Analyzing of the Effects of Missed Deadlines in Control Systems

Anton Cervin

Department of Automatic Control
Lund Institute of Technology
Box 118, SE 221 00 Lund, Sweden
anton@control.lth.se

Abstract

Design based on worst-case assumptions and hard deadlines can give low resource utilization. In control systems, this corresponds to slow sampling and low control performance. By allowing temporary overloads and overruns, we can increase both the average CPU utilization and the control performance.

As a first step toward real-time control design that tolerates overruns, this paper gives two examples which explore the impact of missed deadlines on control performance. In the first example, it is illustrated that control performance and scheduling performance are totally different things. In the second example, it is shown that in some cases we can actually gain control performance by setting tight deadlines which are then sometimes missed. The key parameters that link the areas of scheduling analysis and control analysis are the sampling interval, the input-output latency, and the jitter.

1. Background

Ever since the seminal paper on real-time scheduling theory, [Liu and Layland, 1973], computer-controlled systems have been used as the primary example of *hard real-time systems*. In a hard real-time system, the computer responds to periodic or non-periodic *events* by executing *tasks* that must finish within *hard deadlines*—or else, the system will fail. This description has shaped much of the real-time systems research during the past thirty years. It is questionable, however, whether this description really fits the large majority of computer-controlled systems.

Typically, a controller is implemented as a task that should execute periodically in the computer. In each period, the controller should sample the process output, execute the control algorithm, and send the new control signal, to the process. The controller is often designed using sampled-data control theory, see e.g. [Åström and Wittenmark, 1997], which assumes that the samples are taken at equidistant points in time.

The performance and stability of the closed-loop system depend not only on the control algorithm, but also on the actual implementation and run-time behavior of the controller in the computer system. The computing hardware, the execution-time properties of the control algorithm, the real-time operating system, the scheduling algorithm, and the possible network delays all introduce various amounts of delay and jitter in the control system. In the end, from a control perspective, it is

the *actual sampling period* (including jitter) and the *actual input-output latency* (including jitter) that influence the performance and stability of the closed-loop system. For further discussion on the sources and effects of timing variations, see [Törngren, 1998].

2. Where Do Deadlines Come From?

In the hard real-time scheduling literature, it is often unclear where the timing constraints—including deadlines—actually come from [Ramamritham, 1996]. The paper [Liu and Layland, 1973] is an exception to this, however: One of the assumptions clearly states that "Deadlines consist of run-ability constraints only—i.e. each task must be completed before the next request occurs." This implies that $D = T$ for all tasks. The connection to control theory is then made by mentioning that "Any control loops closed within the computer must be designed to allow at least an extra unit sample delay." Later research has extended the schedulability analysis to handle the cases $D < T$ and $D > T$ as well, see e.g. [Tindell *et al.*, 1994] and [Stankovic *et al.*, 1998], but the origin of the deadlines and their relation to control theory are rarely mentioned.

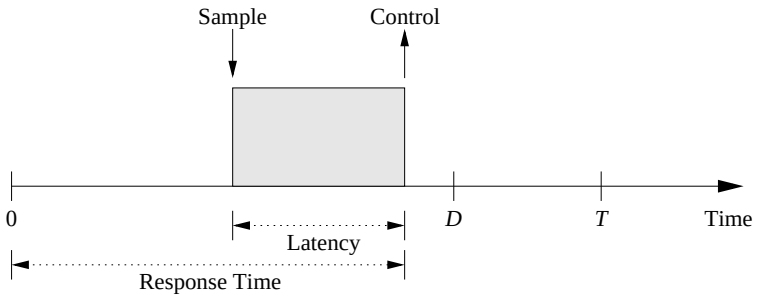

Figure 1 Illustration of the relationship between deadline (D), response time, and input-output latency for a control task. The latency is bounded by the response time, which in turn is bounded by the deadline. The task is assumed to be released at time zero.

From a control perspective, deadlines are primarily used to bound the input-output latency of the controllers. Figure 1 illustrates the relationship between deadline, response time, and input-output latency for a control task. Here, it is assumed that the A-D conversion takes place when the control task starts its execution, and that the D-A conversion takes place when the task completes its execution. A common alternative is to sample the process at the beginning of the period. Furthermore, it is common practice to divide the control algorithm into two parts—*Calculate Output* and *Update State*—and send out the control signal when the first part has completed.

For controlled plants with unstable dynamics, bounded latency is absolutely necessary to guarantee the stability of the closed-loop system. For all plants, bounded latency is necessary to guarantee some minimum level of control performance. Also, in general, the smaller the latency can be made, the better control performance and robustness can be achieved. Thus, typical deadlines assigned to control tasks ($D \leq T$) are often much tighter than what is dictated by pure stability considerations.

Deadlines may also be derived from other real-time constraints in the implementation. For instance, consider the case in Figure 2, where a dedicated, high-priority output task τ_{DA} is used to minimize the output jitter of control task τ_1, see [Locke,

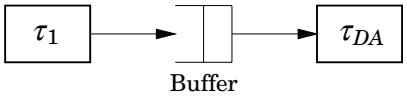

Figure 2 In fixed-priority scheduling with offsets, a high-priority, dedicated output task τ_{DA} can be used to minimize the output jitter of control task τ_1. This enforces a deadline on τ_1.

1992]. The tasks have the same period, but τ_{DA} is released with a fixed *offset* O relative to the release of τ_1. A deadline $D < O$ must be assigned to τ_1 to ensure that fresh output data will be available in the buffer when τ_{DA} starts its execution. See [Audsley *et al.*, 1993] for further details on offset scheduling. Different control structures, such as synchronized loops, cascaded loops, etc., may also enforce additional time constraints, see [Sandström, 1999].

Truly *hard* deadlines in real-time control systems are studied in [Shin *et al.*, 1985]. There, hard constraints on the controlled variables (e.g. physical constraints) are used to derive maximum allowable control latencies in different regions of the state-space. It is also noted that the hard deadline may be a random variable due to stochastic disturbances acting on the process. The approach is extended in [Shin and Kim, 1992], where the stability of the closed-loop system is also considered. In the examples given, the hard deadlines are typically found to be several times longer than the sampling interval. This may not be so surprising. As pointed out, the sampling period of a controller is not only chosen to satisfy Shannon's sampling theorem, but also to achieve the desired performance. Thus, a controller always has some degree of robustness against variations in the sampling interval due to, for instance, missed deadlines.

3. Analysis—A Two-Step Procedure

Investigating the effects of missed deadlines in control systems is a two-step procedure, see Figure 3. Since control performance is a function of the sampling period, the input-output delay, and the jitter, these, possibly stochastic, variables are first obtained by scheduling analysis. The resulting control performance is then found by control analysis. In the general case, both steps are very difficult to carry out, and some approximations may be necessary.

Previous work [Eker and Cervin, 1999; Palopoli *et al.*, 2000] has relied on co-simulations of the real-time kernel and the continuous dynamics to determine the performance of the controllers in the real-time system. While this approach can provide important insight, it is also very time-consuming, especially when the performance should be evaluated for a wide range of parameters. Also, simulations cannot be used to prove stability or a minimum level of performance.

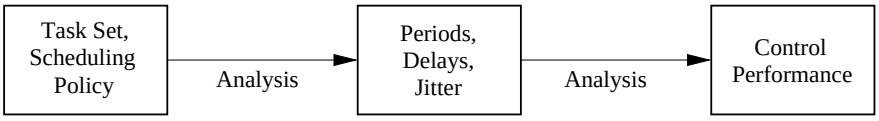

Figure 3 Analysis of the effects of missed deadlines on the control performance is a two-step procedure: scheduling analysis followed by control analysis.

3.1 Scheduling Analysis

The effect of missed deadlines depends not only on the scheduling policy used, but also on the details in the real-time operating system. Often, deadlines are derived and used in the scheduling design phase, but they are not necessarily enforced during run-time. Hence, an overrun in one task may or may not disrupt the timing of the other tasks.

Consider for instance a set of periodic control tasks that are described by the following pseudo-code:

```
t = CurrentTime;
LOOP
  AD-Conversion;
  ControlAlgorithm;
  DA-Conversion;
  t = t + T;
  WaitUntil(t);
END
```

Here, the only time constraint enforced during run-time is, that the task will never be released prior to its nominal release time. If the control task does not meet its design specifications during run-time, e.g. if the execution time of the control algorithm is larger than the predicted worst-case execution time, other task instances will be delayed.

The scheduling analysis methods available today do not allow for derivation of the distribution of the input-output delay and the jitter, except for in very simple cases. One solution could be to rely on simulations to find the approximate distributions and on analysis to find the best-case and the worst-case values.

3.2 Deterministic Control Analysis

If the pattern of sampling periods and delays is deterministic, the control analysis is straight-forward. A sampled plant, including a latency τ, where $0 < \tau \leq T$, can be written as (see e.g. [Åström and Wittenmark, 1997])

$$x(k+1) = \Phi x(k+1) + \Gamma_0 u(k) + \Gamma_1 u(k-1).$$

A controller can be written on general state-space form:

$$\begin{aligned} x_r(k+1) &= \Phi_r x_r(k) + \Gamma_r y(k) \\ u(k) &= C_r x_r(k) + D_r y(k) \end{aligned}$$

Forming the closed-loop system, we get

$$\begin{bmatrix} x(k+1) \\ u(k) \\ x_r(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi + \Gamma_0 D_r C & \Gamma_1 & \Gamma_0 C_r \\ D_r C & 0 & C_r \\ \Gamma_r C & 0 & \Phi_r \end{bmatrix}}_{\Phi_{cl}} \begin{bmatrix} x(k) \\ u(k-1) \\ x_r(k) \end{bmatrix}$$

For a constant Φ_{cl} matrix, the closed-loop system is stable if and only if $\bar{\sigma}(\Phi_{cl}) < 1$, i.e. iff the spectral radius of Φ_{cl} is less than one. When Φ_{cl} varies according a repeating pattern, $\Phi_{cl1}, \Phi_{cl2}, \ldots, \Phi_{cln}$, the closed-loop system is stable if and only if $\bar{\sigma}(\Phi_{cl1}\Phi_{cl2}\cdots\Phi_{cln}) < 1$.

3.3 Stochastic Control Analysis

When the actual sampling periods and delays are stochastic (as they are in the general case, due to random execution times, etc.), the analysis is considerably harder. One approach is to numerically evaluate a linear-quadratic cost function of the form

$$J = \lim_{T\to\infty} \frac{1}{T} \mathrm{E}\left\{ \int_0^T (x^T Q_1 x + u^T Q_2 u) dt \right\},$$

where x is the state vector, u is the control signal, and Q_1 and Q_2 are weighting matrices. The cost function measures the variance of different signals in the control system. With proper choices of weighting matrices, stability can be concluded if $J < \infty$. Furthermore, J is a measure of the performance of the controller.

Theory developed in [Nilsson, 1998; Lincoln and Bernhardsson, 2000] permits evaluation of the cost J of any linear controller for stochastic, independent delays from sampling to actuation. Aborted computations (modeled as infinite delays) can also be accounted for. Work in progress aims at evaluating the cost also in the presence of sampling jitter.

4. Two Examples

Two examples are given that demonstrate the type of analysis that can be carried out today. The first example assumes fixed execution times and investigates stability under overload conditions under different scheduling policies. The second example assumes a random execution time and demonstrates how a soft deadline can be assigned to maximize the control performance.

4.1 Example 1: Overloaded System

Consider a set-up where two mechanical servos,

$$G_{1,2}(s) = \frac{1000}{s(s+1)},$$

should be controlled by two PD (proportional-derivative) controllers that execute as two tasks in a computer. The controllers have been tuned to give good performance in simulations, and suitable sampling periods have been found to be $T_1 = 6$ ms and $T_2 = 4$ ms. It is assumed that $D = T$ for both tasks. Unfortunately, the execution time of the control algorithm is longer than anticipated, $C = 3$ ms. The requested CPU utilization is

$$U = \frac{C}{T_1} + \frac{C}{T_2} = 1.25.$$

Since $U > 1$, the system is overloaded, and no scheduling algorithm can guarantee that all deadlines will be met. This says very little about the performance of the controllers, however.

We analyze stability of the controllers under rate-monotonic (RM) and earliest-deadline-first (EDF) scheduling. Assuming that the tasks are implemented according to the pseudo-code in the previous section, we first derive the actual sampling periods and the actual input-output latency of the controllers. We then apply stability analysis to the controllers.

Rate-Monotonic Scheduling The performance under rate-monotonic (RM) scheduling is investigated first. Task 2 is given priority over Task 1 since $T_2 < T_1$. The first part of the resulting schedule, when both tasks are released at time zero, is shown in Figure 4. Because of preemption, Task 1 misses all its deadlines. The

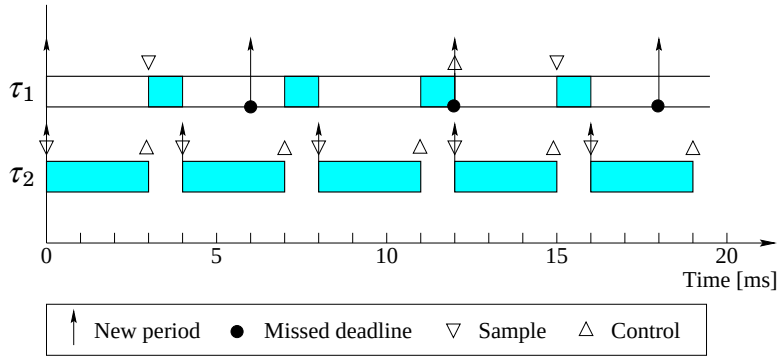

Figure 4 The first part of the schedule under RM scheduling. Task 1 misses all its deadlines.

regularity of the schedule makes it easy to determine the actual sampling period, $\overline{T}$, and the actual input-output latency, $\overline{L}$, of the controllers. By inspection, they are found to be constant (jitter-free) and equal to $\overline{T}_1 = 12$ ms, $\overline{L}_1 = 9$ ms, $\overline{T}_2 = 4$ ms, and $\overline{L}_2 = 3$ ms.

Applying the deterministic stability analysis to Controller 1, we find that $\bar{\sigma}(\Phi_{cl}) > 1$, so the controller is unstable. For Controller 2, we have $\bar{\sigma}(\Phi_{cl}) < 1$, so the controller is stable.

A simulation of the complete set-up is shown in Figure 5. The performance of Controller 2 is good, as expected. Controller 1, however, loses stability due to the long sampling period and the long latency, which the controller has not been designed for.

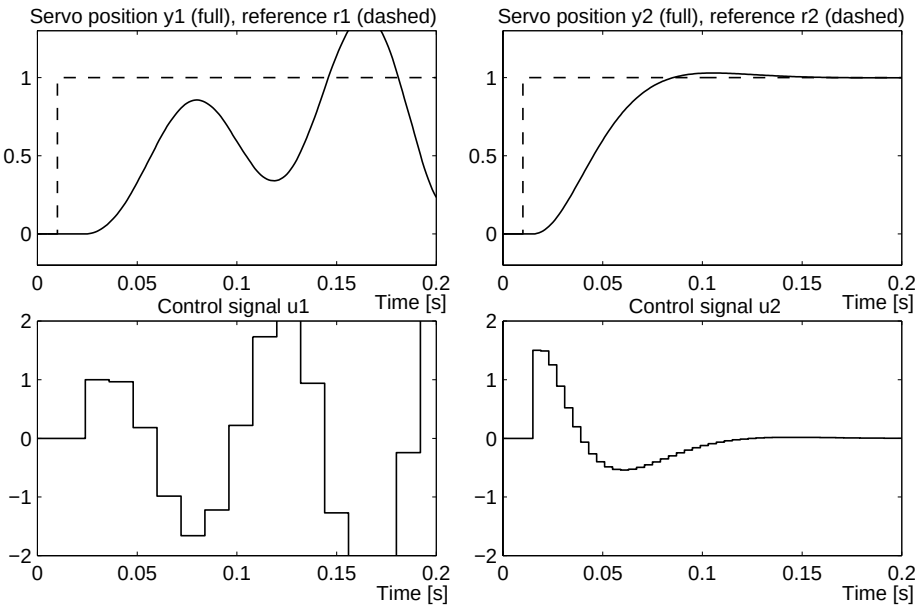

Figure 5 Step responses under RM scheduling for Controller 1 (left) and Controller 2 (right). Controller 1 loses stability due to the long actual sampling period and the long actual input-output latency.

Earliest-Deadline-First Scheduling Next, the performance under earliest-deadline-first (EDF) scheduling is investigated. The tasks execute in the order of their absolute deadlines. Ties can be broken arbitrarily—assume that Task 1 gets to execute before Task 2 in those cases. The first part of the resulting schedule, when both tasks are released at time zero, is shown in Figure 6. After an initial

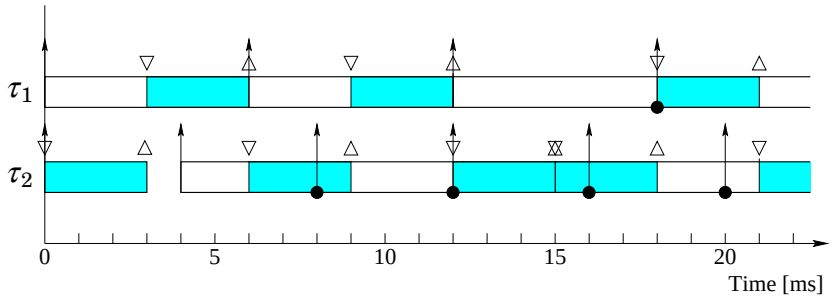

Figure 6 The first part of the schedule under EDF scheduling. After an initial transient, all deadlines are missed.

transient, *all* deadlines are missed. This is due to the well-known *domino effect*. There is no preemption in the resulting schedule so the actual input-output latency is always equal to $\overline{L} = 3$ ms for both controllers. The actual sampling periods are no longer constant, but they exhibit periodic behavior. By inspection, $\overline{T}_1$ is found to exhibit the cycle $\{6, 9\}$ ms while $\overline{T}_2$ exhibits the cycle $\{6, 6, 3\}$ ms.

In this case, we are switching between different sampling intervals, which gives different Φ_{cl} matrices. Applying the stability analysis, we find that for Controller 1, $\bar{\sigma}(\Phi_{cl1}\Phi_{cl2}) < 1$, so this controller is stable. For Controller 2, we have $\bar{\sigma}(\Phi_{cl1}\Phi_{cl1}\Phi_{cl2}) < 1$. so this controller is also stable.

A simulation of the set-up is shown in Figure 7. The systems are stable and the performance is satisfactory for both controllers, despite the missed deadlines. The jitter due to the varying sampling periods is clearly visible in the control signals.

The example shows that control performance and scheduling performance are completely different things. It also displays that EDF scheduling distributes the available computing resources more evenly in overload situations than RM scheduling does.

4.2 Example 2: Optimal Deadline Assignment

Consider a set-up where an inverted pendulum,

$$G(s) = \frac{1}{s^2 - 1},$$

should be controlled by a linear-quadratic controller. The sampling period is chosen as $T = 0.2$ s. The execution-time of the control task is described by the probability distribution function shown in Figure 8. The long tail of the distribution could for instance be the result of hardware interrupts, data dependencies, or cache misses.

An output buffer (a high-priority task) is used to minimize output jitter, see Figure 2. The output task executes with a fixed offset L compared to the controller. This enforces the deadline $D = L$ on the control task. If the output value is put in the buffer after the deadline has expired, the control signal will be lost.

The controller is designed taking into account a fixed delay L and minimizing the cost function

$$J = \lim_{T \to \infty} \frac{1}{T} \mathrm{E} \left\{ \int_0^T (y^2(t) + u^2(t)) dt \right\}. \tag{1}$$

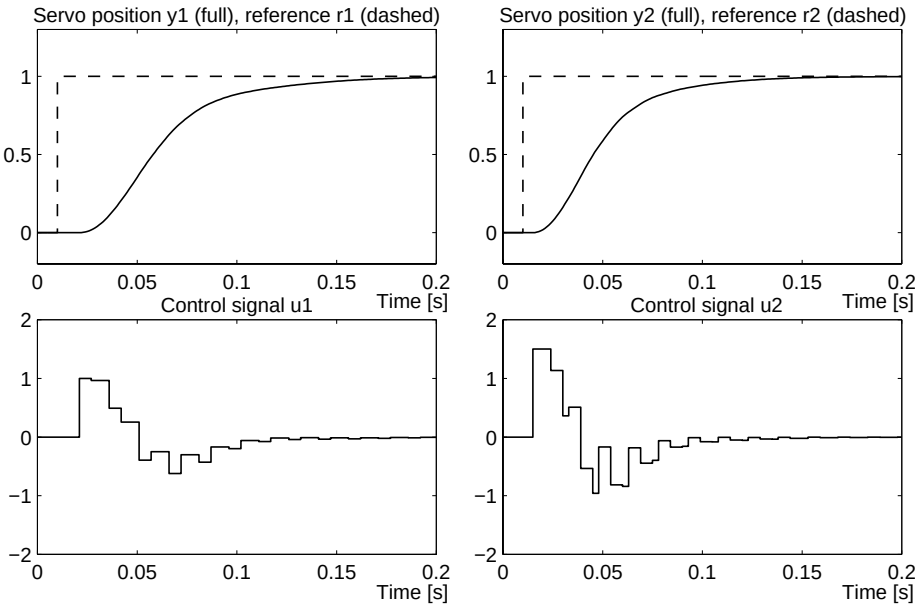

Figure 7 Step responses under EDF scheduling for Controller 1 (left) and Controller 2 (right). Both systems are stable despite the fact that, after an initial transient, all deadlines are missed. The sampling interval jitter is clearly visible in the control signals.

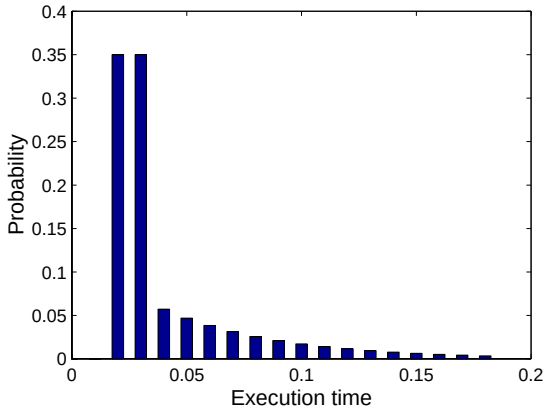

Figure 8 The probability distribution of the execution time of the pendulum controller.

The problem is to find the optimal value of L. If we choose $L = T$, all computations will finish in time, but we will also have a long delay, which is bad from a control performance perspective. If we choose a value $L < T$, the delay may be shorter on average, but some computations will also be aborted. If we choose $L = 0$, the task will never finish before its deadline and, without any control signals, the system will of course be unstable.

The optimal deadline assignment can be found by computing the cost J for different values of the deadline L. The plot is shown in Figure 9. In this case the optimal deadline assignment turns out to be $L = 0.08$, in which case about 10% of the deadlines are missed. The result depends strongly on the execution-time distribution, however. The penalty for aborted computations is quite high, so a more uniform execution-time distribution would push the deadline towards $L = T$. Still, the example shows that, even in the case of aborted computations, we can gain control performance by allowing some percentage of the deadlines to be missed.

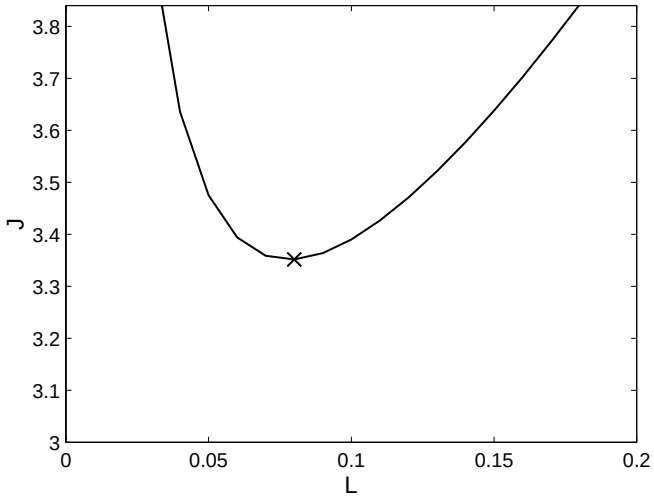

Figure 9 The cost J as a function of the deadline L for the inverted pendulum controller. The optimal deadline assignment is $L = 0.08$.

5. Conclusions

Relaxing the nominal requirements on hard deadlines in real-time control systems is motivated from a resource utilization viewpoint. Modern computing hardware tends to be optimized for high average-case performance rather than guaranteed worst-case performance. Also, many control algorithms display considerable variations in their execution-time demands. Taken together, scheduling based on worst-case execution times and hard deadlines may be infeasible for a large set of control applications.

Exploring the consequences of overruns is necessary in order to make the correct co-design trade-offs. For instance, it may be beneficial to decrease the average period of a controller, and allow it to miss a few deadlines, as long as the worst-case latency does not exceed a certain bound.

In the suggested analysis, the first step is to determine what the actual sampling period and the actual input-output latency will be for the different control tasks. These quantities will be random variables in the general case, depending on the task execution-time distributions, the scheduling algorithm, the mechanisms in the real-time operating system, the controller structure, etc. The next step is to determine what the impact on the control performance will be. Future work will extend the stochastic control performance analysis to handle sampling jitter.

6. References

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*, third edition. Prentice Hall.

Audsley, N., K. Tindell, and A. Burns (1993): "The end of the line for static cyclic scheduling." In *Proceedings of 5th Euromicro Workshop on Real-Time Systems*.

Eker, J. and A. Cervin (1999): "A Matlab toolbox for real-time and control systems co-design." In *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications*, pp. 320–327. Hong Kong, P.R. China.

Lincoln, B. and B. Bernhardsson (2000): "Optimal control over networks with long random delays." In *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems*.

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.

Locke, C. D. (1992): "Software architecture for hard real-time applications: Cyclic vs. fixed priority executives." *Real-Time Systems*, **4**, pp. 37–53.

Nilsson, J. (1998): *Real-Time Control Systems with Delays*. PhD thesis ISRN LUTFD2/TFRT--1049--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Palopoli, L., L. Abeni, and G. Buttazzo (2000): "Real-time control system analysis: An integrated approach." In *Proceedings of the IEEE Real-Time Systems Symposium, Orlando, Florida*.

Ramamritham, K. (1996): "Where do time constraints come from and where do they go?" *International Journal of Database Management*, **7:2**.

Sandström, K. (1999): *Modeling and Scheduling of Control Systems*. Licentiate thesis ISRN KTH/MMK/R--99/5--SE, Mechatronics Laboratory, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden.

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

Shin, K. G., C. M. Krishna, and Y.-H. Lee (1985): "A unified method for evauating real-time computer controllers and its applications." *IEEE Transactions on Automatic Control*, **30:4**, pp. 357–366.

Stankovic, J., M. Spuri, K. Ramamritham, and G. Buttazzo (1998): *Deadline Scheduling for Real-Time Systems*. Kluwer Academic Publishers.

Tindell, K., A. Burns, and A. Wellings (1994): "An extendible approach for analyzing fixed priority hard real-time tasks." *Real-Time Systems*, **6:2**, pp. 133–151.

Törngren, M. (1998): "Fundamentals of implementing real-time control applications in distributed computer systems." *Real-time systems*, **14:3**.