Resource-Constrained Embedded Control and Computing Systems

Henriksson, Dan

2006

Document Version:
Publisher's PDF, also known as Version of record

Resource-Constrained Embedded Control and Computing Systems

Resource-Constrained Embedded Control and Computing Systems

Dan Henriksson

Department of Automatic Control
Lund University
Lund, 2006

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1074--SE

Printed in Sweden by Media-Tryck.
Lund 2006

Abstract

This thesis deals with methods for handling resource constraints in embedded control systems and real-time computing systems. By dynamic feedback-based resource scheduling it is possible to achieve adaptability and increased performance for these systems.

A feedback scheduling strategy is presented, which uses feedback from plant states to distribute computing resources optimally among a set of controller tasks. Linear-quadratic controllers are analyzed, and expressions relating the expected cost to the sampling period and the plant state are derived and used for on-line sample-rate adjustments.

A flexible implementation of model predictive control (MPC) tasks is described. A termination criterion is derived that, unlike traditional MPC, takes the effects of computational delay into account in the optimization. A scheduling scheme is also described, where the MPC cost functions being minimized are used as dynamic task priorities for a set of MPC tasks.

A method for optimizing the use of computational resources in a multi-camera-based positioning system is studied. The covariance of the estimation error is minimized, while meeting computation time constraints.

A novel predictor for delay control in server systems is introduced. The predictor uses instantaneous measurements of queue length and arrival times and is continuously updated as new requests arrive according to a receding horizon principle. The predictor is evaluated in simulation and by experiments on an Apache web server.

The MATLAB/Simulink-based simulator TrueTime is presented. TrueTime is a codesign tool that facilitates simulation of distributed real-time control systems. TrueTime also supports simulation of wireless communication and resource constraints associated with wireless sensor/actuator networks.

Acknowledgments

First of all, I would like to thank my supervisor Karl-Erik Årzén for his support, inspiration, and insightful ideas throughout my work. I am also very grateful for the work he and the other professors at the department are doing to acquire funding for myself and other graduate students. Karl-Erik also shares my interest in football, yet he supports the wrong team. I have always wondered why he kept his football interest so well hidden during my first year at the department. Maybe it had to do with the fact that Malmö was playing in the second division at the time, while Helsingborg beat Inter (compare FC Thun) to qualify for the Champions League.

A special thanks also to Anton Cervin, who originally got me into the field of real-time systems, and whose importance for the completion of this thesis can not be overstated. Anton is also gratefully acknowledged for all the money he has bluffed away during our poker sessions.

I have had the great pleasure to work together with a number of my other PhD student colleagues during my years at the department. Johan Åkesson was a great source of inspiration during the MPC work. Tomas Olsson knows as much about image-based tracking as he is hopeless when it comes to football knowledge. Martin Andersson is the chief architect behind the recent TrueTime developments. Thanks everyone!

The summer of 2003, I had the pleasure to spend three months at the University of Virginia working with Tarek Abdelzaher. This was a very rewarding visit, and I hope for continued collaboration in the future.

I would like to thank all the people at the department for providing such a great working atmosphere! A special thanks to the backbone of the department: the secretaries, Eva, Agneta, and Britt-Marie, and the technical staff, Leif, Anders, and Rolf.

My fellow innebandy friends in PiHHP also need to be mentioned. I respect your commitment, but to be frank, you are not very good. We need to get better players to the team. It is not possible for me to practically play

alone against four opponents in every match. And where is that Mercedes with private driver I was promised when I started playing for the team?

The work has been sponsored by the Swedish Foundation for Strategic Research via the program FLEXCON, the EU Network of Excellence ARTIST2, the Swedish Research Council, and by LUCAS — the VINNOVA-funded Center for Applied Software Research. The development of TrueTime has also been funded by ARTES (A network for Real-Time research and graduate Education in Sweden). ARTES is also gratefully acknowledged for the travel grant that enabled my research visit to University of Virginia.

My friends and family have been important for their help and encouragement, and for making every-day life so easy and enjoyable.

Finally, I would like to thank Elinore for her love and support, and for providing me with another thesis to read when I got bored with my own.

Dan

Contents

Preface

Motivation

Embedded microcomputers have become an integral part of most modern engineering applications, and real-time control systems constitute an important subclass of these embedded systems. One prominent example is automotive systems, which contain many embedded ECUs (electronic control units) used for various feedback control tasks, including engine performance control, anti-lock braking, active stability control, exhaust emission reduction, and cruise control.

Modern applications have strong requirements on resource optimization, since reduced time-to-market and low manufacturing costs often are decisive factors in the development process. As a consequence, processing power often remains constant over long periods of time, while the processing cost is constantly reduced. At the same time, the systems grow continuously more complex and an increasing amount of functionality is added to meet customer demands on performance. In addition, there is a strong trend to use commercially available information technology and commercial-off-the-shelf components also for embedded systems.

The resource constraints associated with embedded systems, combined with non-optimized software components used for their implementation, introduce non-determinism in the real-time system. For control systems this is of particular concern, since timing variations induced by the implementation degrade the control performance. Adding to the non-determinism is the fact that many embedded control systems are implemented using distributed architectures, where the sensors, actuators, and control functionality are located on different nodes in a communication network.

In highly safety-critical applications, such as nuclear power plants or fly-by-wire systems, the main objective in the software design is to max-

imize the determinism in order to guarantee predictable behavior. This motivates the use of static design methodologies, including static cyclic scheduling and time-triggered communication. For the majority of control systems, however, the drawbacks of using a static design outweigh the benefits. While the static techniques increase the predictability and allow for off-line guarantees, they are less resource-efficient and limit the possibilities for dynamic modifications.

Applications of real-time computing have also gradually evolved from closed embedded systems to complex, distributed, heterogeneous platforms operating in unpredictable poorly modeled environments, such as the Internet. One example in this category is modern web servers, for which load and resource capacities are very difficult to predict. The use of control-based approaches for modeling, analysis, and design is a promising foundation to handle this uncertainty and obtain performance guarantees for these systems.

The work in this thesis deals with methods to optimize performance of control systems implemented on resource-constrained platforms and methods to cope with uncertainties in real-time computing systems.

For resource-constrained embedded control systems, codesign of the control and the real-time system is key. This makes it possible for constraints from the target platform to be taken into consideration in the controller design and also allows for implementation techniques and scheduling schemes tailored towards control systems to be developed.

One promising approach is to use feedback for resource allocation. By treating the control performance as a quality-of-service parameter that should be maximized, resources may be dynamically allocated to the controller tasks based on measurements of actual resource consumption. As opposed to the static design methodologies, this approach promises higher control performance under given resource constraints and the possibility to dynamically adapt to changing load conditions. The available resources can be used more efficiently, allowing the use of cheaper hardware or the addition of new functionality in existing products.

Integrated control and real-time system design requires simulation tools tailored for these systems. Few tools exist that merge controller design and dynamic-system simulation with simulation of implementation issues, such as real-time scheduling and network communication. To remedy this situation, a tool that facilitates real-time control system development within the MATLAB/Simulink framework has been developed.

Contributions of the Thesis

The thesis consists of two introductory chapters and six papers. This section describes the contents of the introductory chapters and the contribution of each paper.

Chapter 1 – Background

The background chapter is outlined as follows. Section 1.1 treats issues related to control system implementation, including the basics of sampled-data control, real-time scheduling and communication, and controller timing. Motivated by these issues, Section 1.2 discusses various approaches to codesign of real-time control systems. The concept of feedback scheduling is introduced in Section 1.3. The main objective of feedback scheduling is to optimize the overall performance of control loops implemented in resource-constrained systems. Details are given in Papers I–III. An introduction to control of server systems is given in Section 1.4, delay control of web server requests being the topic of Paper IV. Finally, Section 1.5 gives an introduction to simulation tools for control and computer systems. The TrueTime simulator is introduced and two examples are given to illustrate the tool. The TrueTime simulator is the topic of Papers V and VI, and has also been used for evaluation of the methods proposed in Papers I–IV.

Chapter 2 – Feedback Scheduling for Cooperative Robots

This chapter describes a larger feedback scheduling scenario, which has been developed as a demonstrator within a national research program. The application consists of two cooperating industrial robots balancing a ball on a beam attached to the end-effectors of the robots. The example is intended to demonstrate the application of feedback scheduling in a complex setup and how it can be simulated using the TrueTime simulator.

The feedback scheduler modifies the sampling periods of the robot joint controller tasks based on measurements of actual resource consumption and mode changes. A heuristic feedback scheduling extension is described, which incorporates the fact that the robot joint control loops are coupled, i.e., that the performance of each individual joint is more or less important for the overall performance of the robot.

Paper I

Henriksson, D. and A. Cervin (2005): "Optimal on-line sampling period assignment for real-time control tasks based on plant state information." In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*. Seville, Spain.

A feedback scheduling strategy is presented, which uses feedback from plant states to distribute computing resources optimally among a set of controller tasks.

Contributions Linear-quadratic controllers are analyzed, and expressions relating the expected cost to the sampling period and the plant state are derived and used for on-line sample-rate adjustments. In the case of minimum-variance control of multiple integrator processes, an exact expression for the optimal sampling periods is obtained. In the general case, the cost functions are linearized to facilitate on-line optimization. The approach is exemplified on a set of controllers for first-order systems.

The paper was developed in close collaboration with Anton Cervin.

Paper II

Henriksson, D. and J. Åkesson (2004): "Flexible implementation of model predictive control using sub-optimal solutions." Technical Report ISRN LUTFD2/TFRT--7610--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

A flexible approach to real-time implementation and scheduling of model predictive controllers is presented.

Contributions The control signal in model predictive control (MPC) is computed by on-line optimization of a cost function in every sample. The iterative nature of the control algorithm allows for a trade-off between computational delay and the quality of the obtained control signal. The trade-off is quantified by a delay-dependent termination criterion rendering a sub-optimal, yet stabilizing, MPC formulation. Unlike traditional MPC, the effects of computational delay are taken into consideration in the optimization. A dynamic scheduling policy based on the MPC cost functions is also described.

This paper represents joint work with Johan Åkesson. Åkesson provided the tools used for implementation and analysis of the MPC controller. Henriksson conducted the real-time simulations, using the TrueTime simulator. The delay compensation and dynamic scheduling schemes were developed in close collaboration between the authors.

Related Publications The paper is an extension of the following conference papers

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002): "Feedback scheduling of model predictive controllers." In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium*. San Jose, CA, USA.

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002): "On dynamic real-time scheduling of model predictive controllers." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV, USA.

Paper III

Henriksson, D. and T. Olsson (2004): "Maximizing the use of computational resources in multi-camera feedback control." In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*. Toronto, Canada.

A method for optimizing the use of computational resources in a multi-camera-based positioning system is presented.

Contributions The proposed method exploits the trade-off in a tracking algorithm between computing time and the accuracy of the produced position/orientation estimates. A simplified equation for the covariance of the estimation error is calculated and an efficient algorithm for selection of a suitable subset of the available cameras is presented. The suggested strategy is compared with heuristic algorithms and evaluated in simulations capturing the real-time properties of the tracking algorithm and the effects of the timing on the performance of vision-based control systems.

This paper represents joint work with Tomas Olsson. Olsson has developed the tracking algorithm and provided the tools used for simulation of the vision system. Henriksson connected the vision simulation to the TrueTime tool to conduct the real-time simulations. The resource allocation strategy and the paper were developed in close collaboration between the authors.

Paper IV

Henriksson, D., Y. Lu, and T. Abdelzaher (2004): "Improved prediction for web server delay control." In *Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS 04)*. Catania, Sicily, Italy.

A delay predictor for QoS control in web servers is presented.

Contributions Prediction using queuing theory applies only to long-term averages and is therefore insensitive to sudden load changes. Instead, the proposed predictor uses instantaneous measurements of arrival times and queue length to predict future delays of the requests in the server. The proposed strategy is evaluated in simulation and by experiments on an Apache web server. It is shown that the new predictor performs better than previous approaches based on queuing theory.

This paper represents joint work with Ying Lu and Tarek Abdelzaher and was performed during a research stay at University of Virginia. Abdelzaher and Henriksson developed the delay predictor. Henriksson conducted the simulations, whereas Lu conducted the experiments on the real test-bed. The paper was written in close collaboration between all authors.

Related Publications

Abdelzaher, T., Y. Lu, R. Zhang, and D. Henriksson (2004): "Practical application of control theory to web services." In *Proceedings of the American Control Conference*. Boston, MA, USA.

Paper V

Henriksson, D., A. Cervin, and K.-E. Årzén (2003): "TrueTime: Real-time control system simulation with MATLAB/Simulink." In *Proceedings of the Nordic MATLAB Conference*. Copenhagen, Denmark.

The codesign tool TrueTime, for simulation of distributed real-time control systems, is presented.

Contributions The simulator is based on MATLAB/Simulink and allows for co-simulation of controller task execution in real-time kernels, network communication, and continuous-time plant dynamics.

The simulator represents joint work with Anton Cervin and has been designed in close collaboration between the authors. Henriksson has implemented the major parts of the TrueTime real-time kernel block, whereas Cervin has implemented the major parts of the network block. The publications have been written in close collaboration between the authors.

Related Publications

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén (2003): "How does control timing affect performance?" *IEEE Control Systems Magazine*, **23:3**, pp. 16–30.

Henriksson, D., A. Cervin, and K.-E. Årzén (2002): "TrueTime: Simulation of control loops under shared computer resources." In *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Henriksson, D. and A. Cervin (2003): "TrueTime 1.1—Reference manual." Technical Report ISRN LUTFD2/TFRT--7605--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henriksson, D., O. Redell, J. El-Khoury, M. Törngren, and K.-E. Årzén (2005): "Tools for real-time control systems co-design—a survey." Technical Report ISRN LUTFD2/TFRT--7612--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Paper VI

Andersson, M., D. Henriksson, A. Cervin, and K.-E. Årzén (2005): "Simulation of wireless networked control systems." In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*. Seville, Spain.

Extensions to the TrueTime tool aimed at wireless networked control systems are presented.

Contributions Support for simulation of wireless communication protocols and propagation of radio signals is added together with new means to simulate battery-powered nodes and local clocks. The additions open up a wide range of new application types for simulation, such as teams of collaborating or competing mobile robots interacting with their environment. Another example could be sensor networks with mobile or stationary nodes communicating via wireless ad-hoc networks and, through a gateway node, to back-end servers using wired networks.

The implementation of the new TrueTime features has been performed by Martin Andersson. The simulation models and the paper were developed in close collaboration between all authors.

Other Publications

These are some related publications that were chosen not to be included in the thesis. Some of the material presented in Chapter 1 is taken from these publications.

Årzén, K.-E., A. Cervin, and D. Henriksson (2003): "Resource-constrained embedded control systems: Possibilities and research issues." In *Proceedings of CERTS'03 — Co-design of Embedded Real-Time Systems Workshop*. Porto, Portugal.

Årzén, K.-E., A. Cervin, and D. Henriksson (2005): "Implementation-aware embedded control systems." In Hristu-Varsakelis and Levine, Eds., *Handbook of Networked and Embedded Control Systems*.

Henriksson, D. (2003): "Flexible scheduling methods and tools for real-time control systems." Licentiate thesis ISRN LUTFD2/TFRT--3233--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henriksson, D. and A. Cervin (2004): "Multirate feedback control using the TinyRealTime kernel." In *Proceedings of the 19th International Symposium on Computer and Information Sciences*. Antalya, Turkey.

Nilsson, R. and D. Henriksson (2005): "Test case generation for flexible real-time control systems." In *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*. Catania, Sicily, Italy.

1

Background

1.1 Issues in Control System Implementation

The advances in microelectronics, and the increasing speed of microprocessors during the last 30 years, have led to a situation where today almost all control algorithms are realized by computers. The computer-based control systems usually rely on modern real-time technologies for their implementation, including real-time operating systems and kernels, real-time programming languages, and real-time communication networks. However, the rapid evolution of electronics has led to a situation where implementation issues, such as real-time scheduling, are often dismissed as non-problems.

In the early days of computer control, implementation issues related to the computing hardware were well-known problems among control engineers [Hanselmann, 1987]. Memory was scarce and the computational performance and accuracy were limited. This forced the system designers to have a tight coupling between the control design on one hand, and its real-time implementation on the other.

For embedded control systems, the resource constraints are still present. Today, these constraints mainly stem from economic considerations and increased complexity. Tight product demands require the hardware cost proportion to be minimized. At the same time, the evolution of software makes it possible to realize more and more control-related functionality in different products. Hence, resource scheduling and its effect on performance is an important issue for embedded control systems.

Networked solutions in control system implementation are also becoming more and more common. Introduction of distributed sensing, actuation, and control computation has a lot of advantages, such as increased computational power, reduced cabling costs, and improved data integrity.

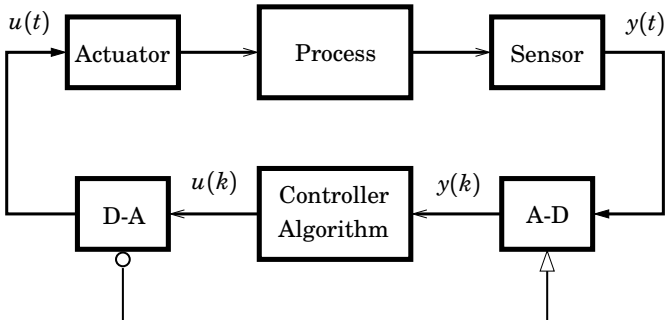

Figure 1.1 Schematic diagram of a computer-controlled system.

On the other hand, networking also increases system complexity and introduces the communication bandwidth as yet another shared resource. Resource constraints can also occur in more non-traditional applications, such as wireless distributed systems, where power and communication channel constraints affect the control system design.

In the control community today, timing effects caused by the computing and hardware platform are generally not taken into account in the controller design. The basic assumptions made in the control community do not hold when implemented in resource-constrained systems, resulting in timing non-determinism and degraded control performance.

This section briefly introduces concepts of computer-based control and reviews implementation-related issues, including real-time scheduling and schedulability analysis, real-time communication, and controller timing variations and their effect on the control system performance.

Traditional Sampled-Data Control

The basic structure of a computer-based control system is shown in Figure 1.1. The physical process under control is associated with a number of sensors and actuators used by the controller. The controller execution consists of three distinct operations executed in sequence: *reading of the sensors (A-D conversion)*, *control signal computation*, and *delivery of the control action (D-A conversion)*.

In the implementation, these operations are triggered by events, with the predominant event trigger being system time. The continuous process output, $y(t)$, is most often sampled at regular time intervals and converted to digital form, $y(k)$, by an A-D converter. The control algorithm reads the sampled process output and computes a control signal, $u(k)$, that is converted back to analog form, $u(t)$, by a D-A converter. The D-A conversion

is usually performed by keeping the output constant between conversions, so called *zero-order hold*.

In the ideal situation, the input and output operations (the A-D and D-A conversions) are periodic and perfectly synchronized, corresponding to fixed sampling intervals and a constant input-output delay. This is referred to as conventional sampling. However, many other types of sampling exist and can be analyzed from a control performance perspective. Examples include multi-rate sampling, multi-order sampling, non-synchronous sampling, and random sampling [Kalman and Bertram, 1959]. Due to resource constraints and implementation effects, most control systems are actually randomly sampled.

The control algorithm is often designed using sampled-data control methods, e.g., [Åström and Wittenmark, 1997], with various degrees of complexity. Common for all design methods, however, is that they assume conventional sampling at equidistant time intervals and zero or constant latency between the sampling and control signal actuation.

The sampling time is chosen in relation to the dynamics of the controlled process and the required bandwidth of the closed-loop system. One rule-of-thumb [Åström and Wittenmark, 1997] for sampling period selection in a digital control system is

$$0.2 \leq \omega_b h \leq 0.6, \tag{1.1}$$

where h is the sampling interval and ω_b is the bandwidth of the closed-loop system.

Periodic control loops are usually implemented on top of a real-time operating system (RTOS) or using a real-time kernel that gives support for task decomposition and provides real-time primitives. The standard implementation of a control task is given by the pseudo code in Listing 1.1.

Listing 1.1 A standard implementation of a periodic control loop.

```
t = currentTime();
LOOP
  Read Inputs;
  Calculate Control;
  Write Outputs;
  Update Internal States;
  t = t + h;
  waitUntil(t);
END
```

The reading of inputs and writing of output signals normally correspond to direct calls to external A-D and D-A conversion interfaces. However, it is also possible to have the sampling and actuation being performed by dedicated tasks or interrupt handlers, in which case buffers are used to communicate the values. In the case of a networked control system the reading and writing of signals may also involve communication with other nodes in the network. To minimize the input-output delay, the control algorithm is often divided into two parts (*CalculateOutput* and *UpdateState*), where the first part computes the control signal based on current measurements and previous states. The second part then updates the internal states of the controller for the next sample.

Real-Time Scheduling

Traditional real-time scheduling theory is concerned with the problem of, given a set of tasks, finding an execution order that guarantees that all tasks meet their timing constraints. Real-time scheduling algorithms fall in two basic categories: *static* and *dynamic* scheduling.

Static scheduling [Locke, 1992; Xu and Parnas, 2000] is an off-line approach, where an optimized execution order is determined once and for all before the system is commissioned. This execution order is then repeated cyclically at run-time. The main benefit of this approach is that it is easy to analyze and thereby guarantee all timing requirements. The main drawback is that the cyclic schedules may be very long and difficult to derive. They also need to be re-calculated every time changes are made to the real-time system.

In dynamic scheduling schemes, the decision of which task to run is taken at run-time. The standard and still most commonly used dynamic scheduling schemes were presented in the seminal paper [Liu and Layland, 1973]. The schedulability theory is based on a hard real-time task model, with all tasks being periodic and where each task, i, is characterized by the following parameters

- a fixed period, T_i,
- a hard deadline, D_i, and
- a fixed and known worst-case execution time (WCET), C_i.

Fixed-Priority Scheduling. Fixed-priority preemptive scheduling is the most common scheduling mechanism and is supported by all major commercial real-time operating systems. In this approach, each task is assigned a fixed priority value. During run-time, the ready task with the highest priority gets access to the CPU. If a task with lower priority is currently running, this task is preempted by the higher-priority task.

For control tasks it is natural to assume that the relative deadlines, D_i, of the tasks are equal to their periods, T_i. In this case, the most common priority assignment is the *rate-monotonic* assignment, where priorities are assigned according to the periods of the tasks. The shorter the period, the higher the priority. It is shown in [Liu and Layland, 1973] that this is an optimal scheduling policy, i.e., if the task set is not schedulable using rate-monotonic assignment it is not schedulable using any other fixed-priority assignment.

Assuming a set of n tasks, the following utilization constraint gives a sufficient condition for schedulability using the rate-monotonic priority assignment:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1). \tag{1.2}$$

In the more general case where $D_i \leq T_i$, deadline-monotonic priority assignment is optimal [Leung and Whitehead, 1982]. Here the priorities are assigned according to the relative deadlines of the tasks.

For any fixed-priority assignment, an exact schedulability analysis can be performed by computing the worst-case response time of each task [Joseph and Pandya, 1986].

Earliest-Deadline-First Scheduling. Under fixed-priority scheduling the priorities of the tasks are static and do not change during run-time. An alternative approach is *earliest-deadline-first* (EDF) scheduling, which uses a dynamic priority assignment based on the absolute deadlines of the tasks. At any point in time, the task with the shortest remaining time to its deadline will get access to the CPU.

EDF is more resource-efficient than rate-monotonic scheduling. A necessary and sufficient condition for schedulability (given $D_i = T_i$) is that the utilization factor is below one:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1. \tag{1.3}$$

A benefit of deadline-based scheduling over priority-based scheduling is that it is usually more intuitive to assign deadlines to tasks than to assign priorities. To assign priorities, global information about the relative importance of all tasks in the system is needed, which is not required to assign deadlines.

One drawback of EDF is that it offers no guarantees at all during overload, in which case all tasks may miss their deadlines. For hard real-time systems this can be fatal. However, the result of a permanent overload situation under EDF, is that the effective periods of all the tasks are scaled

in a fair manner [Cervin *et al.*, 2002]. For reasonable overloads, this will for many control systems still give acceptable performance.

Schedulability Analysis for Aperiodic Tasks. In many applications, the assumption of only periodic tasks does not hold. An important example is web server systems, which handle large volumes of aperiodically arriving requests. Individual requests are often associated with specific deadlines related to their quality-of-service (QoS) requirements. Motivated by this, the derivation of schedulability bounds for aperiodic tasks has been an active research area during recent years.

Schedulability bounds for aperiodic tasks were first presented in [Abdelzaher and Lu, 2001]. The bounds are based on a measure called *synthetic utilization*, $U^{\zeta}(t)$, defined as

$$U^{\zeta}(t) = \sum_{i \in V^{\zeta}(t)} \frac{C_i}{D_i}, \tag{1.4}$$

where $V^{\zeta}(t)$ is the set of current tasks at time t, i.e., tasks that have arrived but whose deadlines have yet to expire.

It can be proved that deadline-monotonic scheduling is an optimal policy for priority scheduling of aperiodic tasks. Using this assignment and assuming n tasks, all tasks will meet their deadlines if, $\forall t$,

$$\begin{aligned} U^{\zeta}(t) &< \frac{1}{2} + \frac{1}{2n}, \quad \text{for } n < 3 \\ U^{\zeta}(t) &< \frac{1}{1 + \sqrt{\frac{1}{2}(1 - \frac{1}{n-1})}}, \quad \text{for } n \geq 3 \end{aligned} \tag{1.5}$$

Real-Time Communication

Communication networks can be divided in two main categories with very different characteristics: *data networks* and *control networks*. Data networks, such as Internet, shuffle large data packets with high irregularity, high data rate, and low real-time requirements. Control networks, on the other hand, instead send small packets regularly and with tight real-time constraints.

Network communication introduces delays in the control system between the sensor, controller, and actuator nodes. Delays occur both in the nodes, due to processing and queuing delays, and between nodes, due to contention and transmission times. Collisions and resendings cause nondeterministic communication delays, which depend on the chosen network protocol.

Three main types of medium access control (MAC) protocols used in control networks can be identified: random access with retransmission of collided packets, random access that uses priorities to avoid collisions, and time-division multiplexing. For a review and comparison of protocols from all these categories, see, e.g., [Lian *et al.*, 2005].

Ethernet is the prime example in the first category and the dominating technology for local area networks, mainly because it is cheap, well supported and can provide bit rates in the order of Gbits/second. Ethernet, however, provides no real-time guarantees and collisions and random back-off may give long and unpredictable delays. Switched Ethernet networks can be used to exploit the advantages of ordinary Ethernet, while at the same time provide reliable communication with predictable real-time behavior [Martinsson, 2002].

Random access with priorities is used in the Controller Area Network (CAN), which has been adopted as one of the standards for real-time communication in the car industry.

Time-division multiplexing protocols include the time-triggered protocol (TTP) and token-passing protocols. These protocols are based on static communication schedules and are characterized by a high level of predictability. On the other hand, the protocols are inflexible and the schedules are hard to maintain and modify. Two recently emerging protocols in the vehicle industry, which aim at combining the time-driven and event-driven communication architectures, are FlexRay [The FlexRay Consortium, 2005] and TTCAN [CAN in Automation, 2005].

Wireless Sensor/Actuator Networks. A new class of communication networks that has emerged during the last years is wireless sensor/actuator networks. In such networks, a collection of small nodes are distributed in an area and designed to achieve a common goal. The individual nodes consist of tiny inexpensive computers that communicate with each other and back-end servers using wireless transmissions and interact with their environment through on-board sensors and actuators.

A common characteristic of these networks is that they usually have severe resource constraints. In addition to resource constraints on CPU speed, memory, and communication bandwidth, the nodes also have hard constraints on the energy consumption. The nodes are battery-driven and, in many applications, placed in remote or hostile regions, which makes recharging of the batteries impossible.

These networks have so far mainly been passive, information-gathering networks. Typical applications consist of surveillance and monitoring systems, and military applications, such as target tracking. In these applications, control techniques can be used to control the performance of the sensor network itself. This includes trade-offs between conflicting ob-

jectives, such as time constraints, power consumption, and information-gathering capabilities. Another option is to close control loops over the sensor/actuator network, which gives rise to other requirements than the ones adopted in normal sensor network design. Matching power constraints against the real-time constraints imposed by closed-loop control is a challenging problem.

Another issue in sensor/actuator networks is the choice of MAC layer protocol for the wireless communication. Two frequently used MAC layer protocols for wireless networks are the IEEE 802.11 WLAN [IEEE, 1999] and IEEE 802.15.4 ZigBee [The ZigBee Alliance, 2004]. While WLAN has a high bit rate, it lacks support for energy conservation. ZigBee, on the other hand, is a standardized protocol aimed at sensor networks that conserves energy by, e.g., avoiding idle listening. This is achieved by sacrificing the real-time performance.

Communication protocols designed specifically for sensor/actuator networks are thus desirable. Successful design of sensor/actuator networks also requires controllers and control design methods that are aware of the properties of the communication links.

Another class of systems that relies on cooperation using wireless communication is mobile ad-hoc networks (MANETs). One example is cooperating mobile robots, such as the teams competing in the annual RoboCup competition [The RoboCup Federation, 2004]. The resource constraints in terms of computing, networking, and power are usually less strict for these systems.

Temporal Non-Determinism

Computer-based control theory is based on idealized assumptions about periodic sampling and constant control delays. However, for reasons outlined above, this can seldom be achieved by the practical implementation, especially in resource-constrained systems.

Depending on the scheduling algorithm, the tasks experience different timing characteristics as a result of interference from other tasks through preemption. Another source of temporal non-determinism is blocking when waiting for common resources. Further, the execution times of the control tasks may be data-dependent or vary due to hardware features such as caches. On the distributed level, the communication gives rise to delays that can be more or less deterministic depending on the communication protocol.

The resulting timing between the reading of the inputs and the generation of the outputs is crucial for the performance of the controlled system. The timing variations introduced by the computer system may lead to substantial performance degradation. The basic timing variations experienced by control tasks are depicted in Figure 1.2. The solid verti-

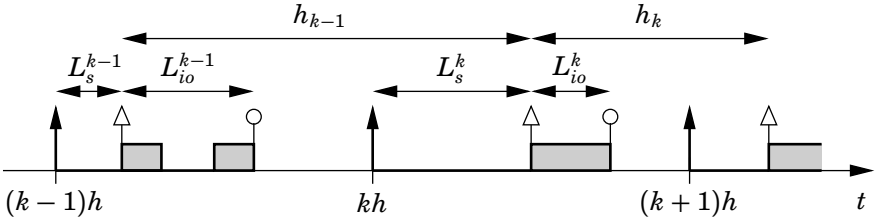

Figure 1.2 Timing non-determinism in periodic control loops (adopted from [Cervin, 2003]). Sampling latency, L_s, input-output latency, L_{io}, and variations (jitter) in these parameters degrade the overall performance.

cal arrows represent the periodic release times of the controller task. The time instants for the A-D and D-A conversions are depicted by transparent arrows and circles, respectively, cf. Figure 1.1.

Input-Output Latency. The delay between the sampling of the measurement signal and the output of the control signal is called the *input-output latency*, denoted L_{io} in Figure 1.2. Input-output latency is primarily caused by preemption from higher-priority tasks, communication delays, and by the execution time of the control algorithm itself. Separation of the control algorithm into *CalculateOutput* and *UpdateState* parts, as shown in Listing 1.1, can be used to reduce the input-output latency.

Jitter. The periodic task that implements the control algorithm is released at equidistant time instants, 0, h, $2h$, ..., where h is the sampling interval of the controller. However, the scheduling may cause the actual start of the task to be delayed some time. This time is known as the *sampling latency* of the task, denoted L_s in Figure 1.2. Variations in the sampling latency is called *sampling jitter*. Sampling jitter will also cause jitter in the actual sampling period.

Another type of jitter is variations in the input-output latency, called *input-output jitter*. This may, e.g., be caused by variations in the execution time of the control algorithm. For simple controllers such as PID-controllers these variations are negligible, whereas more advanced algorithms may have very large execution time variations.

Effects of Temporal Non-Determinism. The effects of temporal non-determinism on control performance are often quite hard to analyze. Already the mapping from the real-time scheduling and communication protocols to the resulting timing properties of the control tasks is complicated.

The input-output latency can be modelled as a time delay at the input of the plant, and has the same effect on the closed-loop system as a process

delay. If not handled properly, the input-output latency may compromise the overall system stability. Performance and stability criteria for systems with time-varying delays are presented in [Lincoln, 2003].

Sampling jitter also has a negative effect on the control performance, which is more profound for slowly sampled systems and systems with a small phase margin. In that case, even small variations in the sampling period may cause instability. An early work analyzing the effects of random delays and random sampling in optimal control is [Davidson, 1973].

In most cases, a shorter but varying latency is better with respect to control performance than a longer, but constant, latency. This is usually true even if the latency is compensated for in the controller design.

1.2 Codesign of Real-Time Control Systems

Many of the assumptions made in the control and real-time scheduling communities are either too restrictive or too idealized to describe the actual behavior of resource-constrained real-time control systems.

The hard real-time task model used in the real-time scheduling community does not capture the special requirements of control tasks. While it might be true that most hard real-time systems are control systems, most control systems are not hard real-time systems. For almost all controllers, single missed deadlines are not critical for the system performance or system stability.

On the other hand, the assumptions made in computer-based control theory do not consider the effects of the actual implementation of the controller as a task in a real-time system. The timing variations introduced by the computer system are crucial for the performance of the control system and must be taken into account at design time.

Consequently, one realizes that the development of real-time control systems under resource constraints is essentially a codesign problem. For optimal use of limited computing resources and for optimal control performance, the controller design and the software design need to be integrated. A definition of the control and scheduling codesign problem may be stated as [Cervin, 2003]:

> Given a set of processes to be controlled and a computer with limited computational resources, design a set of controllers and schedule them as real-time tasks such that the overall control performance is optimized.

To facilitate this development, new ways of thinking are required both from the control and computer science perspectives. Codesign of controllers and their real-time implementation can, thus, be covered by the terms *implementation-aware control* and *control-aware implementation*.

Implementation-Aware Control

Implementation-aware control aims at developing methods that take implementation issues and limited resources into account during the controller design. This includes both synthesis techniques that compensate for timing variations and analysis techniques that can be used to verify the robustness against implementation effects.

Compensating Controllers. Timing non-determinism induced by the implementation platform may be compensated for either off-line, based on knowledge of the timing variations, or on-line, based on measurements of the actual variations.

Compensation for time-varying delays in distributed systems is treated in [Nilsson *et al.*, 1998]. Models of network delays are studied and analyzed and compensating controllers are developed. This work is based on time-stamping of packets, which means that the length of past time delays between controller and actuator and between sensor and controller are known to the controller.

Temporal Robustness. Robustness against parameter variations and structured uncertainties has been an active research area during recent years and a number of design methods for robust control have been developed [Zhou and Doyle, 1998]. However, theory for handling uncertainties related to the implementation is not nearly as well developed.

By temporal robustness is meant robustness against timing variations, e.g., how much sampling and input-output jitter that can be tolerated before losing stability. Such information can then be used to verify that the performance of a chosen computing and communication platform is sufficient for the control applications.

Robustness against constant input-output latency is covered by the classical phase and delay margins. [Cervin *et al.*, 2004] exploits stability results in [Lincoln, 2003] to develop the jitter margin, an extension of the classical delay margin to the case of time-varying delays. The results do not cover sampling jitter, however. Approximate models for linear systems with sampling jitter are treated in [Boje, 2005].

The Jitterbug tool [Lincoln and Cervin, 2002] can be used to perform stochastic control performance analysis under various timing conditions. Robustness to input-output delay, sampling jitter, output jitter, lost sam-

ples, period overruns, and aborted computations can be evaluated in terms of both performance and stability.

Control-Aware Implementation

Approaches to control-aware implementation include hardware solutions, models of computation, real-time scheduling algorithms, and implementation techniques tailored to the special needs of control systems. This can be approached using either static techniques, which aim at maximizing the determinism, or by dynamic techniques, where timing variations are compensated for on-line.

Maximization of Determinism. The temporal non-determinism can be minimized by certain choices of implementation techniques and platforms. These include time-driven architectures such as TTA [Kopetz, 1997] and synchronous programming languages such as Esterel, Lustre, and Signal [Halbwachs, 1993]. The embedded systems programming model Giotto [Henzinger *et al.*, 2001] combines time-triggered I/O with dynamic real-time scheduling of the computations. These techniques have the benefit that they minimize, or even remove, the jitter and keep the input-output latency constant. This simplifies controller design and analysis as well as the verification process for safety-critical applications.

The controllers may also be implemented to remove the jitter by performing the sampling operation in a dedicated high-priority task, and by always delaying the output to the end of the period. This will introduce a constant one-sample delay in the system, which then can be compensated for in the controller design. However, as shown in [Cervin, 2003], the compensation may only recover part of the loss introduced by the extra input-output latency. Therefore, designing and compensating for the worst-case input-output latency may not always be a viable option.

One approach to prevent scheduling-induced input-output delay and jitter could also be to use non-preemptive scheduling. With the increasing speed of modern computers it can be argued that the relative execution times of different tasks will become smaller and smaller, thus making this approach realistic also from a schedulability point-of-view.

Task Models and Dynamic Scheduling for Control Tasks. In the hard real-time task model it is assumed that it is imperative for system correctness that all tasks complete before their deadlines, and that missing single deadlines may have catastrophic consequences for the controlled system. On the other hand, in the soft real-time task model any number of deadlines may be missed without being considered a failure.

However, control system can in most cases neither be said to be hard nor soft. A more fitting term used to describe real-time control tasks is *adaptive* [Bouyssounouse and Sifakis, 2005]. A real-time task is said to be adaptive if missing one or more deadlines does not jeopardize the correct system behavior, but only causes a performance degradation. Taking this approach to modeling control tasks allows the control performance to be viewed as a quality-of-service measure, quality-of-control (QoC).

Other task models that can be suitable for control systems are subtask models and the imprecise task model. The subtask model is motivated by the natural division of a standard control algorithm in a *CalculateOutput* and an *UpdateState* part. Several subtask scheduling models have been proposed in the real-time scheduling community. A control-oriented approach is taken in [Cervin, 1999], where by scheduling the *CalculateOutput* and *UpdateState* parts separately, the input-output latency can be minimized, thus, improving performance. Similar ideas are presented in [Balbastre *et al.*, 2004], where subtask scheduling is used to minimize jitter. The imprecise task model for control tasks is treated in Section 1.3. The aperiodic task model may also be considered in the case of event-based sampling for control systems.

The concept of server-based scheduling has recently gained much interest in the real-time scheduling community. The constant bandwidth server (CBS) [Abeni and Buttazzo, 1998] conceptually divides the CPU into a number of virtual sub-CPUs with given capacities. The CBS then guarantees that tasks running in the virtual CPUs never consume more than the alloted capacity. The control server, an extension of CBS tailored for control tasks, is presented in [Cervin, 2003]. A control server creates the abstraction of a control task with a specified period and a fixed input-output latency shorter than the period. The control server model is well suited for codesign in that the single parameter linking the scheduling design and the controller design is the task utilization factor.

Another approach to dynamically compensate for timing variations in order to optimize control performance, is by means of feedback in the real-time system—feedback scheduling. Feedback scheduling is an approach to achieve flexibility in the run-time scheduling of control tasks. The objective is to optimize the control performance for control loops under resource constraints. In feedback scheduling, the available resources are scheduled dynamically based on measurements of actual timing variations and control performance. An introduction to the feedback scheduling concept will be given next.

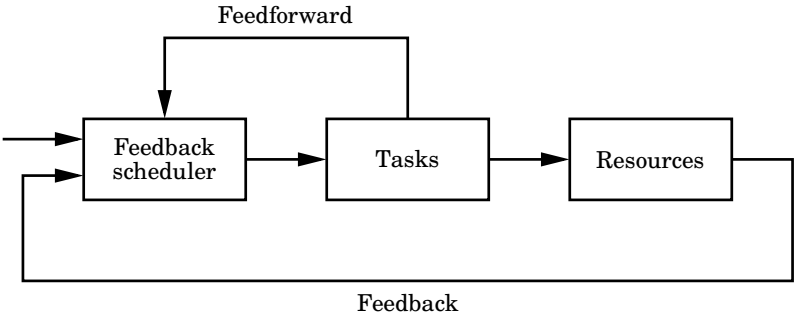

Figure 1.3 A general feedback scheduling system. The scheduler adjusts the tasks' demands based on feedback from the current use of critical resources. The tasks may also inform the scheduler that they are about to consume more resources (feedforward).

1.3 Feedback Scheduling

A characteristic property of feedback is that it can be used to reduce the effects of disturbances and to deal with uncertainties. The idea of feedback scheduling is to use feedback to master uncertainty with respect to resource scheduling, such as variations in task execution times. A general feedback scheduling system is shown in Figure 1.3. The idea is to feed back the actual use of critical resources to the scheduler and to continuously adjust the tasks' demands of resources according to the current situation. The reactive feedback may also be combined with proactive feedforward, for instance, task admission control schemes and corrections due to task mode changes.

Feedback scheduling is not suitable for tasks of truly hard nature, since transient system overloads will cause some tasks to miss deadlines before corrections are taken based on the feedback. Instead, feedback scheduling is primarily intended for tasks of soft or adaptive nature. As described above, most control applications fit into this category of tasks.

Feedback scheduling is closely related to quality-of-service approaches for soft real-time activities, such as multimedia applications. Some efforts have actually been made to treat the control performance as a quality-of-service parameter, and to specify reasonable ranges for the performance metrics, including rise times, overshoots, and steady-state variances [Ryu and Hong, 1998]. In this framework, the on-line resource negotiation can then, e.g., be specified using contracts [Eker and Blomdell, 2000] relating the control performance to the available resources. Quality-of-control approaches based on quadratic cost functions related to time-varying delays and periods is the topic of [Sanfridsson, 2004].

Another example is [Abdelzaher *et al.*, 2000], which considers quality-of-service negotiation in flight control systems. Here task periods and deadlines are treated as negotiable parameters between tasks in the system, allowing graceful quality-of-service degradation under conditions where traditional schedulability analysis fails.

Feedback Scheduling of CPU Resources

A feedback-based scheduling algorithm called Feedback Control EDF (FC-EDF) is presented in [Stankovic *et al.*, 1999; Lu *et al.*, 1999]. The objective is to regulate the deadline miss-ratio for a set of tasks using a PID-controller that applies changes to the CPU utilization of the tasks. The approach is extended in [Lu *et al.*, 2000; Lu *et al.*, 2002] to control the CPU utilization instead of the deadline miss-ratio and to improve the transient performance.

An adaptive rate control mechanism based on an elastic task model is presented in [Buttazzo *et al.*, 1998]. The task period adjustment is based on elasticity coefficients, e_i, related to the utilization factors of the tasks. The utilization of each task is conceptually viewed as the length of a spring with rigidity coefficient, $1/e_i$.

Feedback can also be used in connection with reservation-based scheduling implemented using task servers, such as the constant bandwidth server. [Abeni *et al.*, 2002] develops a mathematical model of a reservation-based scheduler and uses it to design a PI-controller that dynamically assigns bandwidths to a set of constant bandwidth servers. Stochastic control of reservations is the topic of [Cucinotta *et al.*, 2004], where the scheme is also evaluated in an implementation in a Linux kernel. A hybrid-control approach to adaptive reservations is presented in [Palopoli *et al.*, 2003].

Other feedback-scheduled resources than the CPU time may also be considered, such as network bandwidth or memory allocation. An approach to achieve adaptive garbage collection (GC) and incorporate GC scheduling into a general feedback scheduling framework is presented in [Gestegård Robertz, 2003].

Feedback Scheduling of Control Tasks

Some control algorithms are associated with highly varying execution times. Examples include model predictive controllers and controllers that switch between different modes. For these control schemes, the execution time variations are inherent in the algorithms. Other sources to variations may be data-dependencies or specific hardware features.

Varying execution times make traditional task scheduling infeasible. Algorithms such as rate-monotonic and earliest-deadline-first are both

open-loop scheduling algorithms, in the sense that the schedulability results are obtained off-line, assuming complete knowledge of the task parameters.

Since the execution time may vary significantly, the main limitation lies in the assumption of known execution time bounds for all tasks. A design based on worst-case times will likely become far too pessimistic and lead to severe under-utilization of the computer resources. For control tasks this means slower sampling and decreased performance. In reality the execution time bounds are also very difficult to obtain.

The main difference between feedback scheduling of control tasks and feedback scheduling of CPU resources in general is the connection to control performance. For feedback scheduling of control systems, the available resources should be distributed to optimize the global control performance, while still meeting the schedulability constraints in the system.

Actuators and Sensors. There are two main ways to control the CPU demand of control tasks: by manipulating the task periods, or, for a certain class of controller algorithms, by manipulating the execution times.

Task period manipulation is well suited for most controllers, such as standard PID and state-feedback controllers. For these controllers the CPU utilization is controlled by changing the sampling interval within an allowable range. For algorithms of iterative nature it may be possible to abort after an arbitrary number of iterations. This gives the possibility to manipulate the execution time and to do a trade-off between CPU consumption and the quality of the computed control signal.

Sensors used in the feedback scheduling framework typically consist of execution measurements. However, feedback from the actual control performance could also be used in the scheduling decisions.

Task Period Modifications

Dynamic resource allocation by means of task period rescaling has been explored in several papers. [Beccari *et al.*, 1999] considers modulation of sampling rates for robot systems. A range of admissible rates is identified for each task, and different rate-monotonic schemes are presented and evaluated. [Shin and Meissner, 1999] studies resource adaptation in multiprocessor systems. Reallocation of control tasks and on-line adjustment of sampling rates are used to optimize a quadratic performance index related to the global control performance. A feedback scheduled system manipulating sampling intervals can be viewed as a special case of a hybrid control system. An interesting example is given in [Schinkel *et al.*, 2002], which considers switching between two linear quadratic (LQ) controllers designed with different sampling intervals. Although both closed-

loop systems are stable, it is shown that a special switching sequence between the systems leads to instability.

Constrained Optimization Approaches. The approach of constrained optimization for task period selection was first proposed in [Seto *et al.*, 1996]. A performance index, J, expressed as a function of the sampling period, h, was used as basis for the optimization. The problem was introduced as follows:

$$\begin{aligned} &\min_{h_1 \ldots h_n} \sum_{i=1}^{n} J_i(h_i), \\ &\text{subj. to } \sum_{i=1}^{n} \frac{C_i}{h_i} \leq U_{sp}. \end{aligned} \tag{1.6}$$

An optimal feedback scheduling strategy based on this formulation for sampling period adjustments of LQ controllers was presented in [Eker *et al.*, 2000; Cervin, 2003]. Cost functions relating the stationary cost to the sampling interval were developed and analyzed. It was also shown that simple linear,

$$J_i(h_i) = \alpha_i + \gamma_i h_i, \tag{1.7}$$

or quadratic,

$$J_i(h_i) = \alpha_i + \beta_i h_i^2, \tag{1.8}$$

functions could be used as approximations of how the true cost functions depend on the sampling intervals, h_i. Here, α_i, β_i, and γ_i are constants. Based on these approximate functions, explicit solutions to the optimization problem (1.6) can be derived.

It was also stated in [Cervin, 2003] that the assumptions of linear or quadratic cost functions yield simple calculations of the optimal sampling periods to be used on-line by the feedback scheduler. In both cases, linear proportional rescaling of the nominal sampling periods is optimal. All periods are changed by the same factor: $(\gamma_i/C_i)^{1/2}$ for the linear case and $(\beta_i/C_i)^{1/3}$ for the quadratic case. This is a nice property, since it is fast and easy to implement. It also preserves the rate-monotonic ordering among the control tasks, which avoids priority changes of the tasks during run-time. It should, however, be noted that this simple rescaling only works when new tasks enter or leave the system, or if the utilization set-point, U_{sp}, changes. If, on the other hand, the execution time of a task changes, the nominal periods and the proportionality constants need to be recomputed.

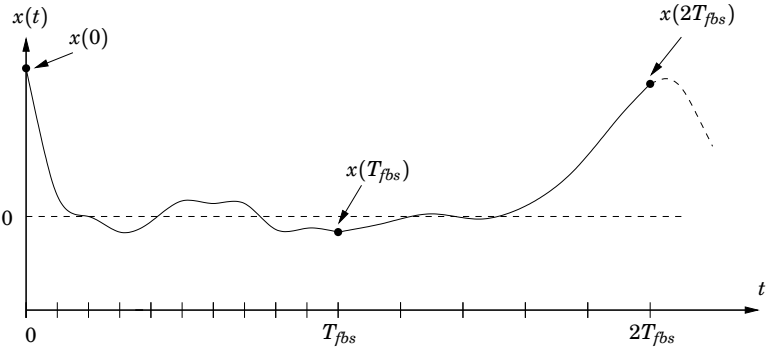

Figure 1.4 Scheduling based on plant states. Deviations in the state cause the corresponding control loop to be sampled faster.

Feedback from Control Performance. The approach in [Eker *et al.*, 2000; Cervin, 2003] is based on stationary, i.e, infinite-horizon, cost functions, and therefore does not exploit any feedback from the current control performance. Plant disturbances do not affect the dynamic resource allocation. Ideally, the scheduler decisions should also contain feedback from the actual control performance and not only resource consumption.

An approach to dynamic resource allocation based on plant state feedback is presented in [Martí *et al.*, 2004]. The suggested state-dependent cost functions are, however, not based on control-theoretic arguments.

Paper I formulates and analyzes state-dependent cost functions for the case of LQ controllers. The cost functions are based on a finite-horizon control formulation. The current states of the processes at the invocation of the feedback scheduler enter the cost functions and may, hence, affect the resource allocation. The idea is illustrated in Figure 1.4. The basic observation is that control loops running in stationarity can be given less resources, i.e., can be sampled slower, than a process in a transient phase. The feedback scheduler is running as a periodic activity with period T_{fbs}. In this formulation, the cost functions transform from being functions of only the sampling period, i.e.,

$$J_i(h_i) \approx \alpha_i + \beta_i h_i^k, \tag{1.9}$$

to also depend on the state, x_i, and the feedback scheduler period, T_{fbs},

$$J_i(x_i, h_i, T_{fbs}) \approx \alpha_i(x_i, T_{fbs}) + \beta_i(x_i, T_{fbs}) h_i^k. \tag{1.10}$$

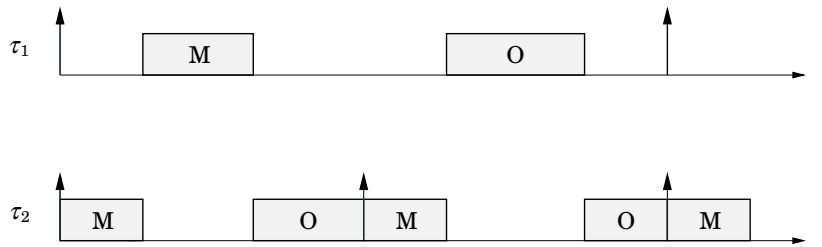

Figure 1.5 Scheduling of imprecise computations is based on a task model where each task can be divided into two parts: a mandatory part and an optional part. The optional part may be aborted to meet scheduling constraints or to improve performance.

Execution Time Modifications

For certain classes of control algorithms, an alternative to sampling period adjustments is manipulation of the actual execution time of the control computation. These types of algorithms are generally referred to as *anytime algorithms* or imprecise computation algorithms.

The main characteristic of anytime algorithms is that they always generate a result, but with a quality level that increases with the execution time. This means that there is a trade-off to consider between the computational time and the result generated by the algorithm.

The task model for scheduling of imprecise computations [Liu *et al.*, 1991; Liu *et al.*, 1994] assumes that all tasks can be divided into two subtasks, a *mandatory* subtask and an *optional* subtask, see Figure 1.5. An imprecise result may be returned by the algorithm as long as the mandatory subtask has completed. In [Liu *et al.*, 1991], imprecise calculation methods are categorized into three main types: *sieve function methods*, *multiple-version methods*, and *milestone methods*.

Sieve functions constitute optional computation steps that may be skipped to save processing time. An example of a sieve function for control algorithms is the updating of the estimated parameters in an adaptive controller. Multiple-version methods exploit several versions of the algorithm, with different processing times and result quality.

Milestone methods are based on monotone algorithms, ensuring that the quality of intermediate results increases monotonically with time. This type of algorithms can be found in many application areas, including numerical optimization, estimation, and prediction. Scheduling of monotone imprecise tasks is treated in [Chung *et al.*, 1990]. In this scheme, each mandatory subtask is scheduled to complete before the deadline of the task, and the optional parts refine the results produced by the tasks.

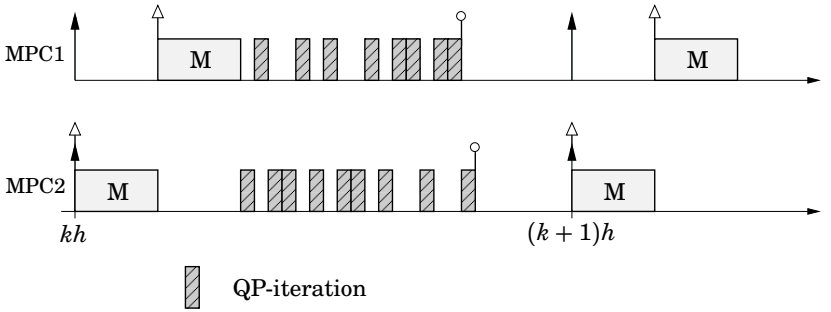

Figure 1.6 Imprecise computation model for model predictive control tasks. The mandatory part (M) consists of finding a feasible initial solution and iterating the QP-solver until the stability requirement is fulfilled. The additional QP-iterations constitute the optional part and may be skipped.

Application to Model Predictive Control. An example of a control methodology that fits the milestone method very well is model predictive control (MPC), which is the topic of Paper II. This control strategy is based on on-line minimization of a quadratic cost function in every sample, subject to constraints on control signals and controlled variables. In the MPC formulation used in Paper II, the optimization problem is solved by an iterative quadratic-programming (QP) solver that guarantees that the value of the cost function is reduced in each step of the algorithm.

The mandatory part of the control algorithm consists of finding an initial solution that fulfills the constraints of the QP-problem and iterating the solver until the solution guarantees closed-loop stability. The optional part consists of the remaining QP-iterations that further reduce the cost. These iterations may be skipped if computing time is scarce.

Figure 1.6 illustrates a situation with two MPC tasks running concurrently. A dynamic scheduling strategy schedules the mandatory parts using distinct high priorities and the optional parts of the tasks using the cost functions as dynamic task priorities. By always executing the task with the highest cost first, the aim is to achieve as small global cost as possible before the optional parts are terminated.

Anytime Sensing. Paper III treats another aspect of anytime algorithms in control systems. In many vision-based algorithm it is possible to do a trade-off between the tracking accuracy and the computational effort, treating the vision-feedback algorithm as an anytime sensor.

More specifically, the tracking algorithm used in Paper III measures the image error as the distance (in the normal direction) from a number of edge detection points to the real edge. By choosing the number of edge

detection points, it is possible to influence both the estimation accuracy and the computational burden of the algorithm.

The setup analyzed in Paper III consists of multiple cameras used to track the position and orientation of a moving object. To increase the size of the tracking region, the cameras have different settings and distances to the target. This avoids the very large variations in execution time that may result if the target is lost and the image search has to be extended. By using multiple standard digital cameras it is also possible to achieve high positioning accuracy at a fairly low cost.

It is assumed that the image pre-processing and the execution of the tracking algorithm is performed by a single central computer. The total time to obtain sensor data in each sample is a function of both the number of cameras used and the total number of edge detection points distributed between the different camera images.

The resource allocation is performed in camera space and consists of finding an optimal subset of the available cameras in each sample. The aim is to minimize the estimation covariance by a proper choice of active cameras and a distribution of the edge detection points between these cameras. The minimization is subject to constraints on the total execution time of the image processing and the tracking algorithm.

The anytime nature of the tracking algorithm can also be used for dynamic scheduling of several vision-based control tasks to optimize control performance under resource constraints. This would require models relating the latency and the estimation accuracy to the control performance.

1.4 Control of Server Systems

In feedback scheduling for control tasks, the performance metrics used for dynamic resource allocation are related to controlled physical plants. However, feedback mechanisms can also be applied to control the internal performance of software applications.

Inspired mainly by the growth of Internet, the use of feedback control theory has emerged as a promising foundation for performance control in large, complex software applications, such as web servers. These typically operate under very unpredictable and poorly modeled load conditions. Managing this uncertainty by means of control theory has proved to be successful in order to provide quality-of-service guarantees for these systems [Hellerstein *et al.*, 2004; Robertsson *et al.*, 2004; Robertsson *et al.*, 2003; Abdelzaher *et al.*, 2003; Sha *et al.*, 2002].

The use of feedback in computer systems is not a new concept. One example is the flow and congestion control mechanisms used by the transmission control protocol (TCP), which were introduced in the 1970s to

solve the congestive failures that had brought down the network. This was an early testament to the effectiveness of feedback control in highly dynamic, decentralized environments. Although feedback has been applied to software systems for a long time, many of these efforts have been ad-hoc, without any real connections to traditional control theory.

The use of control theory to achieve quality-of-service guarantees in modern web server applications presents a number of new challenges compared to control of ordinary physical plants. Server systems are characterized by highly stochastic and nonlinear behavior. Response times increase exponentially for heavy loads and there are both input and output saturations present. The parameters of the stochastic processes may also change abruptly.

One central question is whether to use time- or event-based sampling. Server systems are by nature event-based, with requests entering and leaving the system at any time. In certain situations, however, differential (or difference) equations may be used to accurately describe the systems.

The stochastic nature of the systems requires averaging of the measurements used by the feedback controller. Usually, the servers themselves have quite simple dynamics, and most of the dynamics in these systems is introduced by the filters used for signal processing. The stochastics also make accurate prediction and model-based feedforward more important than for standard control systems. With good prediction, it is seldom necessary to use sophisticated feedback controllers.

A variety of actuators may be used by the controller. Basically, there are two main ways to influence the server load: by either changing the arrival rate or changing the service rate. These actuators may also be used in combination. Manipulation of the arrival rate is typically achieved by admission control, whereas the service rate may be influenced in a number of ways depending of the nature of the server.

A final challenge is the evaluation of the control performance. Which is most important, transient or steady-state behavior? Also, the concept of stability is not straightforward to define for server systems.

Issues related to control of computer systems are treated in [Hellerstein *et al.*, 2004]. A number of example applications are given, including a Lotus e-mail server, an Apache web server, and an overloaded router.

Queuing Models for Server Control

A schematic picture of a web server is shown in Figure 1.7. A client request queue stores incoming requests, from which requests are subsequently dequeued and served by server worker threads. The worker threads are put in different queues, such as the CPU ready queue or I/O access queues, during the processing of the requests. Thus, web-server control can largely be viewed as complex queue management, characterized by a high level

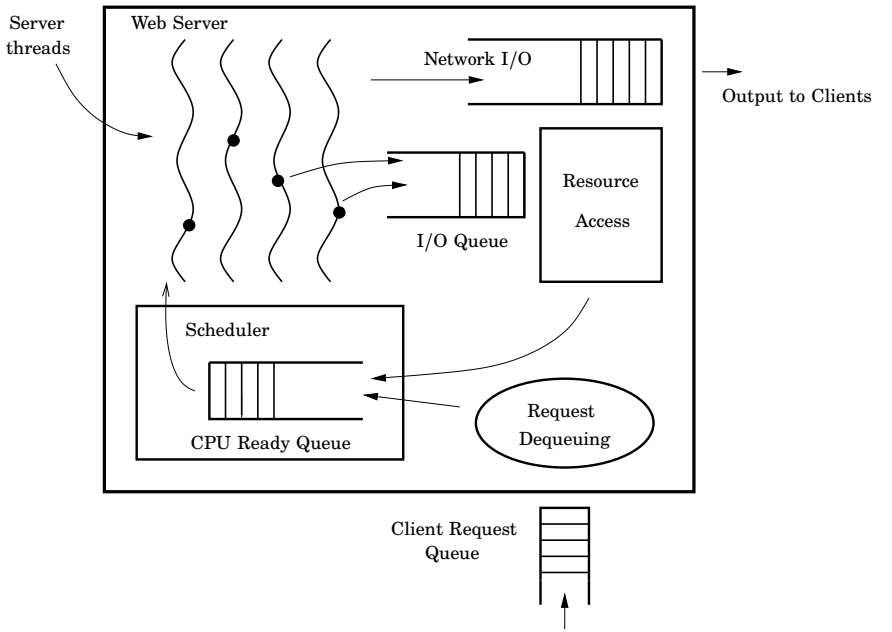

Figure 1.7 Simplified model of a web server (adopted from [Abdelzaher *et al.*, 2003]). Requests are served by worker threads, which are put in different queues, such as the CPU ready queue or I/O access queues, during the processing of the requests.

of uncertainty. The stochastic nature of the queues in the server systems makes traditional dynamic-system modelling and control theory insufficient. Instead, the incoming traffic and the service times can be modeled by stochastic processes, and described and analyzed using queuing theory [Kleinrock, 1975].

Nonlinear Flow Models for Admission Control. The work in [Robertsson *et al.*, 2003] is based on a nonlinear flow model known as Tipper's model [Agnew, 1976; Tipper and Sundareshan, 1990; Sharma and Tipper, 1993; Wang *et al.*, 1996]. This model relates the steady-state behavior of the queue length, x, to the average arrival rate, λ, and service rate, μ, for a single server queue. It is shown that the following relation holds:

$$\dot{x} = \lambda - \mu G(x(t)), \tag{1.11}$$

where the function $G(x(t))$ depends on the statistical properties of the

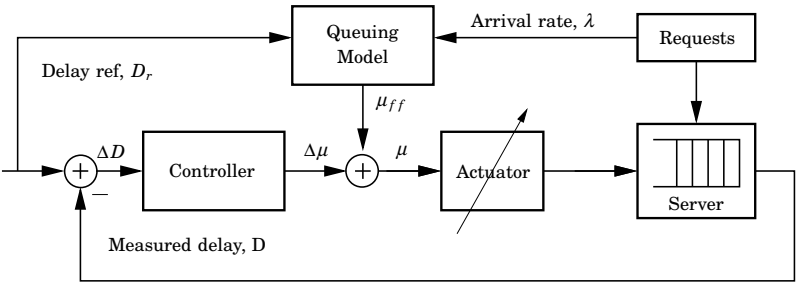

Figure 1.8 Block diagram for combined feedforward/feedback delay control (adopted from [Sha *et al.*, 2002]).

queuing system. For example, for an M/M/1 system, we have

$$G(x(t)) = \frac{x(t)}{x(t)+1}, \tag{1.12}$$

and for an M/G/1 system the expression becomes

$$G(x(t)) = \frac{x(t)+1-\sqrt{x^2(t)+2C^2x(t)+1}}{1-C^2}, \tag{1.13}$$

where C^2 is the squared coefficient of the variance of the service time distribution.

The model (1.11) can be linearized around a chosen operating point $x = x^0$, rendering a first-order linear system with the fraction of the arriving requests as input (admission control).

The nonlinear flow model works well for heavy load situations where the fluid models well approximate the true behavior of the queues in the system.

Queuing-Model-Based Delay Control. A different approach that also exploits models from queuing theory is presented in [Sha *et al.*, 2002]. Here, the queuing model is used to provide a feedforward action according to the controller structure shown in Figure 1.8. In this setup, the manipulated variable is the service rate, which can be changed in different ways, e.g., by allocation of server threads or, for high-performance servers, by dynamic voltage scaling [Bohrer *et al.*, 2002; Sharma *et al.*, 2003].

The idea is to use the feedforward predictor to bring the average delays experienced by the requests close to a desired value, D_r. For the example

of an M/M/1 queue, again, the average delay of the requests in the system can be expressed as

$$D = \frac{1}{\mu - \lambda}, \tag{1.14}$$

where λ is the arrival rate, and μ is the service rate. This equation can then be used to solve for the feedforward term, μ_{ff}, that corresponds to the given delay set-point, D_r,

$$\mu_{ff} = \frac{1}{D_r} + \lambda. \tag{1.15}$$

Since the feedforward predictor moves the system close to the desired delay, a linear P or PI-controller can be used locally to suppress transient errors or errors due to modeling inaccuracies.

Non-Conventional Queuing Theory

The use of traditional queuing theory to model the behavior of web server systems has two main drawbacks. The first is that queuing-theoretic models only relate to the steady-state behavior of the queues, i.e., the long-term averages. Internet traffic, however, is generally very bursty and changes abruptly.

The second drawback is that queuing theory makes certain restrictive assumptions about the arrival and service processes of the system, which are often poorly matched by reality. In real server queues, the statistic nature of the traffic may show considerable variations, and standard Poisson processes do not capture this behavior. Instead, the Pareto distribution has been reported to fit measurements of real web traffic well [Crovella and Bestavros, 1997]. This distribution has typically a long tail, and shows self-similar and long range-dependent characteristics.

Paper IV presents an improved predictor scheme, that makes no assumptions about the statistical properties of the incoming traffic load and the service times. Instead, it predicts future delays as a function of instantaneous measurements of the situation in the server queue. This includes current queue length and the arrival times of the queued requests, which are assumed to be recorded for use in the prediction.

Figure 1.9 shows a geometric picture used to derive the predictor equation. The horizontal axis shows the evolution of time, and the vertical axis shows the cumulative number of arrivals and departures of requests. Each horizontal two-coloured block in the figure represents one request and is divided in queuing time and processing time. The vertical distance in the shaded area at any point in time (for example the distance CB at time t_{now}) represents the actual queue length.

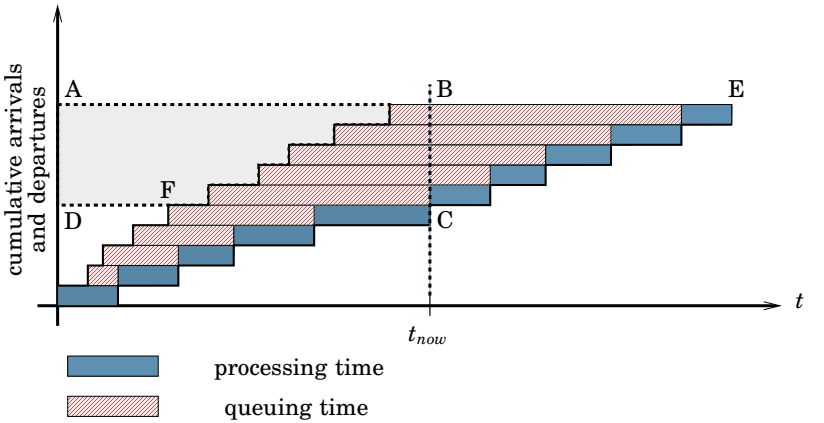

Figure 1.9 Server queuing and processing delay over time.

For the situation in the figure it can be noted that the line from the origin to point B gives the average arrival rate of the ten first requests. Similarly, the line between the origin and point C represents the average service rate of these requests up until time t_{now}. Since the arrival rate exceeds the service rate, it can be seen that the delays experienced by the requests build up.

The idea is to select the service rate to achieve a desired average delay of the requests in the system, taking into account their average queuing delay up until t_{now}. Geometrically, this means modifying the slope of the line CE to obtain a desired area of the quadrangle $CEBF$.

The predictor equation is derived for the requests currently in the system, assuming no more arrivals during the prediction horizon. However, the prediction is repeated as requests depart in a manner similar to the receding horizon principle used in model predictive control. This way sudden variations in the arrival pattern are taken care of continuously, allowing a more rapid response than standard queuing-theoretic models.

The results presented in Paper IV are concerned with absolute delay control by manipulation of the service rate of incoming requests. Note, however, that the predictor could also easily be used in a scheme to adjust the arrival rate using admission control. In that case one would instead change the slope of the arrival curve, under the assumption that the service rate is constant.

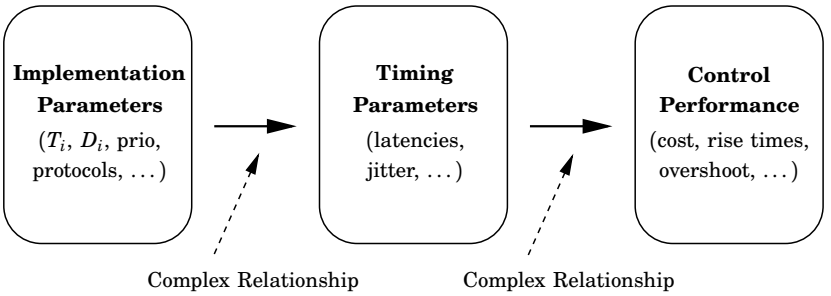

Figure 1.10 The complex relationships between control system performance and computer system design parameters make co-simulation tools important.

1.5 Simulation of Control and Computer Systems

Most control and implementation codesign problems can be stated as constrained optimization problems. One definition of the control and scheduling codesign problem was given in Section 1.2, focusing on optimization of control performance subject to given computational resources. An alternative definition could be stated as

> Given a set of systems to be controlled and control performance specifications for these, choose an implementation platform and decide the allocation of control tasks, and their scheduling, such that the overall production cost is minimized while guaranteeing the specified control performance.

This formulation instead focuses on minimizing the hardware cost subject to constraints on the achieved control performance.

A major challenge in solving both types of codesign optimization problems is that the relationships between the involved parameters are nonlinear and difficult to formulate. Figure 1.10 illustrates this. The mapping from implementation parameters, such as task periods, deadlines and priorities, to the resulting timing parameters in terms of jitter and latencies is often hard to derive and analyze. The same holds for the relation between the jitter and latencies and the resulting control performance as discussed earlier.

Consequently, to aid in the codesign, development of computer-based tools for integrated simulation, analysis, and synthesis of real-time control systems is important. The tools should be able to deal with both the relation between implementation and timing parameters and the effects of the timing parameters on the control performance.

Existing Tools

The main simulation tool used for control system design and simulation is MATLAB/Simulink [The Mathworks, 2001b]. During recent years, Modelica [Tiller, 2001] has emerged as a strong alternative to MATLAB/Simulink for physical modeling and simulation. However, neither of these simulation environments have sufficient support for simulation of real-time implementation issues. Real-Time Workshop [The Mathworks, 2001a] for MATLAB allows prototyping and implementation of real-time control systems, but has no support for simulation of shared CPU resources or communication networks.

In the real-time research community, a number of prototype tools have been developed for schedule simulation, timing analysis and schedule generation. Examples include PERTS/DRTSS [Storch and Liu, 1996] and STRESS [Audsley *et al.*, 1994]. These tools are typically used to prove feasibility of task sets and to perform co-simulation of task execution and hardware architecture and kernels. The simulations do not capture the effects of the scheduling on the performance of the applications implemented by the various tasks.

In recent years, a few co-simulation tools have emerged, which aim at integrated modeling and design of control systems with their real-time implementation. While these tools use different approaches and levels of abstractions, they all aim at bridging the gap between the domains of control and computer system design.

Some of the tools are specifically tailored towards control and real-time codesign, whereas for others, the real-time control systems simulation is just one part of a larger framework. The abstraction levels range from a very high level of abstraction of the computer system in terms of time-varying delays and jitter, to detailed architectural models, which actually mimic the operation of for example an RTOS. These codesign tools are briefly summarized below. See [Henriksson *et al.*, 2005] for an extensive survey of the tools.

AIDA. The Aida toolset [Redell *et al.*, 2004] is an environment for model-based design and analysis of real-time control systems, which allows analysis of implementation effects on control performance and timing analysis of the real-time design. The toolset consists of a modelling environment, *Aidasign*, which interfaces with MATLAB/Simulink, and a response time analysis tool, *Aidalyze*.

The real-time system design starts with the translation of the Simulink model to a *data-flow diagram* (DFD) in Aidasign. The timing aspects of the controller, such as sampling periods and delays, then constitute requirements on the real-time system design. Another fundamental model in Aida is the *hardware structure diagram* (HSD), where the hardware

architecture, in terms of processors and their interconnections via communication links, is designed. In the HSD, the functions and data flows in the associated data-flow diagrams are mapped to processors and communication links, respectively. Based on the two fundamental models and the mapping between them, a real-time implementation is designed.

XILO. XILO (X-in-the-loop simulation) is a toolset built upon MATLAB/Simulink, developed to support detailed architectural design of distributed real-time control systems [El-Khoury and Törngren, 2001]. XILO consists of a number of libraries that let a designer configure a distributed computer control system and to allocate and partition the functionality as desired. Along with the basic toolset, an additional library that supports fault-injection in terms of bit-flips in all types of blocks, signals and constants has been developed [Norberg and Törngren, 2003]. Some of the basic mechanisms of XILO have been reused in the Aida toolset. The tool is entirely graphical and currently limited to the scheduling policies and network protocols provided by the tool libraries.

Ptolemy II. Ptolemy II is the third generation of software produced within the Ptolemy project [Hylands *et al.*, 2003]. Ptolemy II supports *heterogeneous, hierarchical* modeling, simulation, and design of concurrent systems, especially embedded systems. The focus is on complex systems mixing various technologies and operations. Simulation models are constructed under *models of computation* that govern the interaction of the components in the model. Different models of computation are used for modeling different types of systems. The timed multitasking (TM) model of computation [Liu and Lee, 2003] is intended to support deterministic design of concurrent real-time software. This makes it possible to model fixed-priority scheduling of tasks with constant execution times.

RTSIM. RTSIM [Casile *et al.*, 1998] was originally developed as a pure real-time scheduling tool, but has since then been extended [Palopoli *et al.*, 2000; Palopoli *et al.*, 2002] with a numerical module, based on the mathematical library OCTAVE [Eaton, 1998], for continuous plant simulation. The tool consists of a collection of C++ libraries allowing the user to specify a set of plants, the functional controller behavior, the implementation architecture, and a mapping of functional behavior onto the architectural components.

The simulation produces results related both to the real-time performance and the control performance. This includes the generation of execution traces, real-time statistics (e.g., delays and jitter), and control performance metrics such as time responses and quadratic costs. The tool,

however, currently lacks a graphical plant modeling environment, and so far its network simulation capabilities are limited.

Orccad and Syndex. Orccad [Simon and Girault, 2001] is a CAD system and approach aimed at the development of robotic systems from high-level specifications down to the implementation details. It deals with hybrid systems, where continuous-time aspects related to control laws must be merged with discrete-time aspects related to control switches and exception handling.

The Syndex tool supports rapid prototyping of reactive data-driven algorithms implemented on distributed heterogeneous hardware architectures [Pernet and Sorel, 2003; Lavarenne *et al.*, 1991]. Syndex lets the user specify both the algorithm and the distributed hardware in a graphical environment, and then automates the mapping and scheduling of functions and communications on the processors and communication buses. During the mapping and scheduling process, the hardware architecture can be refined to better match the algorithm needs. When a sufficiently good solution has been found, Syndex generates executable code that can be downloaded to the target hardware.

Other Commercial Tools. Apart from an increasing interest in the academic communities, there are also strong industrial needs for tools supporting integrated development. Existing commercial tools, see, e.g., [dSPACE, 2004; ETAS, 2004; National Instruments, 2004], provide a broad range of capabilities, including support for:

- system modeling and design where, e.g., effects due to constant or varying delays can be investigated in simulation,
- rapid control prototyping (RCP), allowing control designs to be prototyped using general purpose controller hardware,
- code generation from control system models,
- RTOS configuration and integration within the design models,
- analysis of quantization effects in, e.g., fixed-point implementations,
- testing of models, generated code, and final implementations, and
- calibration of target systems, e.g., over CAN.

The use of code generation has increased significantly over the last few years in the vehicle industry. For example, Volvo Car Corporation is using Simulink models in the design of power train controllers, including simulation and rapid prototyping. Code generated from the models is used in

the final product [Lygner, 2002]. Here, the code generator design environment acts as an interface between control designers and implementation engineers.

The TrueTime Simulator

Papers V and VI present TrueTime, which is a MATLAB/Simulink-based simulation tool that facilitates integrated simulation of the temporal behavior of multitasking real-time kernels. TrueTime also makes it possible to simulate various models of communication network protocols and their influence on networked control loops. The kernel and network simulation is integrated with the standard dynamic-system simulation support provided by Simulink. Furthermore, TrueTime has recently been extended to support simulation of wireless network protocols, energy consumption, and local clocks with offsets and drifts, allowing full simulation of networked embedded systems interacting with their environment. The TrueTime development has been ongoing since 1999, and an early version of the simulator was presented in [Eker and Cervin, 1999].

TrueTime Usage. The main use of TrueTime is for simultaneous simulation of all aspects of distributed real-time control applications. By cosimulation of continuous process dynamics, task execution in real-time kernels, and network communication, it is possible to evaluate the performance of control loops subject to the constraints of the target system.

In a typical scenario, a controller design has been performed (without considering implementation constraints) and is about to be implemented on the target system. In this case, TrueTime can be used to evaluate different real-time implementations, and the effects of CPU and network scheduling, task attributes, etc., on the control performance. For a given implementation architecture, TrueTime may also be used to obtain temporal statistics that can be used as constraints in the design of the controller.

In the ideal scenario, however, the controller and architectural designs are performed at the same time. Here, TrueTime provides a convenient framework for integrated control and real-time design. TrueTime may be used in all stages of the development process, from the early stages and system specifications, during the actual system construction, and finally for testing and validation.

However, TrueTime is currently used more as an experimental platform than as a development tool. This includes research on flexible approaches to real-time implementation and scheduling of controller tasks and design of networked control systems. Using TrueTime, it is straightforward to simulate feedback scheduling algorithms and compensating controllers. TrueTime also simplifies simulation of event-triggered systems, such as, e.g., combustion engine controllers.

Listing 1.2 Example of a TrueTime code function. The code is divided in segments with associated execution times. This way the user can choose to simulate the code with an arbitrary time granularity.

```
function [exectime,data] = myController(segment,data)
switch segment,
  case 1,
    data.y = ttAnalogIn(1);
    data = calculateOutput(data);
    exectime = 0.002;
  case 2,
    ttAnalogOut(1,data.u);
    data = updateState(data);
    exectime = 0.003;
  case 3,
    exectime = -1; % finished
end
```

Level of Abstraction. The TrueTime tool implements a real-time kernel very similar to the kernels found in commercial real-time systems. It contains a ready queue and a time queue for tasks, supports task synchronization and resource access using events and monitors, and provides a number of real-time primitives that may be called from the task code. The main difference between the TrueTime kernel and ordinary real-time kernels and RTOSs is the execution of task code. Whereas real kernels and RTOSs execute statements, TrueTime code is simulated on a time granularity that is chosen by the user.

An example of a TrueTime code function is given in Listing 1.2. The user code is divided in segments, where the code of each segment is executed instantaneously during simulation. The code can interact with other tasks and with the environment at the beginning of each code segment, e.g., using `ttAnalogIn` and `ttAnalogOut` primitives. This execution model makes it possible to model input-output latencies, blocking when accessing shared resources, etc. Technically it would, for instance, be possible to simulate very fine-grained details occurring at the machine instruction level, such as race conditions. However, that would require a large number of code segments.

The code function returns the execution time of the executed segment, and the next segment is not called before the task has been running for the time associated with the previous segment. This means that preemption by higher-priority activities and interrupts may cause the actual delay between the execution of segments to be longer than the execution time.

In the same way that code execution is not modelled by execution of individual statements, the network transmissions are not modelled on bit level. Rather, only the interactions between nodes relevant for the timing behavior of the transmissions are modelled. That includes pre- and post-processing delays, collision detection and collision avoidance mechanisms, and probabilities of lost packets. Transmission times are determined by the bit rate and the size of the message.

Educational and Industrial Use. TrueTime is available for free download[1], and has attracted a number of academic and industrial users during recent years. As an example, TrueTime is currently used for real-time systems education in three of the major Swedish technical universities.

The Embedded Systems Institute in the Netherlands has used TrueTime to build a multi-disciplinary system-level model of an OCE printer [van den Bosch and van de Waal, 2005]. The model and analysis focused on trade-offs between timing and power consumption. In [Hooman *et al.*, 2004], TrueTime is combined with the UML-based CASE tool Rose RealTime, for evaluation of tool coupling for model-based simulation.

Bosch has used TrueTime to examine the impacts of new time-triggered communication protocols on distributed vehicle control system dynamics [Albert *et al.*, 2005]. For this purpose, TrueTime was extended to incorporate the TTCAN and FlexRay communication protocols.

TrueTime Examples

The TrueTime introduction is here concluded with two examples on how the tool can be used for co-simulation of continuous-time dynamics and implementation-related issues. The examples are intended to illustrate the broad number of areas in which TrueTime can be used.

Test Case Generation. [Nilsson and Henriksson, 2005] presents an add-on, Flextime, to the TrueTime simulator, aimed at supporting automated analysis and mutation-based test case generation for flexible real-time control systems. Mutation operators are used to systematically transform original task set specifications, which are then evaluated using TrueTime. Both timeliness failures for hard tasks, and control performance failures for adaptive tasks are considered.

Figure 1.11 gives an overview of how Flextime is used together with other tools to perform automated test case generation. As seen in the figure, a task set specification is supplied as input to mutation operators, and the mutated task set specifications are used as input to Flextime. The simulation traces are fed through a genetic algorithm, which searches for arrival patterns and combinations of task parameters that lead to failures.

[1]TrueTime is available for download at http://www.control.lth.se/user/dan/truetime

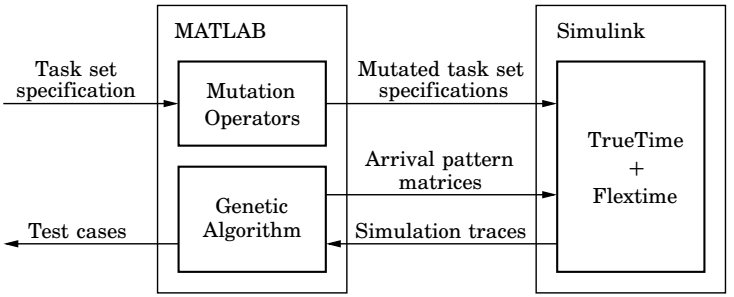

Figure 1.11 Flextime tool usage.

Flextime adapts TrueTime to do efficient simulation of Timed Automata with Tasks (*TAT*) system models [Nordström *et al.*, 1999]. Within *TAT*, Timed Automata (*TA*) [Alur and Dill, 1994] are used for specifying activation patterns of tasks, i.e., the points in time when a task is requested for execution. Each task is described as a quadruple (c, d, SEM, $PREC$), where c is the required execution time and d is the relative deadline of the task. SEM specifies lock and unlock times of semaphores used by the task, and $PREC$ specifies precedence constraints between the task and other tasks.

Flextime also extends TrueTime to support structured parametrization of simulations in terms of tasks and the parameters described above. Furthermore, TrueTime is extended with more advanced resource access protocols, such as the immediate priority inheritance protocol [Sha *et al.*, 1990], and the stack resource protocol [Baker, 1991] under EDF.

Flextime was used to simulate a real-time system with fixed priorities and shared resources under the immediate priority inheritance protocol. The task set consisted of three adaptive periodic tasks implementing flexible controllers for balancing three inverted pendulums. Further, the system had four sporadic real-time tasks with hard deadlines, assumed to implement logic for responding to frequent but irregular events, for example, external interrupts or network messages. The system also had two resources that must be shared with mutual exclusion between tasks.

The study showed that mutation operators for testing of timeliness also can be used to produce mutants that cause control failures in flexible control systems. The Flextime tool makes it possible to use existing mutation-based testing criteria while exploiting the TrueTime ability to interact with continuous-time Simulink blocks modeling the environment.

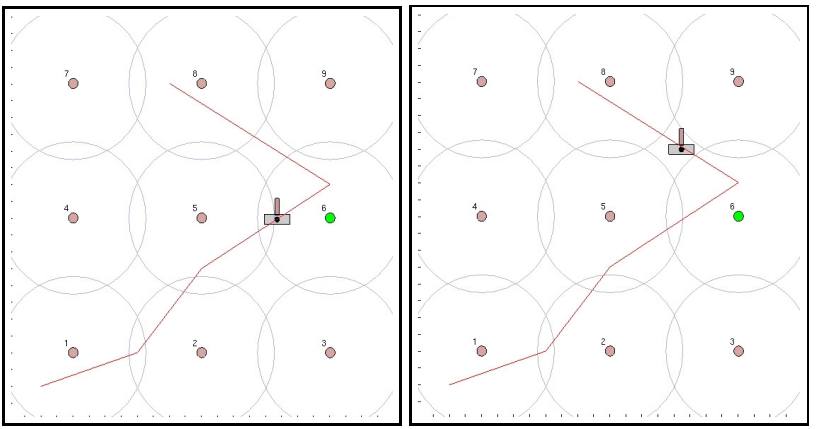

Figure 1.12 Wireless distributed control of an inverted pendulum on a cart. The cart is moving along a pre-specified trajectory in an area of nine symmetrically distributed controller nodes. In the left figure a switch between node 5 and node 6 takes place. In the right figure the cart enters an area without coverage.

Control over Wireless Network. The second example is intended to give an example of the wireless simulation capabilities of TrueTime. The simulation scenario treats an inverted pendulum on a cart moving around in an area of distributed nodes (controllers). Each node consists of a small computer (TrueTime kernel) and communicates using radio. The cart itself is also modeled as a wireless node, augmented with continuous-time dynamics of the pendulum and the cart.

The linearized model of the pendulum and the cart is given by

$$\begin{aligned} \frac{d^2\theta}{dt^2} &= \theta + u, \\ \frac{d^2x}{dt^2} &= 10u, \end{aligned} \tag{1.16}$$

where θ is the pendulum angle and x is the cart's one-dimensional position. The cart is assumed to be moving along a line, i.e., no dynamics for its orientation are included. The control law is of state feedback type and assumes full state measurements,

$$u = -l_1\theta - l_2\dot{\theta} - l_3(x - x_r) - l_4\dot{x}. \tag{1.17}$$

The cart should move along a predefined trajectory according to Figure 1.12. The control loop is closed over the wireless network, with the cart sending periodic samples to the closest controller node. The controller

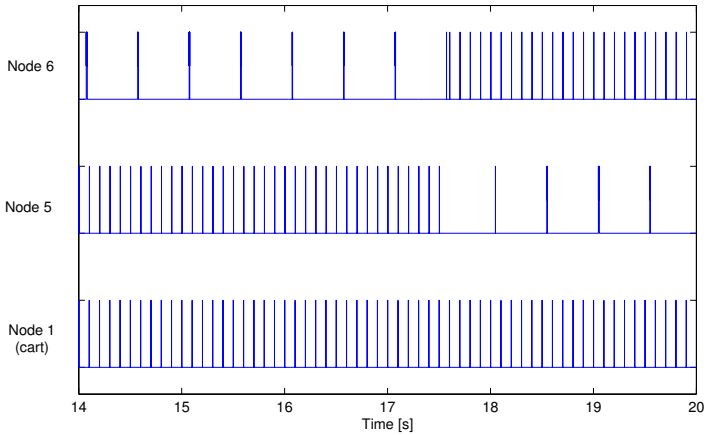

Figure 1.13 Communication schedule showing the transmissions of node 1 (the cart), node 5 and node 6. The sampling period of the control loop is 100 ms and ping messages are sent every 500 ms. A switch between node 5 and node 6 occurs at time $t = 17.5$.

node computes the control signal and sends it back to the cart, where it is actuated. The controller nodes periodically send out ping signals, telling the cart where they are located. The cart, knowing its own position, then decides which controller is the closest one.

The simulation was animated using MATLAB graphics and Figure 1.12 shows two interesting situations as the cart is moving along the path. The circles around the nodes show the distance at which the transmitted signal is no longer possible to detect (receiver signal threshold).

In the left figure, the cart is about to switch between controller 5 and controller 6. This can be seen in the closeup of the communication schedule in Figure 1.13. Here it seen that the ping messages are sent with a period of 500 ms, whereas the sampling period of the control loop is 100 ms. During the switching phase, the controller may, thus, lose up to five samples before it detects that it should switch to another controller node. Various types of hand-over techniques could be used to avoid this problem. Another problem related to the switching is the propagation of states between the controller nodes. In the case of observer-based control, the current state estimate should ideally be kept updated in all nodes.

As can be seen in Figure 1.12, the distributed controllers do not have full coverage of the area in which the cart is moving. In the right part of the figure the cart is entering one such area. The resulting control performance and control signal are shown in Figure 1.14.

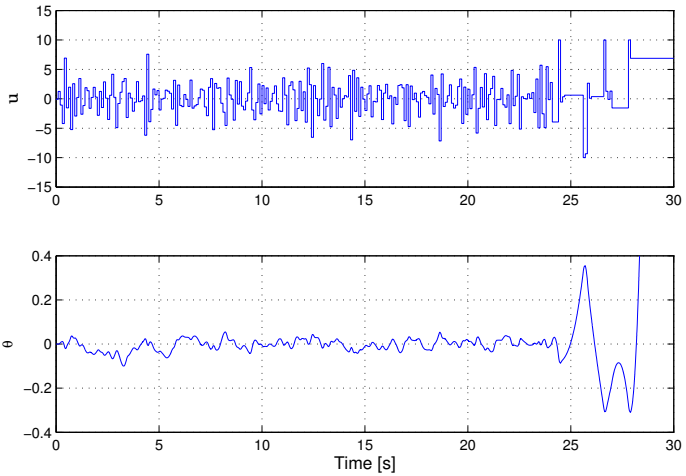

Figure 1.14 Control signal, u, and pendulum angle, θ, as the cart moves along the path shown in Figure 1.12. At around $t = 25$, the cart enters a region without wireless coverage, and the pendulum falls down.

Figure 1.15 The Telos B mote is a low power wireless module, equipped with an 8 MHz Texas Instruments MSP430 microcontroller with 10 kByte RAM and 48 kByte Flash memory.

The presented scenario has also been investigated in a real setup. A three-wheeled robot (the RBbot) with an inverted pendulum was constructed and equipped with a Moteiv Telos B mote for sensing, local control, and wireless communication [Moteiv, 2005], see Figures 1.15 and 1.16. Successful remote control was implemented, where the robot moved around in a network of Telos B motes, over which the control loops were closed. Vision feedback was used for localization. The Telos B motes use the IEEE 802.15.4 ZigBee standard for the communication, a protocol that is also available for simulation in TrueTime.

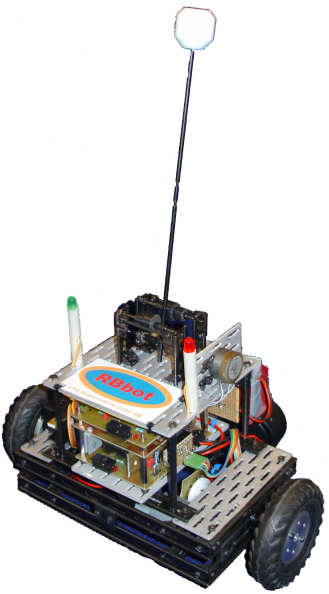

Figure 1.16 The RBbot is an inverted pendulum on a three-wheeled cart equipped with a Telos B mote for sensing, local control and communication with other nodes in a controller network. Constructed by Rolf Braun and Anders Blomdell, Department of Automatic Control, Lund.

2

Feedback Scheduling for Cooperative Robots

2.1 Introduction

This chapter describes dynamic resource allocation for a set of controller tasks in a cooperative robot application. The scenario has been developed as a demonstrator within a national research program, and is intended to show the application of feedback scheduling in a complex setup and how it can be simulated using the TrueTime simulator.

The robot setup is taken from the Robotics Laboratory at the Department of Automatic Control in Lund, and consists of two ABB industrial robot manipulators of type IRB6 and IRB2000, with the Open Robot Control System Architecture [Nilsson, 1996]. The IRB6 and IRB2000 robots have five and six degrees of freedom, respectively. The IRB2000 is also equipped with a six-degrees-of-freedom force/torque sensor.

The robots cooperate in a ball-and-beam scenario, see Figure 2.1, where a beam is held between the end-effectors of the robots. The objective is to control the angle of the beam in order to balance and position the ball along the beam. In the most advanced scenario, this should be done while moving the beam along pre-specified trajectories.

The controller structure consists of a ball balancing controller, a contact force controller, and the individual robot joint controllers. The ball balancing controller computes a desired beam angle based on vision measurements of the ball position. The desired beam angle is translated to robot joint angle references, taking into account the constraints of the beam between the robots and restricted joint velocities. The force controller then modifies the joint references based on force feedback.

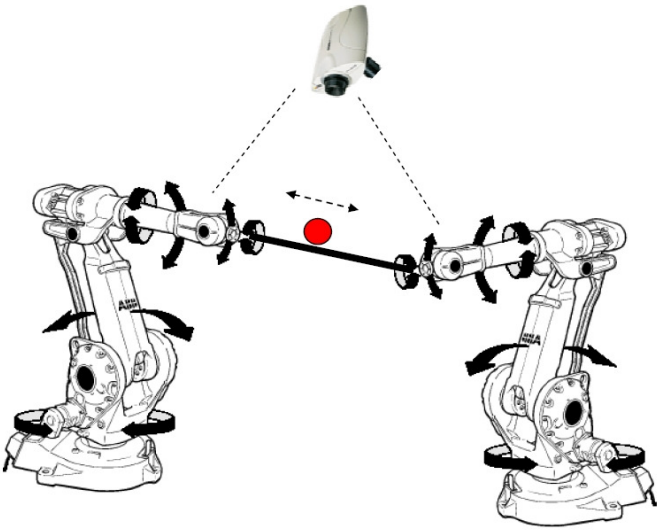

Figure 2.1 The cooperative ball-and-beam robot task.

The scenario is divided in three modes of operation with varying resource requirements. A feedback scheduler is designed, which modifies the sampling periods of the joint controller tasks based on measurements of actual resource consumption and feedforward from mode changes. The velocity Jacobian is used to dynamically incorporate the coupling between the joint control loops in the scheduling decision process.

2.2 The Robot Systems

The IRB2000 robot is shown to the right in Figure 2.2, and consists of seven links connected by three cylindrical and three revolute joints. Joint one turns the robot around its base, whereas joints two and three move the lower and upper arms, respectively. Joint four turns the wrist unit, and joint five bends the wrist around is centre. A force sensor is mounted at the wrist of the robot between the end-effector and joint six, which is a revolute joint at the tip of the wrist. The force sensor measures forces in the x-, y-, and z-directions as well as the corresponding torques.

The IRB6 robot, Figure 2.2 left, has a simpler structure than the IRB2000 robot, and consists of six links connected by five joints. Joint one turns the robot around its base, and joints two and three move the

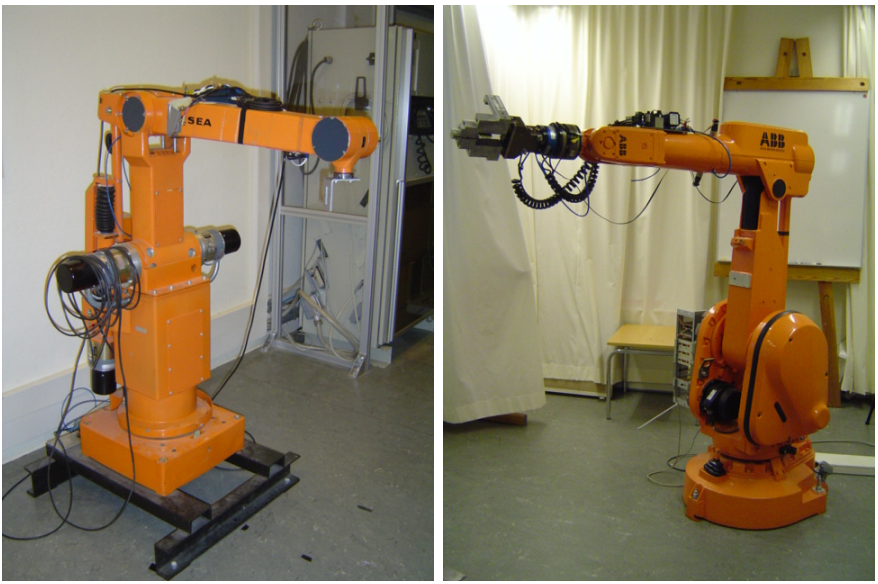

Figure 2.2 The IRB6 industrial robot (left) and the IRB2000 industrial robot (right) with wrist-mounted force/torque sensor.

lower and upper arms. Joint four turns and joint five bends the wrist unit. IRB6 only has five degrees of freedom and can, thus, not reach all points in the workspace with arbitrary orientation.

The positions of the robots relative to each other in the lab are shown in Figure 2.3. IRB2000 is located 1.15 meters in the x-direction and -2.15 meters in the y-direction of the coordinate frame of IRB6. It is also rotated $\pi/4$ radians around the z-axis.

A master and slave configuration is used for the robots, in which the master commands the trajectory to be followed and the slave follows using force control. The IRB2000 robot is chosen as slave, since it is equipped with a force sensor to enable the force feedback-based modifications of the trajectories commanded by the master robot. IRB2000 also has a much larger workspace since it has six degrees of freedom. The robots will be referred to as master and slave in the sequel.

2.3 Design

The ball-and-beam application is simulated in MATLAB/Simulink using the TrueTime simulator, and visualized using Java3D graphics, see Figure 2.3. The visualization environment is developed at the Department of

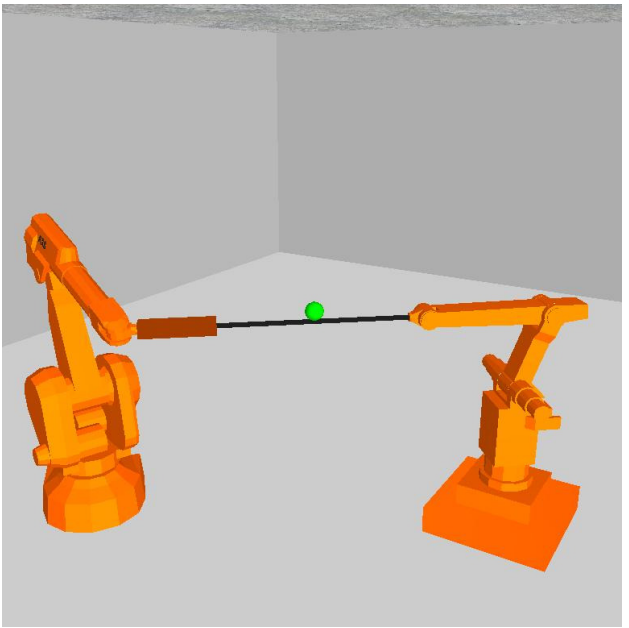

Figure 2.3 The cooperative robot task visualized using Java 3D graphics.

Computer Science, Lund University [Haage, 2004], and is written using the Eclipse platform [Eclipse Foundation, 2005]. The robot visualization tool contains Java classes for the robots and their kinematics. The simulated robot trajectories are communicated from MATLAB to the Java3D visualization program over TCP/IP.

In the simulation it is assumed that the ball position can be directly measured. In the real setup this will be achieved using camera feedback. It is further assumed that the beam is held by spherical joints at the endpoints, to avoid the beam breaking due to shear forces. This is incorporated in the force model used for simulation.

Controller Structure

The controller structure of the master robot simply consists of the individual joint controllers that are fed with references corresponding to certain trajectories. The slave robot, on the other hand, has a more advanced controller structure, since this robot is responsible for both controlling the beam angle and complying to the movements using force control. The controller structure for the slave robot is depicted in Figure 2.4, and consists of three cascaded controller components: the ball position controller, the force controller, and the individual joint controllers.

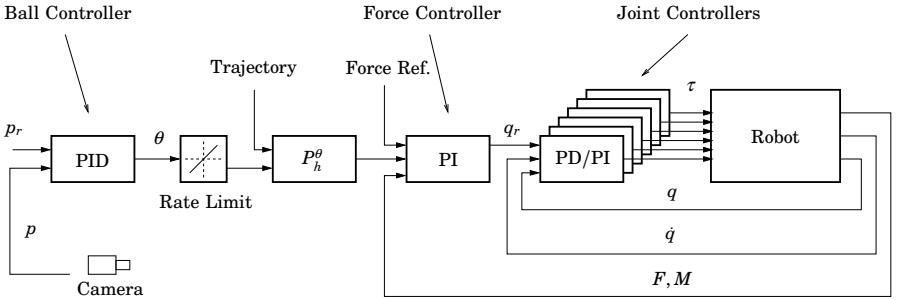

Figure 2.4 Controller structure for the slave robot.

Ball Position Control. The ball position controller is a PID-controller that uses measurements of the ball position along the beam. The output of the controller is the desired angle of the beam, which is fed through a rate limiter to constrain the speed of the robot.

To obtain the desired beam angle, the joint reference angles of the robot need to be modified according to the transformation shown in Figure 2.5. Seen in the coordinate frame of the end-effector, the beam movement corresponds to translations in the y- and z-directions, and a rotation about the x-axis. The translations are given (see Figure 2.5) as

$$\begin{aligned} \Delta y &= l \cdot \sin\,\theta, \\ \Delta z &= l - l \cdot \cos\,\theta, \end{aligned} \tag{2.1}$$

where l is the length of the beam and θ is the desired beam angle. The transformation from the horizontal beam position to the position corresponding to the angle θ (see Figure 2.5) is denoted P_h^θ and is given by

$$P_h^\theta = T(0, \Delta y, \Delta z) \cdot R_x(\theta)\,, \tag{2.2}$$

where the rotation and translation matrices are given by

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\,\theta & -\sin\,\theta & 0 \\ 0 & \sin\,\theta & \cos\,\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad T(0, \Delta y, \Delta z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The resulting transformation matrix is translated to joint reference angles using the inverse kinematics for the slave robot.

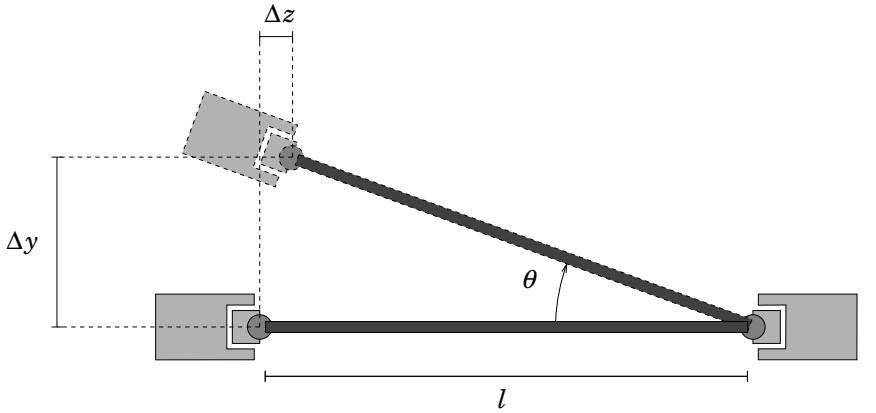

Figure 2.5 Moving the beam an angle, θ, from the horizontal position corresponds to translations, Δy and Δz, in the y- and z-directions, and a rotation, θ, about the x-axis.

Force Control. Due to the joint dynamics and disturbances there are always small deviations between the true joint angles and the desired joint angles as computed by the ball controller. These deviations result in forces along the beam. To control the compression/tension in the beam, the joint reference angles are therefore modified by a force controller based on measurements of the contact forces at the IRB2000 end-effector.

The force control scheme is direct force control, which aims at obtaining specified reference forces and torques along the different dimensions. The force controller is a PI-controller acting on the difference between the desired and measured forces. The output of the controller is a modification of the robot trajectory in Cartesian coordinates, which is then converted to joint space to obtain modified references for the joint control loops. Design and evaluation of various force control schemes for the cooperative robot task have been investigated in [Chong, 2005].

Joint Control. The individual joint controllers are cascaded PD and PI loops for the position and velocity control, respectively. In the simulations, the dynamics of the robot joints are modeled as second order systems,

$$\ddot{q}_i + D_i \dot{q}_i = k_i \tau_i \,, \tag{2.3}$$

where τ_i is the input torque and q_i is the angle of joint i. The parameter D_i represents the damping in the joint, and the flexibility in the gear box is neglected. The controllers are designed to give a critically damped response of the closed-loop system, with a time-constant around 0.1 seconds.

Task Decomposition. In the simulation of the cooperative robot application it is assumed that both the master and slave joint control loops are implemented on the same CPU, together with the ball position controller and the force controller. Consequently, a total of 13 tasks are used, with one task for each joint controller (five for IRB6 and six for IRB2000), one force control task, and one task for the ball position control.

The sampling interval of the ball position control loop is restricted by the camera frame rate, which typically lies around 20–30 Hz. The sampling rate was chosen to 20 Hz, and the force controller was designed with the same sampling rate. A nominal sampling rate of 200 Hz was chosen for the joint controllers, which is sufficient for the closed-loop dynamics described above. In the real robotics lab, however, these loops run at 4 kHz, mainly to improve the disturbance rejection and to deliver references to the current loops of the AC-motors.

Scenarios

The application is divided in three different scenarios (modes), each with different resource requirements in terms of active real-time tasks.

1. In the first scenario, the objective is simply to position the ball along the beam. To obtain a desired angle of the beam only the slave robot is used, treating the master robot simply as a beam-end fixture.
2. In the second scenario, the objective is to move the beam along a trajectory, ignoring the ball position. In this case all robot joint controllers are switched on, and the ball position controller is switched off.
3. In the third scenario, the objective is to position the ball along the beam and move the beam along a trajectory simultaneously. In this scenario all controllers are switched on.

The Feedback Scheduler

The objective of the feedback scheduler is to adjust the sampling periods of tasks in the system, such that the overall control performance is maximized, while meeting constraints on the system utilization. The standard formulation of this problem, Equation (1.6), in general assumes that the individual cost functions, $J_i(h_i)$, are only related to the performance of control loop i. However, for coupled control tasks, such as the joint control loops of the slave robot, another dimension is added to the resource allocation problem. The control performance should be evaluated in terms of the overall robot behavior, and in this case the individual joints may be more

or less important. To cope with this, the following weighted formulation of the feedback scheduling problem will be used,

$$\begin{aligned} &\min_{h_1 \ldots h_n} \sum_{i=1}^{n} w_i V_i(h_i), \\ &\text{subj. to } \sum_{i=1}^{n} \frac{C_i}{h_i} \leq U_{sp}, \end{aligned} \tag{2.4}$$

where the weight factors, w_i, reflect the relative importance of each joint control loop. The cost functions are here denoted V_i to avoid confusing them with the velocity Jacobian, J, introduced below.

No state dependencies are included in the cost functions, since the results from Paper I are not directly applicable to the cascaded PD/PI-controllers used for the joint control. Linear approximations,

$$V_i(h_i) \approx \alpha_i + \gamma_i h_i, \tag{2.5}$$

of the infinite-horizon cost functions are used, whereby the closed-form solution to the optimization problem (1.6) is given as [Cervin, 2003]:

$$\frac{1}{h_i^\star} = f_i^\star = \left(\frac{\gamma_i}{C_i}\right)^{1/2} \frac{U_{sp}}{\sum_{j=1}^{n} (C_j \gamma_j)^{1/2}}. \tag{2.6}$$

For the weighted problem (2.4), we instead get

$$\frac{1}{h_i^\star} = f_i^\star = \left(\frac{\tilde{\gamma}_i}{C_i}\right)^{1/2} \frac{U_{sp}}{\sum_{j=1}^{n} (C_j \tilde{\gamma}_j)^{1/2}}, \tag{2.7}$$

where $\tilde{\gamma}_i = w_i \gamma_i$.

Choosing the Weights — The Velocity Jacobian. The velocity Jacobian is used to map joint velocities to the linear and angular velocities of the end-effector. The relation is expressed as

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = J\dot{q} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \dot{q}, \tag{2.8}$$

where J is the velocity Jacobian, v is the linear velocity, and ω is the angular velocity. The velocity Jacobian is a function, $J = J(q)$, of the joint angles, and changes as the robot moves.

The velocity Jacobian can also be used to approximately express the relation between small changes in the joint angles, Δq, and small changes in the Cartesian space translation and rotation, ΔT and ΔR,

$$\begin{pmatrix} \Delta T \\ \Delta R \end{pmatrix} = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \Delta q. \tag{2.9}$$

For the overall robot performance it is more relevant to minimize a performance criterion related to ΔT and ΔR, rather than the joint angle deviations, Δq. Using the notation

$$\Delta T = \begin{pmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \end{pmatrix}, \quad \Delta R = \begin{pmatrix} \Delta r_x \\ \Delta r_y \\ \Delta r_z \end{pmatrix}, \tag{2.10}$$

a reasonable cost function could be defined as

$$\bar{V} = \mathrm{E}\left\{ \int_0^{Nh} \Big(\Delta t_x^2 + \Delta t_y^2 + \Delta t_z^2 + \rho^2 \cdot (\Delta r_x^2 + \Delta r_y^2 + \Delta r_z^2) \Big) dt \right\}, \tag{2.11}$$

where ρ^2 is a scale factor expressing the relation between translation and rotation errors. Using the velocity Jacobian, the cost function (2.11) can be given as a function of the joint angle deviations, Δq. We obtain

$$\begin{aligned} \bar{V} &= \mathrm{E}\left\{ \int_0^{Nh} \Big(\begin{pmatrix} \Delta T \\ \rho \cdot \Delta R \end{pmatrix}^T \begin{pmatrix} \Delta T \\ \rho \cdot \Delta R \end{pmatrix} \Big) dt \right\} \\ &= \mathrm{E}\left\{ \int_0^{Nh} \Big(\Delta q^T \begin{pmatrix} J_v \\ \rho \cdot J_\omega \end{pmatrix}^T \begin{pmatrix} J_v \\ \rho \cdot J_\omega \end{pmatrix} \Delta q \Big) dt \right\}. \end{aligned} \tag{2.12}$$

Based on this equation, a heuristic choice of the weights, w_i, in Equation (2.4), may be to use the diagonal elements of the matrix

$$\bar{J} = \begin{pmatrix} J_v \\ \rho \cdot J_\omega \end{pmatrix}^T \begin{pmatrix} J_v \\ \rho \cdot J_\omega \end{pmatrix}. \tag{2.13}$$

The weights change dynamically as the robot moves, affecting the optimal sampling periods computed on-line by the feedback scheduler using (2.7). The Jacobian can be pre-computed for a range of position/orientation points and stored in a look-up table. Since some force control schemes use the velocity Jacobian, another possibility would be to communicate the matrix between the force control task and the feedback scheduler.

2.4 Simulation Results

The simulation model is shown in Figure 2.6 and consists of a TrueTime controller node, the robot dynamics, and simulation models for the contact forces and the ball position.

The contact force model is a spring-damper model adopted from [Chong, 2005] and is given as

$$\begin{aligned} F_x &= K_x \cdot \Delta x + D_x \cdot \Delta \dot{x}, \\ F_y &= K_y \cdot \Delta y + D_y \cdot \Delta \dot{y}, \\ F_z &= K_z \cdot \Delta z + D_z \cdot \Delta \dot{z}, \\ M_x &= K_\gamma \cdot \Delta \gamma, \\ M_y &= K_\beta \cdot \Delta \beta, \\ M_z &= K_\alpha \cdot \Delta \alpha. \end{aligned} \tag{2.14}$$

The deviations in Cartesian position, Δx, Δy, and Δz, and the deviations in roll/pitch/yaw angles, $\Delta\gamma$, $\Delta\beta$, and $\Delta\alpha$, are determined by expressing the position and orientation of the master robot end-effector in the coordinate frame of the end-effector of the slave robot. This transformation is performed assuming a beam of length 1000 mm between the end-effectors and a relative orientation between the robots as expressed by

$$R = \begin{pmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}. \tag{2.15}$$

The spring and damping constants are chosen to model a flexible beam. In the simulations they are chosen as

$$\begin{aligned} K_x &= K_y = K_z = 4 \text{ N/mm}, \\ D_x &= D_y = D_z = 0.1 \text{ Ns/mm}. \end{aligned} \tag{2.16}$$

To model the spherical joints at the end-effectors, the spring constants, K_γ, K_β, and K_α, for the moment equations are chosen as zero.

The robot joint angles are transformed to Cartesian coordinates, which are used to compute the beam angle. The transfer function from beam angle, in degrees, to the ball position, in meters, is given by

$$G_{\theta \to p} = \frac{-0.055}{s^2}. \tag{2.17}$$

A 30 second simulation is performed, starting in mode 1, switching to mode 2 at $t = 10$, and to mode 3 at $t = 20$. In modes 1 and 3, the

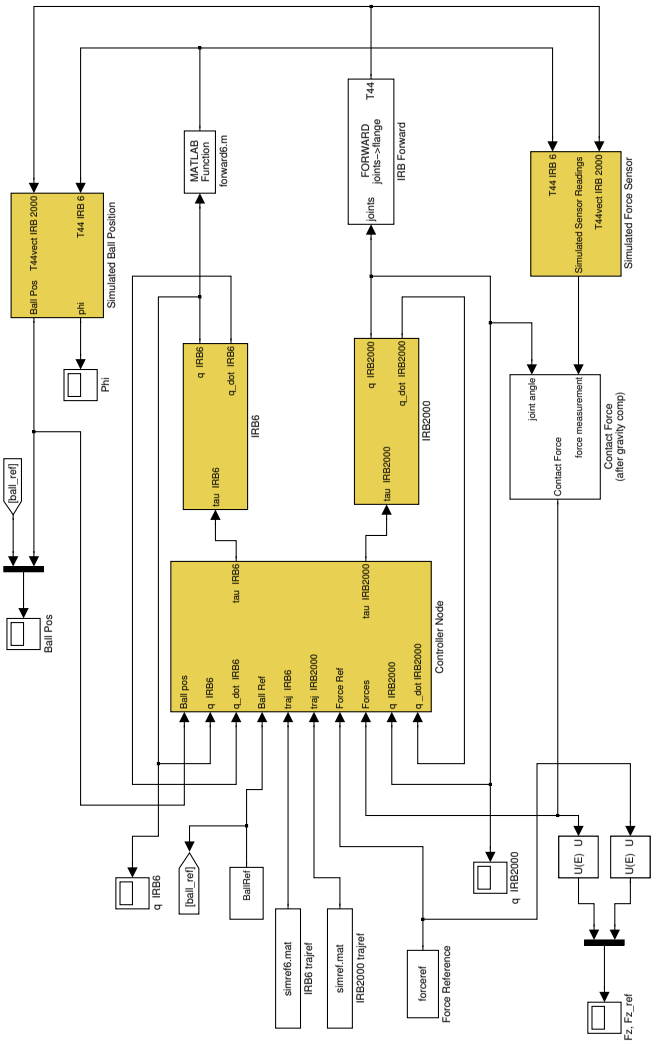

Figure 2.6 Simulink model of the cooperative robot task, consisting of a TrueTime controller node and models of the robot dynamics, contact forces and ball position.

ball should follow a square-wave reference with amplitude 0.25 meters and period 10 seconds. The trajectory to be followed in modes 2 and 3 consists of a circle in the y-z plane of the coordinate frames of the robot end-effectors. The force control references in the x-, y-, and z-directions are zero.

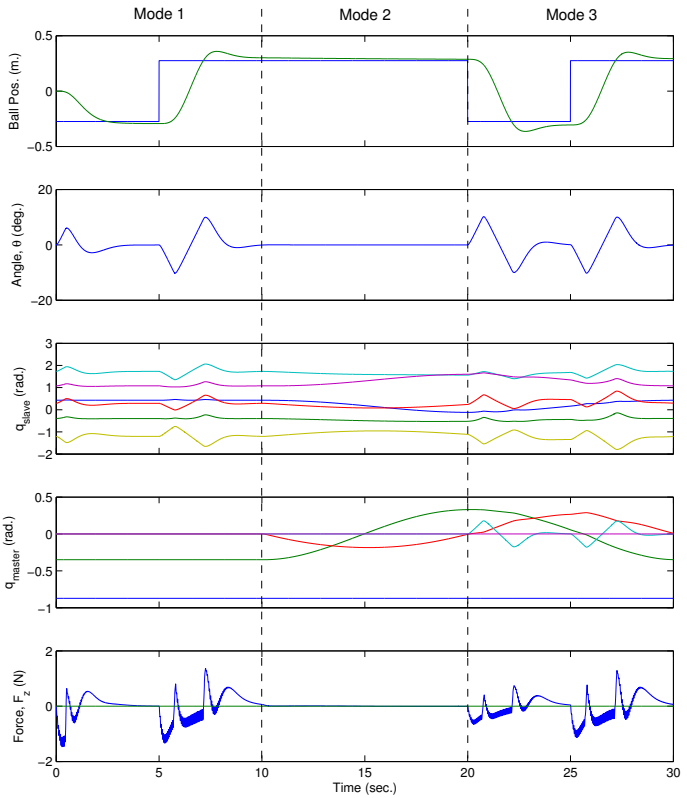

Figure 2.7 Results from an ideal simulation.

Ideal Simulation

Figure 2.7 shows the results of an ideal simulation, where the execution times of all tasks are set to zero. The first plot shows the ball position and reference and the second plot shows the beam angle. The next two plots show the joint angles of the two robots. The last plot shows the force in the z-direction of the force sensor, i.e., along the beam.

Open-Loop Scheduling

In the next simulation, the simulated execution time for the joint controller tasks is set to $C_{joint} = 400$ μs. The ball and force controllers are more advanced and contain coordinate transformations and forward and inverse kinematics computations. These loops are modelled with a ten times longer execution time, $C_{ball,force} = 4$ ms. The tasks are scheduled us-

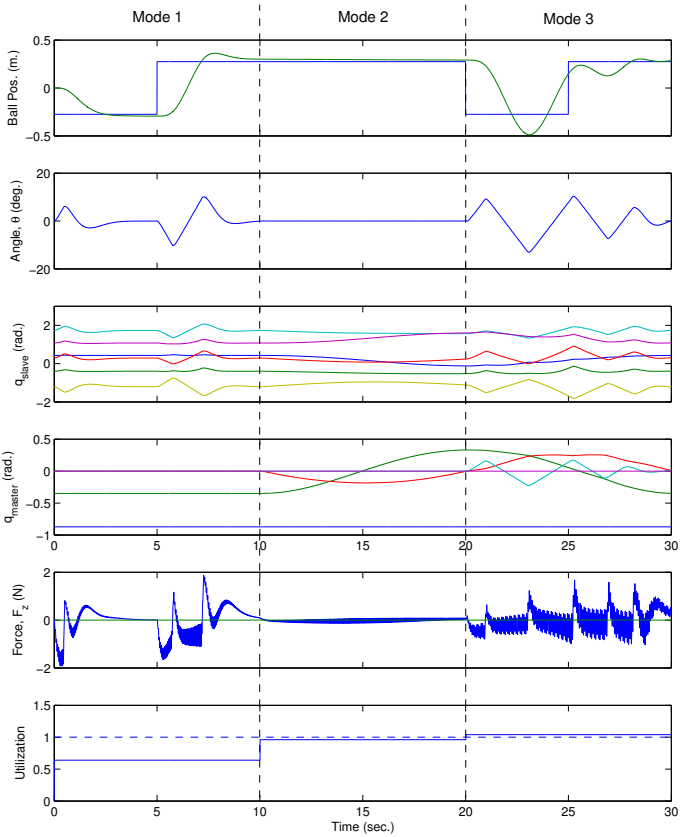

Figure 2.8 Results using rate-monotonic scheduling. The system becomes overloaded in modes 2 and 3 with utilization close to and over 100 percent. The deteriorated control performance is clearly seen in, e.g., the plot of the ball position.

ing rate-monotonic scheduling, which gives the joint control tasks higher priority than the ball position and force control tasks.

Figure 2.8 shows the results of this simulation, with the bottom plot showing the system utilization increasing by each mode change. In mode 1, the joint controllers of the slave robot and the ball and force controllers are active, giving a utilization of

$$U = 6 \cdot \frac{0.0004}{0.005} + 2 \cdot \frac{0.004}{0.05} = 0.64. \tag{2.18}$$

The utilization increases to 0.96 in mode 2 and to 1.04 in mode 3. As a

consequence, the control loops miss deadlines and experience long delays. The deteriorated control performance is clearly seen in the plots of the ball position and beam angle. The ball position and force controllers have the lowest priority and suffer the most due to the overload.

Based on knowledge of the execution times, a worst-case design could have been used to compute the sampling periods. According to Equation (1.2) for $n = 13$ tasks, the utilization bound for guaranteed schedulability becomes 0.71. Assuming unmodified periods of the ball and force control tasks, this corresponds to a sampling period for the joint controller tasks of $h = 0.008$. This unnecessarily long sampling interval would, on the other hand, result in a utilization of only 0.46 in mode 1.

Feedback Scheduling

In a third simulation, a feedback scheduling task is introduced, running with a period of 200 ms. The feedback scheduler has highest priority, whereas the application tasks are again scheduled using rate-monotonic scheduling. The utilization set-point for the feedback scheduler is chosen to 0.8. The results from this simulation are shown in Figure 2.9. It can be seen that the feedback scheduler keeps the utilization level around the set-point except for transient overloads during the mode changes. By increasing the sampling periods of the joint controller tasks, the performance of the ball control is improved significantly. The reduced performance of the joint control is seen as somewhat increased force signal oscillations.

Figure 2.10 shows the sampling periods of the slave robot joint controllers as functions of time. The sampling periods increase by each mode change but also vary slowly within each mode as the position and orientation of the robot change. For the initial robot configuration shown in Figure 2.3, we have (for $\rho = 600$)

$$\frac{\bar{J}}{\bar{J}_{11}} = \begin{pmatrix} 1.00 & -0.075 & 0.014 & -0.15 & 0.50 & 0.00 \\ -0.075 & 0.66 & -0.43 & 0.052 & -0.059 & 0.00 \\ 0.014 & -0.43 & 1.56 & -0.10 & -0.10 & 0.41 \\ -0.15 & 0.052 & -0.10 & 0.48 & 0.00 & 0.22 \\ 0.50 & -0.059 & -0.10 & 0.00 & 0.49 & 0.00 \\ 0.00 & 0.00 & 0.41 & 0.22 & 0.00 & 0.47 \end{pmatrix}, \qquad (2.19)$$

which gives an indication of the relative importance of the individual joint control loops for the overall cost defined by (2.11). The effect of the dynamic weighting is seen in Figure 2.10, where the fastest sampled task in each mode corresponds to the controller for joint three. It should, however, be noted that the matrix also contains some non-negligible off-diagonal elements, which are neglected in this heuristic choice of weights.

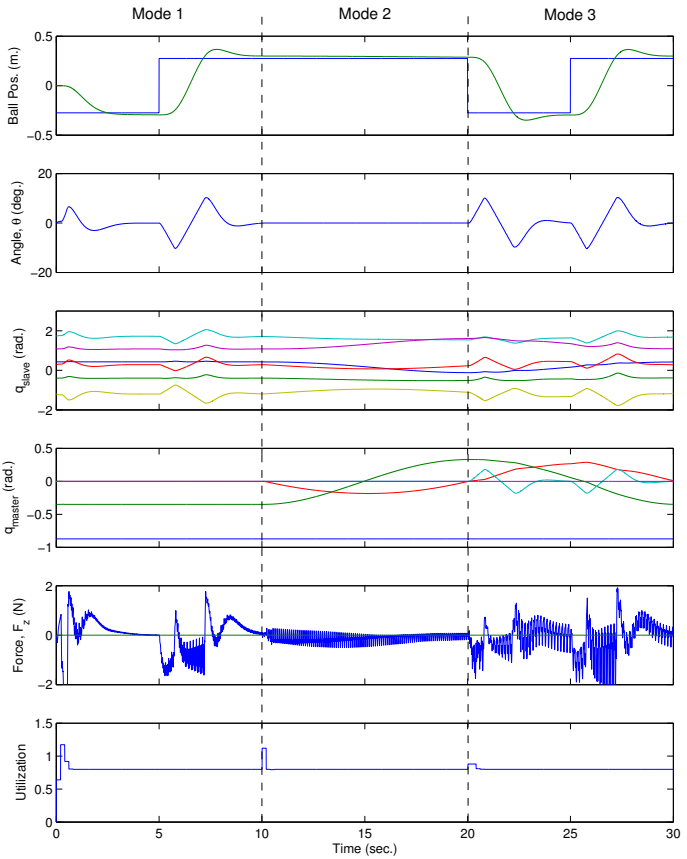

Figure 2.9 Results using feedback scheduling. The feedback scheduler regulates the utilization to the set-point of 80 percent.

Finally, a 20 second simulation is run to quantify the performance of the modified feedback scheduling strategy. The simulation is run in mode 3 and with the reference for the ball position set to zero, i.e., at the center of the beam. The beam trajectory and the task execution times are the same as in the previous simulations. The cost function (2.11) is used to evaluate the performance of the modified feedback scheduling strategy compared to the standard feedback scheduling formulation. Two simulations are run, one using equal weights of the cost functions, and one using dynamic weights based on the velocity Jacobian. Evaluations of the cost function (2.11) for the two cases are given in Figure 2.11.

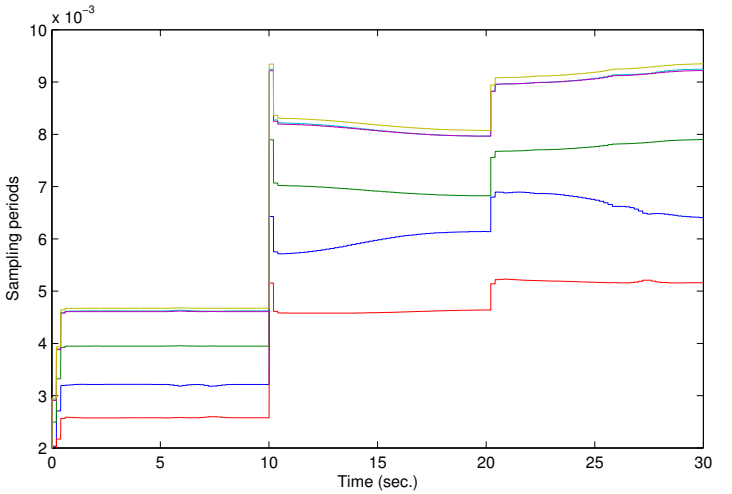

Figure 2.10 The sampling periods of the slave robot joint controllers as functions of time. The periods increase by each mode change, but also vary within each mode as the position/orientation of the robot changes.

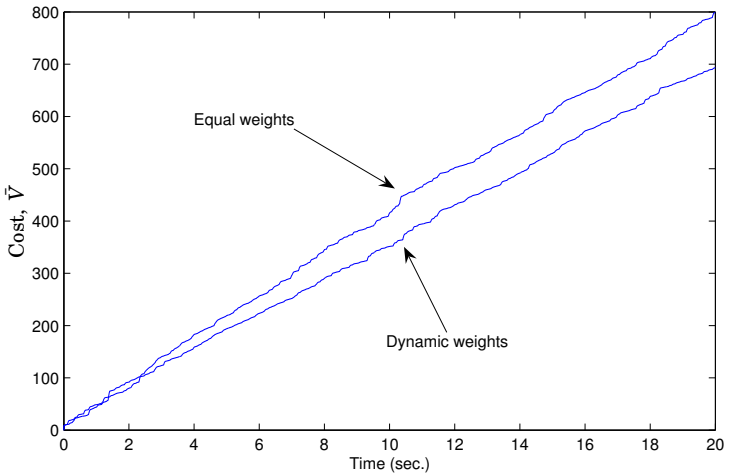

Figure 2.11 Evaluation of the cost function (2.11) for the two feedback scheduling formulations: equally weighted cost functions and dynamically weighted cost functions based on the velocity Jacobian.

2.5 Summary

This chapter has demonstrated the co-simulation capabilities of TrueTime, with complex cascaded control algorithms implemented as several tasks on a single CPU interacting with the dynamics of the robots and the dynamic models of the ball position and contact force. The example can be further extended to also simulate the network communication between sensor, controller, and actuator nodes that exist in the real robotics lab. The real-time networking solution in the lab is based on switched Ethernet.

The chapter also described a heuristic extension to the standard constrained optimization problem formulation, Equation (1.6), for optimal sampling period selection. In the modified approach, the cost functions were weighted dynamically to incorporate the relative importance of each loop for the overall performance of the robot. The dynamic weights were based on the velocity Jacobian, which relates incremental changes in the end-effector position and orientation to incremental changes of the individual joint angles.

The open robot architecture in the robotics lab is based on Linux RTAI [The RTAI Project, 2004], a hard real-time Linux kernel extension. To be able to test the simulated resource allocation schemes, a prototype feedback scheduler has been implemented for RTAI as a student project [Schmid and Knutsson, 2004]. The implementation focuses on limited modifications to the RTAI core. The only functionality that is implemented in RTAI code is measurements of the periodic runtime of the tasks.

The feedback scheduler module consists of a periodic task, which in each invocation uses the task execution time measurements to compute the current CPU load. The task periods are then recalculated based on the CPU load using linear rescaling. Incorporation of more advanced feedback scheduling schemes is straightforward.

3

Future Work

Resource-constrained embedded control and computing systems is an active research area, and the work presented in this thesis can be extended in several directions. A few suggestions for possible extensions and open questions are given below.

Feedback Scheduling

The basic feedback scheduling optimization formulation could be extended to also include more involved combinations of tasks. One example would be to combine control tasks of anytime nature with ordinary control tasks where the control performance decreases monotonically with the input-output latency and sampling interval. In this case it is not trivial how to assign the computation resources. It could also be possible to directly control timing parameters such as delays and jitter.

The approach to feedback scheduling based on plant states could be improved by introducing more realistic process/noise descriptions and cost functions. Deviations in the state due to, e.g., reference changes with imperfect plant models, could be modeled using non-stationary noise processes. This would allow the plant state to have a larger impact on the scheduling decisions.

More general feedback scheduling structures could be developed, such as, e.g., hierarchical or cascaded structures. In these schemes, a global controller could be used to assign dynamic reservations to a set of virtual sub-CPUs, and local feedback schedulers are designed to enforce the utilization constraints of the sub-CPUs.

A further possibility includes a direct approach to feedback scheduling, where the scheduling decisions are made based on instantaneous cost functions for the different control tasks. What is the best way to design the cost functions and how should the resulting event-based system be analyzed?

Implementation Techniques for MPC

The work on scheduling of MPC tasks may be extended in a number of ways. One question is what happens if the task has not been terminated at the deadline. Is it then better to abort and output the control signal or to continue the optimization into next sample. Another extreme is when there is no execution time available at all. In this case it may be possible to use the previous solution sequence shifted one step. The stability issue of MPC under time-varying delays and using sub-optimal solutions is another challenging area.

Web Server Control

Model predictive control has also been suggested as a promising approach for control of web server systems. This requires accurate models of the behavior of the server systems. For high load, flow models work well, whereas medium load and rapidly changing traffic is more challenging. In many situations it may also be required to combine event- and time-based approaches. Another important area for control of computer systems is to provide RTOS and middle-ware support to aid in the development. Finally, the predictor presented in this thesis should be extended to also treat the admission control case.

Wireless Sensor/Actuator Networks

Wireless sensor/actuator networks is an emerging area containing many challenging control and real-time problems, mainly motivated by the severe resource constraints in these systems. Issues in this area include, development of specially tailored communication protocols, control of network bandwidth, and communication-aware control design. Almost all applications within this field are wireless sensor networks, designed merely for information gathering. It remains to be seen whether a killer application for wireless sensor *and* actuator networks will be found.

TrueTime Extensions

The TrueTime kernel could be extended and be made more realistic. Currently it is possible to simulate context switch overhead, but the kernel model could also include interrupt latencies and execution times associated with the various real-time primitives. One major limitation with TrueTime is the question of how to assign the execution times of tasks. One possibility would be to integrate TrueTime with available compiler and execution time analysis tools. Support for code generation and the possibility to import production code, e.g. using the POSIX standard, should also be added.

References

Abdelzaher, T. F., E. M. Atkins, and K. Shin (2000): "QoS negotiation in real-time systems and its application to automated flight control." *IEEE Transactions on Computers*, **49:11**, pp. 1170–1183.

Abdelzaher, T. F. and C. Lu (2001): "Schedulability analysis and utilization bounds for highly scalable real-time services." In *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium*. TaiPei, Taiwan.

Abdelzaher, T., Y. Lu, R. Zhang, and D. Henriksson (2004): "Practical application of control theory to web services." In *Proceedings of the American Control Conference*. Boston, MA, USA.

Abdelzaher, T. F., J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu (2003): "Feedback performance control in software services." *IEEE Control Systems Magazine*, **23:3**, pp. 74–90.

Abeni, L. and G. Buttazzo (1998): "Integrating multimedia applications in hard real-time systems." In *Proceedings of the 19th IEEE Real-Time Systems Symposium*. Madrid, Spain.

Abeni, L., L. Palopoli, G. Lipari, and J. Walpole (2002): "Analysis of a reservation-based feedback scheduler." In *Proceedings of the 23rd IEEE Real-time Systems Symposium*. Austin, TX, USA.

Agnew, C. E. (1976): "Dynamic modeling and control of congestion-prone systems." *Operations Research*, **24:3**, pp. 400–419.

Albert, A., P. Pietsch, and F. Voetz (2005): "Simulation environment for investigating the impacts of time-triggered communication on a distributed vehicle dynamics control system." In *Proceedings of 1st International Workshop on Real-Time and Control*. Palma de Mallorca, Spain.

Alur, R. and D. L. Dill (1994): "A theory of timed automata." *Theoretical Computer Science*, **126:2**, pp. 183–235.

Andersson, M., D. Henriksson, A. Cervin, and K.-E. Årzén (2005): "Simulation of wireless networked control systems." In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*. Seville, Spain.

Årzén, K.-E., A. Cervin, and D. Henriksson (2003): "Resource-constrained embedded control systems: Possibilities and research issues." In *Proceedings of CERTS'03 — Co-design of Embedded Real-Time Systems Workshop*. Porto, Portugal.

Årzén, K.-E., A. Cervin, and D. Henriksson (2005): "Implementation-aware embedded control systems." In Hristu-Varsakelis and Levine, Eds., *Handbook of Networked and Embedded Control Systems*.

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

Audsley, N., A. Burns, M. Richardson, and A. Wellings (1994): "STRESS—A simulator for hard real-time systems." *Software—Practice and Experience*, **24:6**, pp. 543–564.

Baker, T. P. (1991): "Stack-based scheduling of real-time processes." *The Journal of Real-Time Systems*, **3:1**, pp. 67–99.

Balbastre, P., I. Ripoll, J. Vidal, and A. Crespo (2004): "A task model to reduce control delays." *Real-Time Systems*, **27:3**, pp. 215–236.

Beccari, G., S. Caselli, M. Reggiani, and F. Zanichelli (1999): "Rate modulation of soft real-time tasks in autonomous robot control systems." In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*. York, England.

Bohrer, P., E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony (2002): *The Case for Power Management in Web Servers*. Power Aware Computing, Kluwer Academic Publications.

Boje, E. (2005): "Approximate models for continuous-time linear systems with sampling jitter." *Automatica*, **41:12**, pp. 2091–2098.

Bouyssounouse, B. and J. Sifakis (2005): *Embedded Systems Design – The ARTIST Roadmap for Research and Development*. Springer-Verlag.

Buttazzo, G., G. Lipari, and L. Abeni (1998): "Elastic task model for adaptive rate control." In *Proceedings of the 19th IEEE Real-Time Systems Symposium*. Madrid, Spain.

CAN in Automation (2005): "TTCAN – time-triggered communication on CAN." Home page, http://www.can-cia.org/can/ttcan/.

Casile, A., G. Buttazzo, G. Lamastra, and G. Lipari (1998): "Simulation and tracing of hybrid task sets on distributed systems." In *Proceedings of the 5th IEEE International Conference on Real-Time Computing Systems and Applications*. Hiroshima, Japan.

Cervin, A. (1999): "Improved scheduling of control tasks." In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*. York, England.

Cervin, A. (2003): *Integrated Control and Real-Time Scheduling*. PhD thesis ISRN LUTFD2/TFRT--1065--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Cervin, A., J. Eker, B. Bernhardsson, and K.-E. Årzén (2002): "Feedback-feedforward scheduling of control tasks." *Real-Time Systems*, **23:1–2**, pp. 25–53.

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén (2003): "How does control timing affect performance?" *IEEE Control Systems Magazine*, **23:3**, pp. 16–30.

Cervin, A., B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo (2004): "The jitter margin and its application in the design of real-time control systems." In *Proceedings of the 10th International Conference on Real-Time and Embedded Computing Systems and Applications*. Göteborg, Sweden.

Chong, C. Y. (2005): "Cooperative robots." Master's thesis ISRN LUTFD2/TFRT--5741--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Chung, J.-Y., J. W. S. Liu, and K.-J. Lin (1990): "Scheduling periodic jobs that allow imprecise results." *IEEE Transactions on Computers*, **39:9**, pp. 1156–1174.

Crovella, M. E. and A. Bestavros (1997): "Self-similarity in world wide web traffic: Evidence and possible causes." *ACM/IEEE Transactions on Networking*, **5:6**, pp. 835–846.

Cucinotta, T., L. Palopoli, L. Marzario, G. Lipari, and L. Abeni (2004): "Adaptive reservations in a Linux environment." In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*. Toronto, Canada.

Davidson, C. (1973): *Random Sampling and Random Delays in Optimal Control Systems*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden.

dSPACE (2004): "Solutions for control." Home page, http://www.dspace.de

Eaton, J. W. (1998): "OCTAVE." Home page, http://www.octave.org/.

Eclipse Foundation (2005): "Eclipse." Home page, http://www.eclipse.org.

Eker, J. and A. Blomdell (2000): "A contract-based language for embedded control systems." In *Proceedings of the 25th IFAC/ IFIP Workshop on Real-Time Programming*. Palma de Mallorca, Spain.

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*. Hong Kong, P.R. China.

Eker, J., P. Hagander, and K.-E. Årzén (2000): "A feedback scheduler for real-time control tasks." *Control Engineering Practice*, **8:12**, pp. 1369–1378.

El-Khoury, J. and M. Törngren (2001): "Towards a toolset for architectural design of distributed real-time control systems." In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*. London, England.

ETAS (2004): "Engineering products and services." Home page, http://www.etasgroup.com.

Gestegård Robertz, S. (2003): "Flexible automatic memory management for real-time and embedded systems." Technical Report Licentiate Thesis. Department of Computer Science, Lund Institute of Technology, Lund, Sweden.

Haage, M. (2004): "Flexible interaction with productive robots in partly unstructured environments." Technical Report Licentiate Thesis. Department of Computer Science, Lund Institute of Technology, Lund, Sweden.

Halbwachs, N. (1993): *Synchronous Programming of Reactive Systems*. Kluwer.

Hanselmann, H. (1987): "Implementation of digital controllers – a survey." *Automatica*, **23:1**, pp. 7–32.

Hellerstein, J. L., Y. Diao, S. Parekh, and D. M. Tilbury (2004): *Feedback Control of Computing Systems*. Wiley-IEEE Press.

Henriksson, D. (2003): "Flexible scheduling methods and tools for real-time control systems." Licentiate thesis ISRN LUTFD2/TFRT--3233--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henriksson, D. and J. Åkesson (2004): "Flexible implementation of model predictive control using sub-optimal solutions." Technical Report ISRN LUTFD2/TFRT--7610--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henriksson, D. and A. Cervin (2003): "TrueTime 1.1—Reference manual." Technical Report ISRN LUTFD2/TFRT--7605--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henriksson, D. and A. Cervin (2004): "Multirate feedback control using the TinyRealTime kernel." In *Proceedings of the 19th International Symposium on Computer and Information Sciences*. Antalya, Turkey.

Henriksson, D. and A. Cervin (2005): "Optimal on-line sampling period assignment for real-time control tasks based on plant state information." In *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*. Seville, Spain.

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002): "Feedback scheduling of model predictive controllers." In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium*. San Jose, CA, USA

Henriksson, D., A. Cervin, J. Åkesson, and K.-E. Årzén (2002): "On dynamic real-time scheduling of model predictive controllers." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV, USA

Henriksson, D., A. Cervin, and K.-E. Årzén (2002): "TrueTime: Simulation of control loops under shared computer resources." In *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Henriksson, D., A. Cervin, and K.-E. Årzén (2003): "TrueTime: Real-time control system simulation with MATLAB/Simulink." In *Proceedings of the Nordic MATLAB Conference*. Copenhagen, Denmark.

Henriksson, D., Y. Lu, and T. Abdelzaher (2004): "Improved prediction for web server delay control." In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*. Catania, Sicily, Italy.

Henriksson, D. and T. Olsson (2004): “Maximizing the use of computational resources in multi-camera feedback control.” In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*. Toronto, Canada.

Henriksson, D., O. Redell, J. El-Khoury, M. Törngren, and K.-E. Årzén (2005): “Tools for real-time control systems co-design—a survey.” Technical Report ISRN LUTFD2/TFRT--7612--SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Henzinger, T. A., B. Horowitz, and C. M. Kirsch (2001): “Embedded control systems development with Giotto.” In *Proceedings of the International Conference on Languages, Compilers, and Tools for Embedded Systems*. Snowbird, UT, USA.

Hooman, J., N. Mulyar, and L. Posta (2004): “Supporting model-based simulation of embedded systems by coupling tools.” In *Proceedings of the 5th PROGRESS Symposium on Embedded Systems*. Nieuwegein, The Netherlands.

Hylands, C., E. Lee, J. Liu, X. Liu, S. Neuendorffer, Y. Xiong, Y. Zhao, and H. Zheng (2003): “Overview of the Ptolemy project.” Technical Report UCB/ERL M03/25. Department of Electrical Engineering and Computer Science, University of California Berkeley, CA.

IEEE (1999): “ANSI/IEEE Std 802.11.”

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

Kalman, R. E. and J. E. Bertram (1959): “A unified approach to the theory of sampling systems.” *Journal of the Franklin Institute*, **267:5**, pp. 405–436.

Kleinrock, L. (1975): *Queueing Systems Volume 1: Theory*. John Wiley & Sons.

Kopetz, H. (1997): *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer.

Lavarenne, C., O. Seghrouchni, Y. Sorel, and M. Sorine (1991): “The Syndex software environment for real-time distributed systems design and implementation.” In *Proceedings of the European Control Conference*. Grenoble, France.

Leung, J. Y. T. and J. Whitehead (1982): “On the complexity of fixed-priority scheduling of periodic, real-time tasks.” *Performance Evaluation*, **2:4**, pp. 237–250.

Lian, F.-L., J. R. Moyne, and D. M. Tilbury (2005): "Network protocols for networked control systems." In Hristu-Varsakelis and Levine, Eds., *Handbook of Networked and Embedded Control Systems*.

Lincoln, B. (2003): *Dynamic Programming and Time-Varying Delay Systems*. PhD thesis ISRN LUTFD2/TFRT--1067--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Lincoln, B. and A. Cervin (2002): "Jitterbug: A tool for analysis of real-time control performance." In *Proceedings of the 41st IEEE Conference on Decision and Control*. Las Vegas, NV, USA.

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. 46–61.

Liu, J. and E. Lee (2003): "Timed multitasking for real-time embedded software." *IEEE Control Systems Magazine*, **23:1**, pp. 65–75.

Liu, J. W. S., K.-J. Lin, W.-K. Shih, A. Yu, J.-Y. Chung, and W. Zhao (1991): "Algorithms for scheduling imprecise computations." *IEEE Transactions on Computers*, **24:5**, pp. 58–68.

Liu, J. W. S., W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung (1994): "Imprecise computations." *Proceedings of the IEEE*, **82:1**, pp. 83–94.

Locke, C. D. (1992): "Software architecture for hard real-time applications: Cyclic executive vs. fixed priority executives." *Real-Time Systems*, **4:1**, pp. 37–53.

Lu, C., J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley (2000): "Performance specifications and metrics for adaptive real-time systems." In *Proceedings of the 21st IEEE Real-Time Systems Symposium*. Orlando, FL, USA.

Lu, C., J. A. Stankovic, S. H. Son, and G. Tao (2002): "Feedback control real-time scheduling: Framework, modeling and algorithms." *Real-time Systems*, **23:1/2**, pp. 85–126.

Lu, C., J. A. Stankovic, G. Tao, and S. H. Son (1999): "Design and evaluation of a feedback control EDF scheduling algorithm." In *Proceedings of the 20th IEEE Real-Time Systems Symposium*. Phoenix, AZ, USA.

Lygner, M. (2002): "Model-based development tool chain at Volvo Cars." dSPACE News, 1/2002, http://www.dspace.de.

Martí, P., C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes (2004): "Optimal state feedback based resource allocation for resource-constrained control tasks." In *Proceedings of the 25th IEEE Real-Time Systems Symposium*. Lisbon, Portugal.

Martinsson, A. (2002): “Scheduling of real-time traffic in a switched ethernet network.” Master’s thesis ISRN LUTFD2/TFRT--5683--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Moteiv (2005): “Moteiv: Wireless sensor networks.” Home page, http://www.moteiv.com/products.php.

National Instruments (2004): “Test and measurement.” Home page, http://www.ni.com.

Nilsson, J., B. Bernhardsson, and B. Wittenmark (1998): “Stochastic analysis and control of real-time systems with random time delays.” *Automatica*, **34:1**, pp. 57–64.

Nilsson, K. (1996): *Industrial Robot Programming*. PhD thesis ISRN LUTFD2/TFRT--1046--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Nilsson, R. and D. Henriksson (2005): “Test case generation for flexible real-time control systems.” In *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation*. Catania, Sicily, Italy.

Norberg, J. and M. Törngren (2003): “Fault injection into control algorithms.” Technical Report TRITA–MMK 2003:37, ISSN 1400–1179, ISRN KTH/MMK/R-03/11-SE. Department of Machine Design, KTH, Sweden.

Nordström, C., A. Wall, and W. Yi (1999): “Timed automata as task models for event-driven systems.” In *Proceedings of 6th International Conference on Real-Time Computing Systems and Applications*. Hong Kong, P.R. China.

Palopoli, L., L. Abeni, F. Conticelli, M. Di Natale and G. Buttazzo (2000): “Real-time control system analysis: An integrated approach.” In *Proceedings of the 21st IEEE Real-Time Systems Symposium*. Orlando, FL, USA.

Palopoli, L., L. Abeni, and G. Lipari (2003): “On the application of hybrid control to CPU reservations.” In *Proceedings of Hybrid systems, Computation and Control (HSCC03)*. Prague, The Czech Republic.

Palopoli, L., G. Lipari, G. Lamastra, L. Abeni, G. Bolognini, and P. Ancilotti (2002): “An object-oriented tool for simulating distributed real-time control systems.” *Software – Practice and Experience*, **32:9**, pp. 907–932.

Pernet, N. and Y. Sorel (2003): "Optimized implementation of distributed real-time embedded systems mixing control and data processing." In *Proceedings of the ISCA 16th International Conference: Computer Applications in Industry and Engineering*. Las Vegas, NV, USA.

Redell, O., J. El-Khoury, and M. Törngren (2004): "The AIDA tool-set for design and implementation analysis of distributed real-time control systems." *Journal of Microprocessors and Microsystems*, **28:4**, pp. 163–182.

Robertsson, A., B. Wittenmark, and M. Kihl (2003): "Analysis and design of admission control in web-server systems." In *Proceedings of the 2003 American Control Conference*. Denver, CO, USA.

Robertsson, A., B. Wittenmark, M. Kihl, and M. Andersson (2004): "Admission control for web server systems - design and experimental evaluation." In *Proceedings of the 43rd IEEE Conference on Decision and Control*. Paradise Island, Bahamas.

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

Sanfridsson, M. (2004): *Quality of Control and Real-time Scheduling*. PhD thesis, Royal Institute of Technology, Sweden.

Schinkel, M., W.-H. Chen, and A. Rantzer (2002): "Optimal control for systems with varying sampling rate." In *Proceedings of the 2002 American Control Conference*. Anchorage, AK, USA.

Schmid, E. and P. Knutsson (2004): "Feedback scheduling in RTAI." Project in Real-Time Systems, Department of Automatic Control, Lund Institute of Technology, Sweden.

Seto, D., J. P. Lehoczky, L. Sha, and K. G. Shin (1996): "On task schedulability in real-time control systems." In *Proceedings of the 17th IEEE Real-Time Systems Symposium*. Washington, DC, USA.

Sha, L., X. Liu, Y. Lu, and T. Abdelzaher (2002): "Queuing model based network server performance control." In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*. Austin, TX, USA.

Sha, L., R. Rajkumar, and J. P. Lehczky (1990): "Priority inheritance protocols: An approach to real-time synchronization." *IEEE Transactions on Computers*, **39:9**, pp. 1175–1185.

Sharma, S. and D. Tipper (1993): "Approximate models for the study of nonstationary queues and their applications to communication networks." In *Proceedings of IEEE International Conference on Communications*. Geneva, Switzerland.

Sharma, V., A. Thomas, T. Abdelzaher, and K. Skadron (2003): "Power-aware QoS management in web servers." In *Proceedings of the 24th IEEE Real-Time Systems Symposium*. Cancun, Mexico.

Shin, K. and C. Meissner (1999): "Adaptation and graceful degradation of control system performance by task reallocation and period modification." In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*. York, England.

Simon, D. and A. Girault (2001): "Synchronous programming of automatic control applications using Orccad and Esterel." In *Proceedings of the 40th IEEE Conference on Decision and Control*. Orlando, FL, USA.

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*. York, England.

Storch, M. F. and J. W.-S. Liu (1996): "DRTSS: A simulation framework for complex real-time systems." In *Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium*. Boston, MA, USA.

The FlexRay Consortium (2005): "FlexRay – the communication system for advanced automotive control applications." Home page, http://www.flexray.com.

The Mathworks (2001a): *Real-Time Workshop; User's Guide*. The MathWorks Inc., Natick, MA, USA.

The Mathworks (2001b): *Simulink: A Program for Simulating Dynamic Systems – User's Guide*. The MathWorks Inc., Natick, MA, USA.

The RoboCup Federation (2004): "RoboCup." Home page, http://www.robocup.org.

The RTAI Project (2004): "RTAI – real-time application interface." Home page, http://www.rtai.org.

The ZigBee Alliance (2004): "ZigBee." Home page, http://www.zigbee.org.

Tiller, M. (2001): *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers.

Tipper, D. and M. K. Sundareshan (1990): "Numerical methods for modeling computer networks under nonstationary conditions." *IEEE Journal on Selected Areas in Communication*, **8:9**, pp. 1682–1695.

van den Bosch, P. F. A. and E. van de Waal (2005): "A case study of multi-disciplinary modeling using MATLAB/Simulink and TrueTime." In *Proceedings of INCOSE 2005 International Symposium*. Rochester, NY, USA.

Wang, W., D. Tipper, and S. Banerjee (1996): "A simple approximation for modeling nonstationary queues." In *Proceedings of IEEE Infocom'96*. San Francisco, CA, USA.

Xu, J. and D. L. Parnas (2000): "Priority scheduling versus pre-run-time scheduling." *Real-Time Systems*, **18:1**, pp. 7–23.

Zhou, K. and J. C. Doyle (1998): *Essentials of Robust Control*. Prentice Hall.