Control Oriented Direct Feedback Scheduling

Feng Xia[1], Xiaohua Dai[1], Youxian Sun[1], and Jianxia Shou[2]

[1] National Laboratory of Industrial Control Technology,
Institute of Modern Control Engineering
Zhejiang University, Hangzhou 310027, China
xia@iipc.zju.edu.cn
[2] College of Software Engineering,
Zhejiang University of Technology, Hangzhou 310014, China
sjx@zjut.edu.cn

Abstract

The performance of multitasking control systems with resource constraints depends not only on controller design, but also on efficient scheduling of the shared computing resources. To support codesign of feedback control and real-time scheduling under overload conditions, a direct feedback scheduling approach is proposed. The goal is to optimize the overall system performance through dynamically allocating limited CPU time based on control requirements. A control application oriented task model is suggested, providing an interface between control performance and task scheduling decision. Based on this model, a novel scheduling algorithm intended for control tasks is presented. The overload condition is effectively handled using a job skipping scheme. Preliminary simulations exhibit the benefits of the proposed approach, with comparison against traditional control systems design methodologies.
Keywords. Direct feedback scheduling, flexible task model, embedded control systems, real-time computing, resource constraint

1 Introduction

With the rapid evolution of embedded computing techniques, more and more control applications are built upon embedded systems. Due to various reasons, these systems often have limited computing capabilities [1,2]. On the other hand, practical control applications have become more and more complex. This makes it common that a set of control tasks reside in one embedded processor. In these cases, control tasks have to compete for the limited CPU time. As a consequence, the overall control performance not only depends on the design of control algorithms (from the control perspective), but also relies on the efficient scheduling of the shared CPU resource (from the computing perspective).

Over the years, many algorithms have been developed for scheduling of real-time tasks, e.g. Rate Monotonic (RM) and Earliest Deadline First (EDF) [3]. These scheduling algorithms are built on fixed timing constraints. They require complete knowledge about execution time, deadline etc. of the task set. In order to provide performance guarantee in predictable environments, *a priori* knowledge about the workload and system characteristics are also needed. Often it is assumed that the execution time and real-time constraints (e.g. period) of a task are precisely known. In real world control applications, however, this hypothesis is often unrealistic, since almost all timing parameters of control tasks are suffering from uncertainty. For example, due to measurement noise, the execution time and period of a control task are always imprecise. These uncertain properties greatly impact the performance of multitasking control systems built upon classical scheduling algorithms. From the control point of view, these algorithms are all *open-loop* [4]. Once established at system set-up, schedules are not dynamically adjusted. Although these scheduling algorithms can perform well in resource sufficient environments, they are prone to yield a highly underutilized system due to pessimistic estimations of workload. In addition, their performance degrades rapidly in overload conditions. A transient overload may cause a critical task to fail, which is certainly undesirable, e.g. in safety-critical control systems. Therefore, for control applications over resource constrained embedded systems, open-loop scheduling algorithms might be unsuitable.

Although several efforts, e.g., optimal feedback scheduling [5], neural feedback scheduling [6], fuzzy feedback scheduling [7,8], robust feedback scheduling [9], and those for anytime controllers [10-12], have been made in the area of feedback scheduling where closed-loop scheduling approaches rather than open-loop ones are explored for control tasks, most of them are indirect [1]. In these works, the quality of control (QoC) is indirectly determined by the execution times or periods of control tasks, which are dynamically adjusted to make sure that the task is schedulable. The performance of these approaches highly relies on accurate formulation of the relationship between tasks' timing parameters and control performance index, which is often a hard work. Additionally, the timing uncertainty of embedded control tasks still remains to be an open issue.

In this paper, we consider an embedded CPU under overload conditions where independent control tasks that feature timing uncertainty compete for the limited computing resources. Two kinds of real-time characteristics, execution time and period, may be concerned by uncertainty. We attempt to develop an adaptive scheduling algorithm using a closed-loop paradigm in order for the overall control performance to be maximized. Particularly, our attention is focused on direct feedback scheduling techniques that are able to make scheduling decisions based on continuous feedback of each controlled process' performance. As an interface of control performance and scheduling decisions, a flexible control task model is suggested. Some task models, e.g. [13-15], have been presented to facilitate co-design of control and scheduling, but none of them directly relates scheduling decision-making to control dynamics as we do. Based on this model, we propose a direct feedback scheduling algorithm [1,16] to online optimize resource allocation for control tasks. The basic principles of the scheduling policies in [17,18] are adopted. The uncertainty of execution time and period is indirectly handled via the direct mapping of control requirements to task scheduling. In order to deal with overload, a job-shipping scheme is suggested. In the context of the flexible control task model, our scheduling algorithm as well as the job-skipping scheme provides run time adaptation of control tasks execution with respect to both quick reactions to perturbations and saving of resources when the system is in steady state.

The rest of this paper is structured as follows. Section 2 suggests a flexible control task model for integrating control and scheduling, with timing characteristics being defined. The direct feedback scheduling algorithm for multitasking control applications, together with a job-shipping scheme for overload handling, is illustrated in Section 3. The proposed approach is evaluated and compared with classical methodologies in Section 4. Section 5 concludes this paper.

2 Flexible Control Task Model

In this section, we suggest a novel task model intended for control applications oriented computing. Before describing the task model, we would like to identify the requirements imposed on a real-time task model for control applications, from a viewpoint of integrating feedback control and real-time scheduling. Although aperiodic tasks may exist in a practical application, only periodic control tasks are of our interest in this paper. The basic idea behind the control task model is to associate the scheduling decisions inside a CPU with the dynamics of controlled processes.

Particularly, it is argued that the model should provide the following properties. First, it should accommodate basic timing characteristics that allow scheduling under classical scheduling algorithms. Secondly, it must reflect the dynamic behaviors of controlled processes, which in turn result in different timing properties. This ensures the final task model is for control purpose. Thirdly, to facilitate integration of control and scheduling, it should provide a simple interface between control requirements and scheduling decisions.

We believe that these properties make the task model suitable for online distribution of computing resources e.g. using feedback scheduling, while being real-time and control oriented at the same time. One may notice that the above requirements specification is different from that in [13]. This is raised from different focuses of them. Our focus in this paper is on control oriented scheduling while that of [13] is on short input-output latency and jitter.

Accordingly, in our flexible control task model, each control task τ_i is characterized by three parameters:

C_i: the execution time, associated with the computational delay in control loop *i*,
T_i: the period given by the sampling period of control loop *i*,
U_i: the *urgency*, a notion we introduce.

In control applications, the deadline of a task is generally assumed to be equal to the sampling period. With this model, the requirements specified above can be easily addressed. Classical scheduling algorithms can be adopted based on the first two parameters (with an additional deadline equal to the period). The model is clearly control oriented, since each parameter is associated with certain property of the control application. In particular, we employ the notion of urgency U_i, which has been employed in our previous work [16], for each task to relate online scheduling decisions and dynamic control requirements. The definition of U_i is given by:

$$U_i = w_i \cdot e_i \,. \qquad \textbf{(1)}$$

The closed-loop system error e_i is defined as the absolute difference between the desired reference and the actual response of the controlled process *i*. It is directly associated with the control performance of the corresponding control loop. The weight coefficient *w* corresponds to the relative criticality of different control loops that are in charge of controlling processes. The basic idea behind the use of weights is that different control loops may play different roles in the whole control application. This situation is common in many fields, such as automotive systems, aircraft, and process control, etc. Although all control tasks run upon the same processor, they may hold different importance in the contribution to final system value from an application or economic viewpoint. The weight coefficients of all tasks are assigned offline and do not change during runtime.

Since the system error e_i is generally time-varying, the value of U_i will change over time. In this sense, our task model is a dynamic model rather than a static one that holds fixed timing constraints. It can be regarded as a dynamic image of control requirements. Both execution time and period can be used to reflect the temporal behavior of the control applications. Control requirements are tightly associated with the parameter urgency of a control task. An increase of urgency means that the control performance goes worse, and vice versa. As shown in Fig.1, the introduction of U_i in the task model provides a runtime interface between instantaneous control requirements and scheduling decisions.

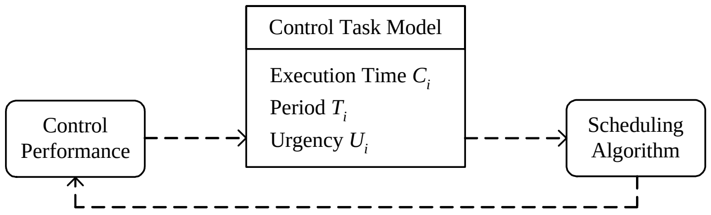

Fig. 1. The control task model for integrating control and scheduling

Recall that the execution time and period of a control task might be uncertain or imprecise. In these cases, the above task model enables a natural alternative that makes the scheduling decisions based on the newly introduced notion, i.e. urgency. We will detail this methodology next.

3 Scheduling Algorithm for Multitasking Control Systems

In this section, we present a direct feedback approach to adaptive scheduling decision making in a multitasking embedded CPU. Based on the control task model presented in Section 2, a direct feedback scheduling algorithm is illustrated. The methodologies in [17,18] and our previous work [16] are extended. And then, a job-shipping scheme is suggested to manage overload conditions.

3.1 Direct Feedback Scheduling

The real-time scheduling algorithm developed is MUF (Maximum Urgency First). As the name implies, the MUF algorithm uses the urgency of a control task as its priority. The task with the largest urgency has the highest priority, while the task with the smallest urgency has the lowest priority. At any time, the scheduler chooses to execute the task with the highest urgency.

In the cases considered, we assume that: 1) all control algorithms to be executed have been well designed without considering any resource constraint so that closed-loop system dynamics are pretty good; 2) both sensor and controller in each control loop are time-triggered, with a fixed sampling period; 3) new samples are always available exactly when new instances of control tasks are invoked; 4) the execution time of each control task is non-deterministic. In resource insufficient embedded environments, the main factor that impacts the control performance will then be the availability of computing resources.

To optimally allocate CPU resources among control tasks, the MUF algorithm follows the emerging methodology of feedback scheduling. From the point of view of feedback control, there exists a feedback loop in the structure of our approach, see Fig.2. The urgency of a task is updated according to current performance of the corresponding process. Consequently, it is used as the priority in MUF. And the scheduling algorithm in turn affects the QoC of all loops (see also Fig.1). In this way, MUF works as a direct feedback scheduling algorithm [1,16]. That is, the scheduler bases the decision of which control task to execute on current control requirements.

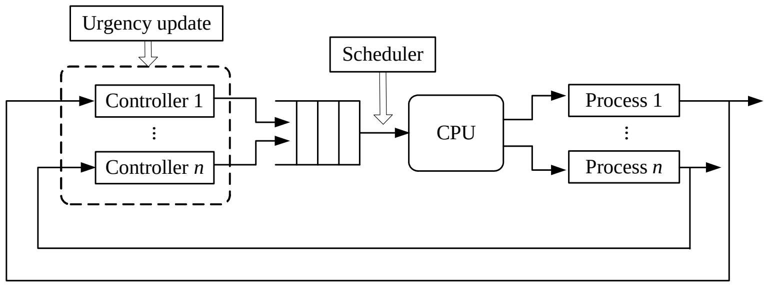

Fig. 2. Direct feedback scheduling structure

Based on our flexible control task model and the MUF algorithm, the proposed scheduling algorithm operates as follows. When the system output is available, it is fed back to the control task, and hence the urgency of the corresponding task is recalculated according to equation (1). We assume that the underlying operating system provides real-time support for this calculating operation. During run time, the scheduler will always make a scheduling decision according to current urgencies of all control tasks. As we can see, since neither execution time nor period is involved in the direct feedback scheduling algorithm, the impact of their uncertain characteristics on task scheduling is consequently removed.

Example 1. In order to illustrate the operation of the direct feedback scheduling methodology, Fig. 3 gives a simple example of two control tasks running on one CPU. For the sake of simplicity, the periods are set to be the same, equal to 5. The execution times are $C_1 = C_2 = 2$. All timing parameters are given in units of time. At t = 0, two control tasks are released at the same time. Since the urgency of τ_1 is initialized to be larger than that of τ_2 in this case, τ_1 starts to execute at t = 0. Once τ_1 completes the first execution at t = 2, a control signal will then be issued onto the relevant controlled process 1 (e.g. via the action of an actuator). Also at t = 2, τ_2 starts its first execution. Once the output of process 1 is available at time t_1, the urgency of τ_1 will immediately be updated by the underlying real-time system. The similar operation will be conducted on τ_2 at time t_2. Consequently, when two tasks are synchronously invoked again at t = 5, the one (τ_1 in this example) that has the maximum urgency will be executed according to the principle of MUF algorithm. And hence, τ_1 is scheduled before τ_2 again in the time segment from t = 5 to 10. A control signal for process 1 will be produced more quickly so that the performance of process 1 can be efficiently improved at the expense of a less urgent delay in loop 2. In this sense, the computing resources are cost-effectively used through distributing computing resources to a task exactly when it most needs. The urgencies of two tasks are updated again at time t_3 and t_4 respectively. At t = 10, there is an exchange in the execution order, i.e., τ_2 is scheduled because its latest urgency (updated at t_4) is larger than that of τ_1 (updated at t_3), implying that a control signal for process 2 is needed more urgently than process 1 at this time. Note that the MUF algorithm can be either preemptive or non-preemptive, although it is not reflected in this example.

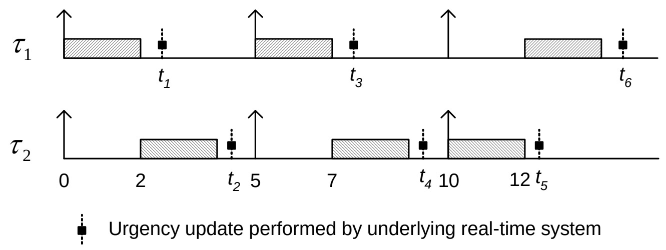

Fig. 3. Execution of two example control tasks under MUF

3.2 Overload Handling

The above example is simple in that the system is not overloaded, while the overload conditions must be effectively handled. Generally, the requested CPU utilization can be scaled down through changing task parameters, e.g. enlarging the period [5-8] or shortening the execution time [10-12]. In the context of direct feedback scheduling, we instead employ a job-skipping [19] scheme. For priority based scheduling systems, lower priority tasks may be queued under overload conditions. For a overloaded task queue, two situations may occur when a new task instance (with a new sample) is invoked: 1) the current instance is executing but not finish; 2) the current instance is still waiting for CPU access and does not start execution. In our scheme, there will be only one single instance of each control task to be queued at any time. For the first situation, the new task instance will be skipped. While for the second situation, the current instance will be discarded. Intuitively, if the previous task instance has completed its execution before the new instance is invoked, the new task instance will be queued consequently. By skipping some instances, which performs like vacant sampling, the requested CPU utilization of low priority control tasks is sure to be cut down. In this way, the overload conditions could be effectively managed.

Combined with the MUF algorithm, the effect of our job-skipping scheme is that more computing resources will be dynamically allocated to more urgent control tasks. This is an intuitive result of their underlying principles. Because lower priorities always correspond to lower urgency in the context of MUF and only the lowest priority task will suffer from job skipping in the considered situations, the CPU utilization of the lowest urgent task will be reduced at any time, thereby providing sufficient CPU resources to higher urgent tasks. Since the task urgencies change with system dynamics, the overload conditions are dealt with in an adaptive way that more urgent control tasks consume more CPU time.

4 Performance Evaluation

In order to evaluate the performance of the proposed direct feedback scheduling approach for multitasking control applications under overload conditions, preliminary simulations are performed based on Matlab/TrueTime [1]. Three different scheduling methodologies, RM, EDF and MUF (denoting here the direct feedback scheduling approach we present), are compared. The underlying real-time kernel is assumed to be priority-preemptive. We consider the case with three PID (Proportional-Integral-Derivative) [20] tasks running concurrently on the same CPU to control three independent DC servo motors. Let the DC motor models be described by the following transfer functions:

$$G_1(s) = \frac{1000}{s^2 + 2s},\ G_2(s) = \frac{1000}{s^2 + s},\ G_3(s) = \frac{800}{s^2 + s}. \tag{2}$$

Once the PID parameters are well-designed pre-runtime, they remain the same in all simulations. The sampling periods of three controllers are fixed and chosen to be

T = [T_1, T_2, T_3] = [9, 7, 6] ms. Their execution times may vary over time, according to the normal distribution with a mean of 3 ms. Let $\overline{C}_i$ denote the average execution time of control task *i*. And hence we have $\mathbf{\overline{C}} = [\overline{C}_1, \overline{C}_2, \overline{C}_3] =$ [3, 3, 3] ms. The weights of these three loops are given as **w** = [w_1, w_2, w_3] = [1, 2, 3].

In each experiment, the cost function given by equation (3) is recorded. One can notice that the cost function in (3) is exactly the sum of the integral of the urgency of each control task. At the same time, it also can be viewed as a weighted sum of the IAE (Integral of Absolute Error) [20] values for the three DC motors. The reasons for choosing this cost function include that the system considered is for control purpose and IAE is a generally used performance index in the control community.

$$J(t) = \sum_{i=1}^{3} (w_i \int_0^t e_i d\tau) \tag{3}$$

4.1 Simulation Results

Each control loop experiences an input step change at the start of the experiment. The overall costs under different scheduling algorithms are given in Fig.4.

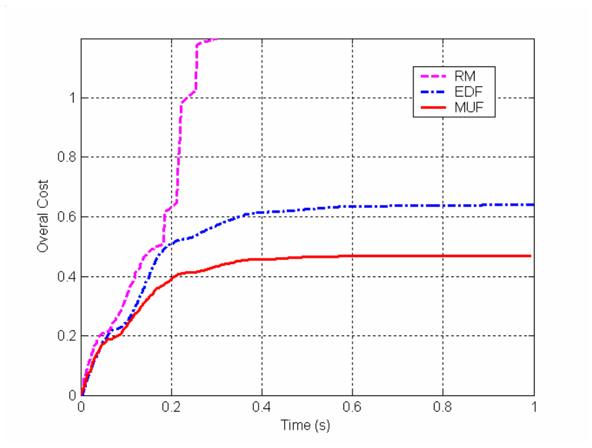

Fig. 4. Overall costs under different scheduling algorithms

Under the RM algorithm, fixed priorities will be assigned according to periods of the tasks, i.e., higher priority will be assigned to task with shorter period. For the RM algorithm, the schedulable bound for three tasks is 0.78. In this experiment, the average CPU utilization of three control tasks can be approximately calculated as $\overline{C}_1/T_1 + \overline{C}_2/T_2 + \overline{C}_3/T_3 = 3/9 + 3/7 + 3/6 = 1.26 > 0.78$, driving the task set to be unschedulable. As shown in Fig.4, the overall cost under RM goes to infinite, implying that the system becomes unstable. We have found that the first DC motor is out of control while the other two DC motors are stable.

Under the EDF algorithm, the schedulable bound is equal to one for all task set, and the priorities become dynamic. In our experiment, the system is overloaded, due to the fact that $\overline{C}_1/T_1+\overline{C}_2/T_2+\overline{C}_3/T_3$ = 1.26 >1. However, it is interesting that the cost reaches a finite value of 0.64 in the end of the experiment, which implies that all DC motors remain stable (see Fig.4). Although the deadline miss ratio is not zero because of the overload condition, the DC motors still exhibit satisfactory performance under EDF.

Different from RM and EDF, the scheduler under MUF will decide the task execution based on the dynamic urgency characteristics of three control tasks. Since the urgencies deliver instantaneous computing resource requirements for overall control performance optimization, the CPU time will always be distributed to the task that most urgently needs it. In this experiment, it is clear that the cost function goes with the smallest values among the three scheduling cases. The final overall cost is 0.47. Fig.4 illustrates the benefits of the proposed approach.

5 Conclusions

Today's embedded control systems are often resource constrained, which inevitably results in overload conditions. The inherent uncertainty of execution time and real-time constraints further makes the scheduling of control tasks difficult. At the same time, traditional control systems are often designed without taking into account the computing resource availability. As a result, the performance of multitasking control applications is always impacted.

In this paper, we propose a direct feedback scheduling approach to maximizing overall performance of multitasking control systems that features resource constraints and workload uncertainty. The basic idea is to base the control task scheduling decisions on the instantaneous control requirements, so that the computing resources are dynamically allocated in an optimal fashion and the resulting overall control performance is maximized. A control oriented real-time task model is suggested to facilitate the co-design of feedback control and real-time scheduling. The new timing parameter, urgency, makes the model extremely suitable for the application of feedback scheduling techniques in embedded control systems. Based on this model, we present a direct feedback scheduling algorithm as well as a job-shipping scheme. The overload condition is effectively handled. The limited computing resource will be dynamically distributed to the control tasks that really need it. Using the proposed scheduling algorithm, the impact of timing uncertainty on control task scheduling is naturally avoided. It is found that the proposed approach outperforms traditional design methodologies in resource constrained control systems, as we have shown in the simulation experiments.

However, the proposed approach inevitably introduces extra overhead to the underlying operating system, although this is not accounted for in this paper. Certain computing resources are needed for frequently updating the urgency of each task. Fortunately, this overhead is relatively small due to the simplicity of urgency definition. On the other hand, this simplicity also leaves room for further improvement of our approach. For example, to take into account how the control error is going to evolve, an improved urgency may involve the derivative of control error. This remains to be one of our future works. Another issue left to be addressed is analysis of the performance of MUF in resource sufficient cases. Intuitively, when the computing resources are sufficient, one can imagine that the benefits of MUF will not be so significant as under overload conditions. This also needs further study.

Acknowledgement. This work is supported by National Natural Science Foundation of China (NSFC 60434030), National Basic Research Program of China (2002CB312200) and Advanced Research Program of France-China (PRA SI03-02).

References

1. K.-E. Årzén, A. Cervin, D. Henriksson, "Resource-Constrained Embedded Control Systems: Possibilities and Research Issues", *Workshop on Co-design for Embedded Real-time Systems (CERTS)*, Porto, Portugal, 2003.
2. Feng Xia, Zhi Wang, and Youxian Sun, "Integrated Computation, Communication and control: Towards Next Revolution in Information Technology", *Lecture Notes in Computer Science*, Vol. 3356, 2004, pp. 117-125.
3. C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", *Journal of ACM*, Vol. 20, No. 1, 1973, pp. 46-61.
4. J. A. Stankovic, C. Lu, S. H. Son, and G. Tao, "The Case for Feedback Control Real-Time Scheduling", *IEEE Proc. of the 11th Euromicro Conference on Real-Time Systems (ECTRS)*, Los Alamitos, 1999, pp. 11-20.
5. A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, "Feedback-Feedforward Scheduling of Control Tasks", *Real-Time Systems*, Vol. 23, No.1, 2002, pp. 25-53.
6. Feng Xia and Youxian Sun, "Neural Network Based Feedback Scheduling of Multitasking Control Systems", *Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, Vol. 3682, 2005, pp. 193-199.
7. Feng Xia, Liping Liu, and Youxian Sun, "Flexible Quality-of-Control Management in Embedded Systems Using Fuzzy Feedback Scheduling", *Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, Vol. 3642, 2005, pp. 624-633.
8. Feng Xia, Xingfa Shen, Liping Liu, Zhi Wang, and Youxian Sun, "Fuzzy Logic Based Feedback Scheduler for Embedded Control Systems", *Lecture Notes in Computer Science*, Vol. 3645, 2005, pp. 453-462.
9. Daniel Simon, David Robert, Olivier Sename, "Robust control/scheduling co-design: application to robot control", *11th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Francisco, California, 2005.
10. D. Henriksson, A. Cervin, J. Åkesson, K.-E. Årzén, "Feedback Scheduling of Model Predictive Controllers", *Proc. of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, San Jose, CA, 2002, pp. 207-216.
11. Feng Xia and Youxian Sun, "NN-based Iterative Learning Control under Resource Constraints: A Feedback Scheduling Approach", *Lecture Notes in Computer Science*, Vol. 3498, 2005, pp. 1-6.

12. Feng Xia and Youxian Sun, "Anytime Iterative Optimal Control Using Fuzzy Feedback Scheduler", *Lecture Notes in Artificial Intelligence (LNCS/LNAI)*, Vol. 3682, 2005, pp. 350-356.
13. A. Cervin, and J. Eker, "The Control Server: A Computational Model for Real-Time Control Tasks", *Proc. of the 15th Euromicro Conference on Real-Time Systems (ECTRS)*, Porto, Portugal, 2003, pp. 113-120.
14. G. Buttazzo, G. Lipari, and L. Abeni, "Elastic Task Model for Adaptive Rate Control", *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, Madrid, Spain, 1998, pp. 286-295.
15. M. Velasco, P. Martí and J. M. Fuertes, "The Self Triggered Task Model for Real-Time Control Systems", *WIP Session of the 24th IEEE Real-Time Systems Symposium (RTSS)*, Cancun, Mexico, 2003.
16. Feng Xia, Xiaohua Dai, Zhi Wang, and Youxian Sun, "Feedback Based Network Scheduling of Networked Control Systems", *Proc. of the 5th International Conference on Control and Automation (ICCA)*, Budapest, Hungary, Vol. 2, 2005, pp. 1231-1236.
17. J. Yépez, P. Martí and J. M. Fuertes, "The Large Error First (LEF) Scheduling Policy for Real-Time Control Systems", *WIP Session of IEEE Real-Time Systems Symposium (RTSS)*, Cancun, Mexico, 2003.
18. Gregory C. Walsh and Hong Ye, "Scheduling of networked control systems", *IEEE Control System Magazine*, Vol. 21, No. 1, 2001, pp. 57-65.
19. G. Koren, D. Shasha, "Skip-over: Algorithms and complexity for overloaded systems that allow skips", *6th IEEE Real-Time Systems Symposium (RTSS)*, Pisa, Italy, 1995.
20. K.J. Åström and B. Wittenmark, *Computer Controlled Systems*, Prentice Hall, 1997.

Feng Xia received the B.S. degree in automation from Zhejiang University, China, in 2001, as the Excellent Graduate of Zhejiang Province. Since then, he has been a Ph.D. student in the Institute of Modern Control Engineering, Zhejiang University. He was an Organizing Committee Member of the 5th World Congress on Intelligent Control and Automation (WCICA2004). In 2005 he was awarded Zhu Kezhen Scholarship at Zhejiang University. His research interests include real-time control systems, feedback scheduling, low-power computing, embedded systems, and wireless networked systems. He is currently working on flexible resource management of networked and embedded control systems.

Xiaohua Dai received the B.S. degree in Chemical and Mechanical Engineering from Fushun Petroleum Institute and the M.S. degree in Mechanical Engineering from Zhejiang University, China, in 1999 and 2002, respectively. Since then, he has been a Ph.D. student in the Institute of Modern Control Engineering, Zhejiang University. His current research interests include wireless sensor networks and embedded systems.

Youxian Sun graduated in Department of Chemical Engineering at Zhejiang University in 1964. In the same year, he became a teacher in the Department of Chemical Engineering and Institute of Industrial Process Control. From 1984 to 1987, he went to Chemical Engineering Department at University of Stuttgart, Germany as a visitor. He was promoted as Professor in 1988 and Ph.D. supervisor in 1991. He was awarded the model worker in the National Educational System and the Medal of the People Teacher. He was the delegate of the Eighth National Peoples Congress. In 1995 he was elected as the Academician of the Engineering Academics and the Zhejiang primary commissary of Chinese Democracy Federation in 1996. He was awarded the first National Excellent Science and Technology worker in 1999. In the same year, he was elected the Adjunct Director of the Standing Committee of People’s Congress of Zhejiang Province. His research fields include industrial process control, robust control theory and applications, etc.

Jianxia Shou received the B.S. and M.S. degrees in computer science from Northeast University, China, in 1987 and 1990 respectively. From 1990 to 1993, she was with the Automation Instrument Company, Hangzhou, China, as an Embedded Software Developer. From 1994 to 2000, she was with the Hangzhou University of Electronic Technology as a Lecturer. From 2001 to 2002, she was with the Sprint Corporation, Kansas, USA, as a Software Developer (Contractor). Since 2003, she was with the College of Software, Zhejiang University of Technology as an Associate Professor. Her current research interests include learning control, artificial intelligence, etc.