Conference on Control and Decision, December, 2000, Sydney, Australia

An Introduction to Control and Scheduling Co-Design

Karl-Erik Årzén, Anton Cervin, Johan Eker

Department of Automatic Control
Lund Institute of Technology
Email:{karlerik/anton/johane}@control.lth.se

Lui Sha

Department of Computer Science
University of Illinois at Urbana-Champaign
Email: lrs@uiuc.edu

Abstract

The paper presents the emerging field of integrated control and CPU-time scheduling, where more general scheduling models and methods that better suit the needs of control systems are developed. This creates possibilities for dynamic and flexible integrated control and scheduling frameworks, where the control design methodology takes the availability of computing resources into account during design and allows on-line trade-offs between control performance and computing resource utilization.

1. Introduction

Most control systems are embedded systems where the computer is a component in a larger engineering system. The control system is often implemented on a microprocessor using a real-time kernel or a real-time operating system (RTOS). The real-time kernel or OS uses multiprogramming to multiplex the execution of the tasks on the CPU. The CPU time, hence, constitutes a shared resource which the tasks compete for. To guarantee that the time requirements and time constraints of the individual tasks are all met, it is necessary to schedule the usage of the shared resource. During the last two decades, scheduling of CPU time has been a very active research area and a number of different scheduling models and methods have been developed.

The most common, and simplest, model used within the real-time scheduling community assumes that the tasks are periodic, or can be transformed to periodic tasks, with a fixed period, T_i, a known worst-case bound on the execution time (WCET), C_i, and a *hard deadline*, D_i. The latter implies that it is imperative that the tasks always meets their deadlines, i.e., that the actual execution time (response time) is always less or equal to the deadline, for each invocation of the task. This is in contrast to a *soft deadline*, that may occasionally be violated.

The most common example used by the real-time scheduling community for when this model is applicable is in computer-controlled systems. The fixed-period assumption of the simple task model has also been widely adopted by the control community and has, e.g., resulted in the development of the sampled computer-control theory with its assumption on deterministic, equi-distant sampling. Another result of the simple model is that it has provided a separation between the control community and the real-time scheduling community. The separation has allowed the control community to focus on its own problem domain without worrying about how scheduling is being done, and it has released the scheduling community from the need to understand what impact scheduling has on the stability and performance of the plant under control. From a historical perspective, the separated development of control and scheduling theories for computer-based control systems has produced many useful results and served its purpose. However, the separation has also had negative effects. The two communities have partly become alienated, which has led to a lack of mutual understanding between the fields. A closer interaction between the fields is also needed for control applications requiring high degrees of flexibility, or when computing resources are limited.

The aim of this paper is to present the emerging field of integrated control and scheduling. In this field a closer interaction between control design and scheduling is employed, and more general scheduling models and methods that better suit the needs of control systems are developed. The development of more general scheduling models, and the complementary control theory, create a possibility for dynamic and flexible integrated control and scheduling frameworks where the control design methodology takes the availability of computing resources into account during design and allows on-line trade-offs between control performance and computing resource utilization. The computing resources could include CPU time and communication bandwidth. Here, we will, however, focus on CPU time. A more extensive survey can be found in [Årzén *et al.*, 1999].

2. Real-Time Scheduling

In 1973, Liu and Layland proposed two optimal priority-based scheduling algorithms, *earliest deadline first scheduling* (EDF) and *rate-monotonic scheduling* (RM), [Liu and Layland, 1973]. EDF is based on the principle that the task with the shortest remaining time to its deadline should run. The EDF approach is dynamic in the sense that the priority between the tasks are decided dynamically on-line. The deadline can also be viewed as a dynamic priority, in contrast to the RM case where the priority is fixed. Rate-monotonic scheduling is sometimes referred to as fixed priority scheduling.

In the simplest case, i.e., $D_i = T_i$, no interprocess communication, and an ideal real-time kernel, the schedulability condition for EDF is that the CPU utilization, U, should be less than 100 %, i.e.,

$$U = \sum_{i=1}^{i=n} \frac{C_i}{T_i} \leq 1$$

For RM scheduling a sufficient condition is that

$$U \leq n(2^{1/n} - 1)$$

A sufficient and necessary condition based on the calculation of the response times, R_i, i.e., the worst-case execution time in the presence of the other tasks, was developed in [Joseph and Pandya, 1986]. During the last decade the RM and EDF analysis have been generalized and extended, e.g., [Klein *et al.*, 1993].

3. Control Loop Timing

A control loop consists of three main parts: data collection, control algorithm computation, and output transmission. In most cases the control is executed periodically with a constant sampling period determined by the process dynamics and the requirements on the closed loop performance.

The two basic timing constraints of a control loop are shown in Fig. 1. The first is the period which should be constant, i.e., without jitter. The second constraint involves the input-output latency, also known as the *control delay*. This should be as small as possible, and also without jitter. From a control perspective, sampling jitter and latency jitter can be interpreted

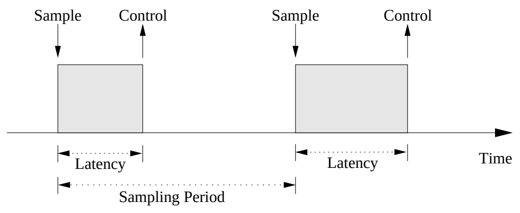

Figure 1: Basic timing constraints of a control loop.

as disturbances acting on the control system. The input-output latency decreases the stability margin and limits the performance of the system. If the jitter and the latency are small, they could be ignored. Otherwise, they should be accounted for in the control design.

Scheduling theory can be used to analyze the time variations and delays in control loops when implemented as real-time tasks. Understanding the control requirements, the implementation could be made such that the resulting delay and the jitter are small.

The following example shows that a simple-minded implementation of control loops can introduce a lot of jitter and delays:

EXAMPLE 1
Three control loops with different sampling periods are implemented in a priority-preemptive real-time OS with rate-monotonic priority assignment. The task code for each control loop looks like this:

```
t := currentTime;
LOOP
  AD-Conversion;
  ControlAlgorithm;
  DA-Conversion;
  t := t + h;
  WaitUntil(t);
END
```

Assume that the execution time is 2 ms for all three tasks, and that the sampling periods are $T_1 = 12$ ms, $T_2 = 8$ ms, and $T_3 = 5$ ms. Fixed priorities are assigned to the tasks according to the rate-monotonic theory. Figure 2 shows the execution graph of the three control tasks when released at time zero. Task 3 has the shortest period, thus the highest priority, and executes with perfect periodicity. Tasks 1 and 2, on the other hand, are frequently preempted. The preemption causes variations in both the sampling period and in the input-output latency. □

The above situation where a controller task is disturbed by the execution of other higher priority, controller or non-controller, tasks is very common. However, nondeterminism is common also in the absence

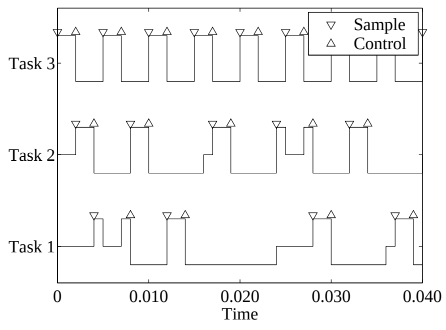

Figure 2: The activation graph (high=running, medium=preempted, low=sleeping) of the three control tasks in Example 1
.

of competing tasks. There are strong market trends in the direction of using general purpose hardware and off-the-shelf operating systems also for control system implementations. These systems are designed to achieve good average performance rather than guaranteed worst-case performance. They often introduce significant non-determinism in task scheduling. For compute-intensive high-end applications, the large variability in execution time caused by modern hardware architecture also becomes visible. The effect of this is again jitter in sampling period and control delay.

4. Timing Compensation

The reason why conventional real-time kernels and scheduling theory still can be used for control system implementation is the robustness of most control loops to timing variations. However, in many cases better performance can be obtained if the controller is allowed to actively compensate for the variations from sample to sample by, e.g., recomputing the controller parameters. This requires that the necessary time measurements are available.

EXAMPLE 2—SAMPLING JITTER
Consider PD control of a DC servo. The goal of the control is to make the servo position, $y(t)$, follow the reference position, $r(t)$, as closely as possible. Let the servo be described by the continuous-time transfer function

$$G(s) = \frac{1000}{s(s+1)}.$$

A good implementation of the PD controller, which includes filtering of the derivative part, is

$$\begin{aligned} P(t) &= K(r(t) - y(t)), \\ D(t) &= a_d D(t-h) + b_d(y(t-h) - y(t)), \\ u(t) &= P(t) + D(t), \end{aligned}$$

where $a_d = \frac{T_d}{Nh+T_d}$, $b_d = \frac{NKT_d}{Nh+T_d}$.

A nominal sampling period of $h = 10$ ms is chosen, and the PD controller is tuned to give a fast and well-damped response to set-point changes. The resulting parameters are $K = 1$, $T_d = 0.04$, and $N = 30$. The parameters a_d and b_d are normally pre-calculated, assuming that the sampling interval is constant.

A first simulation of the closed-loop system, where there is no jitter in the sampling interval, is shown in Fig. 3. The controller behaves as expected, and the performance is good. A second simulation, where the

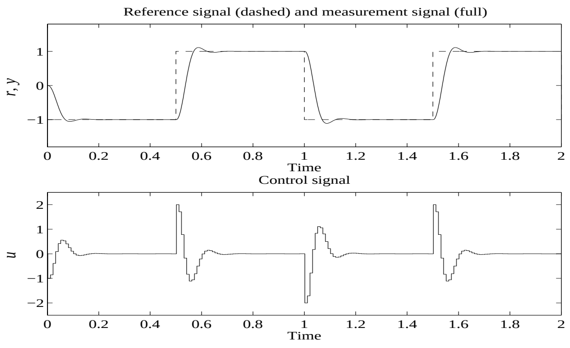

Figure 3: When no sampling jitter is present, the control performance is good.

actual sampling interval varies randomly between $h_{min} = 2$ ms and $h_{max} = 18$ ms, is shown Fig. 4. The sampling jitter causes the controller to repeatedly take either too small or too large actions. The resulting performance is quite poor. This is especially visible in the control signal. Finally, the controller is

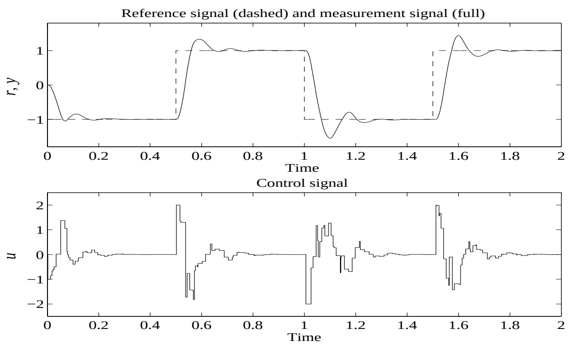

Figure 4: Sampling jitter causes the control performance to degrade.

redesigned to compensate for the jitter. This is done by measuring the actual sampling interval and recalculating the controller parameters a_d and b_d at each sample. Fig. 5 shows that this version of the controller handles the sampling jitter well. □

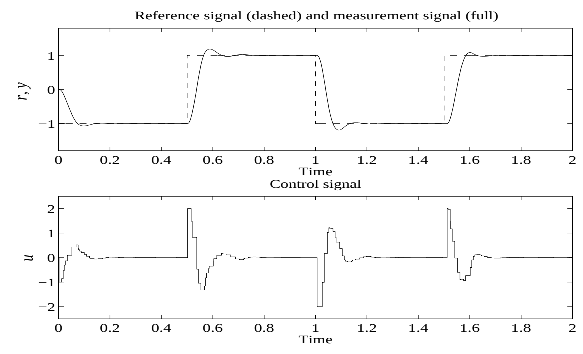

Figure 5: When compensating for the sampling jitter, the control performance is good again.

5. Task Attribute Adjustments

The possibility for controllers to compensate for timing variations was in the previous section used to reduce the effects of nondeterminism. It can, however, also be used as a way to increase flexibility. Assume that the computer contains a set of controller tasks. The number of controller tasks and their execution time bounds may change over time. The latter can be due to too optimistic execution time bounds, causing occasional overruns, or due to different operation modes in the controller. As the workload changes, the scheduler may then adjust the task attributes, e.g., the sampling periods, of the controller tasks in order to optimize global control performance under the constraint that the task set should remain schedulable.

A prerequisite for this type of on-line integration of control and scheduling is that it is possible to make an integrated off-line design of control algorithms and scheduling algorithms. Such a design process should allow an incorporation of the availability of computing resources into the control design. This is an area where, so far, relatively little work has been performed. One of the first references that addressed the problem was [Seto *et al.*, 1996]. An algorithm was proposed that translates a control performance index into task sampling periods considering schedulability among tasks running with preemptive priority scheduling. The sampling periods were considered as variables and the algorithm determined their values so that the overall performance was optimized subject to the schedulability constraints. On-line application of the approach is suggested in [Shin and Meissner, 1999].

An approach to optimization of sampling period and input-output latency subject to performance specifications and schedulability constraints is presented in [Ryu and Hong, 1998]. The performance is specified in terms of steady state error, overshoot, rise time, and settling time. These performance parameters are expressed as functions of the sampling period and the input-output latency. A heuristic iterative algorithm is proposed for the optimization of the parameters subject to schedulability constraints.

Much of the work on dynamic task adaptation during recent years is motivated by the requirements of multimedia applications. Activities such as voice sampling, image acquisition, sound generation, and video playing are performed periodically, but with less rigid timing requirements. Missing a deadline may decrease the quality of service (QoS) but does not cause critical system faults. Depending on the requested QoS, tasks may adjust their attributes to accommodate the requirements of other concurrent activities.

In [Buttazzo *et al.*, 1998] an elastic task model for periodic tasks is presented. A task has has an associated elasticity coefficient $e_i \geq 0$, and may change its period within certain bounds. When this happens the periods of the other tasks are adjusted so that the overall system is kept schedulable. An analogy with a linear spring is used, where the utilization of a task is viewed as the length of a spring that has a given rigidity coefficient $(1/e_i)$ and length constraints. The elasticity coefficient is used to denote how easy or difficult it is to adjust the period of a given task (compress the string). A task with $e_i = 0$ can arbitrarily vary its period within its range, but it cannot be varied by the scheduler during load reconfiguration. The approach can be used under fixed or dynamic priority scheduling. Task attribute adjustment strategies are also presented in [Nakajima, 1998; Kuo and Mok, 1991; Kosugi *et al.*, 1994].

6. Feedback Scheduling

A scheduler that on-line adjusts task attributes in order to optimize control performance or QoS can be interpreted as a controller in itself. Important issues that then must be decided are what the right control signals, measurement signals, and setpoints are, what the control structure should be, and which process model that may be used. The block diagram of a feedback scheduler is shown in Figure 6. The goal of the scheduler is to keep the

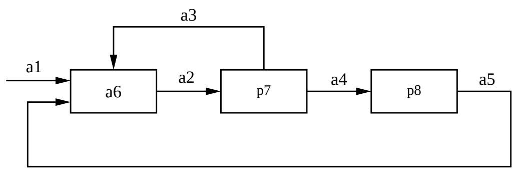

Figure 6: Feedback scheduler structure

CPU utilization, U, at a desired value. In order to do this it adjusts the sampling frequencies of the controller tasks. Feedforward is used to compensate for mode changes. The idea of using feedback in scheduling has to some extent been used previously in general purpose operating systems in the form of multi-level feedback queue scheduling [Kleinrock, 1970; Blevins and Ramamoorthy, 1976; Potier *et al.*, 1976]. However, this has mostly been done in an ad-hoc way.

So far relatively little has been done in the area of real-time feedback scheduling. In [Stankovic *et al.*, 1999] it is proposed to use a PID controller as an on-line scheduler under the notion of Feedback Control-EDF (FC-EDF). The measurement signal (the controlled variable) is the deadline miss ratio for the tasks and the control signal is the requested CPU utilization. Changes in the requested CPU utilization are effectuated by two mechanisms (actuators). An admission controller is used to control the flow of workload into the system and a service level controller is used to adjust the workload inside the system. The latter is done by changing between different versions of the tasks with different execution time demands.

For multimedia applications, feedback-based scheduling mechanisms that dynamically adjust the QoS level have been proposed in a few cases. In [Li and Nahrstedt, 1998] a general framework is proposed for controlling the application requests for system resources using the amount of allocated resources for feedback. It is shown that a PID controller can be used to bound tasks' resource usage in a stable and fair way. In [Abeni and Buttazzo, 1999] task models suitable for multimedia applications are defined. Two of these use PI control feedback to adjust the reserved fraction of CPU bandwidth.

A feedback scheduler for LQ-control is proposed in [Eker *et al.*, 2000]. The LQ cost is calculated as a function of the sampling interval. An optimization routine minimizes the global cost (the sum of cost function for of each controller) subject to a schedulability constraint. The minimization of the global cost function constitutes a nonlinear programming problem.

Control performance optimization in the context of task attribute adjustments needs cost functions that relate the sampling period with the control performance for each control loop, and cost functions that relate the input-output latency with the control performance. In addition to this it would also be beneficial if the control design methodology could provide cost functions for the jitter in sampling period and input-output latency. An interesting question is the general nature of these cost functions. For example, it is not always so that faster sampling gives better performance. The quadratic cost function as a function of the sampling interval for a non-inverted pendulum is shown in Figure 7. Note that the cost does not depend monotonously on the sampling period.

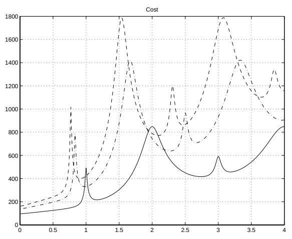

Figure 7: The cost $J_i(h)$ as a function of the sampling interval for a non-inverted pendulum. The plots shows the graphs for $\omega_0 = 3.1416$(full), 3.7699(dot-dashed), and 4.0841(dashed).

7. Anytime Controllers

A task attribute adjustment strategy can reduce the workload required by a controller task in two ways: by increasing the sampling period or by decreasing the maximum allowed execution time for the task. The latter would be a possibility for controllers that could be expressed on "anytime" form, i.e., controllers that would monotonously improve the control performance as a function of the allotted execution time. It is also possible to have control algorithms with optional parts that may be skipped.

Examples of anytime control algorithms can be found in model-predictive control. In the standard MPC form a quadratic optimization problem is solved every sample. The techniques employed are based on sequential unconstrained minimization. The calculations performed every sample are organized as a sequence of iterations. An interesting possibility is to calculate bounds on how much the objective function decreases for each new iteration.

8. Event-Based Sampling

An integrated control and scheduling system that dynamically adjusts sampling frequencies to compensate for workload changes or uncertainties can from a system point of view be seen as an event-based system. Much theoretical work on event-based systems was done in 1960–1980. The analysis is considerably harder than for time-based sampled system, primarily due to the fact that sampling is no longer a linear operation. The analysis is related to general work on discontinuous systems, e.g., [Utkin, 1987] and impulse control, see [Bensoussan and Lions, 1984]. There is also strong relationships to hybrid and switching control systems, e.g., [Morse, 1995].

9. Conclusions

Control and scheduling co-design is motivated for applications requiring high degrees of flexibility or when computing resources are limited. The field contains a number of interesting research issues, some of which have been discussed in this paper.

The work behind this paper has been funded by ARTES, NUTEK, and LUCAS - Center for Applied Software Research.

10. References

Abeni, L. and G. Buttazzo (1999): "Adaptive bandwidth reservation for multimedia computing." In *Proceedings of IEEE Real Time Computing Systems and Applications, Hong Kong*.

Årzén, K.-E., B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson, and L. Sha (1999): "Integrated control and scheduling." Report ISRN LUTFD2/TFRT--7586--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Bensoussan, A. and J.-L. Lions (1984): *Impulse control and quasi-variational inequalities*. Gauthier-Villars, Paris.

Blevins, P. and C. Ramamoorthy (1976): "Aspects of a dynamically adaptive operating system." *IEEE Trans Computers*, **25:7**.

Buttazzo, G., G. Lipari, and L. Abeni (1998): "Elastic task model for adaptive rate control." In *Proceedings of the IEEE Real-Time Systems Symposium*.

Eker, J., P. Hagander, and K.-E. Årzén (2000): "A feedback scheduler for real-time control tasks." *Accepted for publication in Control Engineering Practice*.

Joseph, M. and P. Pandya (1986): "Finding response times in a real-time system." *The Computer Journal*, **29:5**, pp. 390–395.

Klein, M. H., T. Ralya, B. Pollak, R. Obenza, and M. Gonzalez Härbour (1993): *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publisher.

Kleinrock, L. (1970): "A continuum of time-sharing scheduling algorithms." In *Proc. of AFIPS, SJCC*.

Kosugi, N., K. Takashio, and M. Tokoro (1994): "Modification and adjustment of real-time tasks with rate monotonic scheduling algorithm." In *Proceedings of the 2nd Workshop on Parallel and Distributed Systems*, pp. 98–103.

Kuo, T.-W. and A. Mok (1991): "Load adjustment in adaptive real-time systems." In *Proceedings of the 12th IEEE Real-Time Systems Symposium*.

Li, B. and K. Nahrstedt (1998): "A control theoretic model for quality of service adaptations." In *Proceedings of Sixth International Workshop on Quality of Service*.

Liu, C. L. and J. W. Layland (1973): "Scheduling algorithms for multiprogramming in a hard real-time environment." *Journal of the ACM*, **20:1**, pp. 40–61.

Morse, A. S. (1995): "Control using logic-based switching." In Isidori, Ed., *Trends in Control. A European Perspective*, pp. 69–113. Springer.

Nakajima, T. (1998): "Resource reservation for adaptive QoS mapping in real-time Mach." In *Proceedings of the Sixth International Workshop on Parallel and Distributed Real-Time Systems*.

Potier, D., E. Gelenbe, and J. Lenfant (1976): "Adaptive allocation of central processing unit quanta." *Journal of ACM*, **23:1**.

Ryu, M. and S. Hong (1998): "Toward automatic synthesis of schedulable real-time controllers." *Integrated Computer-Aided Engineering*, **5:3**, pp. 261–277.

Seto, D., J. Lehoczky, L. Sha, and K. Shin (1996): "On task schedulability in real-time control systems." In *Proceedings of the IEEE Real-Time Systems Symposium*.

Shin, K. G. and C. L. Meissner (1999): "Adaptation of control system performance by task reallocation and period modification." In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, pp. 29–36.

Stankovic, J. A., C. Lu, S. H. Son, and G. Tao (1999): "The case for feedback control real-time scheduling." In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*, pp. 11–20.

Utkin, V. I. (1987): "Discontinuous control systems: State of the art in theory and applications." In *Preprints 10th IFAC World Congress*. Munich, Germany.