ISTE Wiley, 2010, pp. 149–183.