Firm Real-Time System Scheduling Based on a Novel QoS Constraint

Donglin Liu, Xiaobo Sharon Hu, *Senior Member*, *IEEE*, Michael D. Lemmon, and Qiang Ling

Abstract—Many real-time systems have *firm* real-time requirements which allow occasional deadline violations but discard any jobs that are not finished by their deadlines. To measure the performance of such a system, a quality of service (QoS) metric is needed. Examples of often used QoS metrics for *firm* real-time systems are average deadline miss rates and (m,k)-$firm$ constraints. However, for certain applications, these metrics may not be adequate measures of system performance. This paper introduces a novel QoS constraint for *firm* real-time systems. The new QoS constraint generalizes existing *firm* real-time constraints. Furthermore, using networked control system as an example, we show that this constraint can be directly related to the control system's performance. We then present three different scheduling approaches with respect to this QoS constraint. Experimental results are provided to show the effectiveness of these approaches.

Index Terms—Firm real-time system, scheduling, quality of service, networked control system.

1 Introduction

Real-time systems are abundant in our everyday life. They can be found in many different applications, such as networked control systems and multimedia conferencing systems. Real-time systems are usually classified as being *hard*, *soft*, or *firm*, depending on how the system performs when one or more deadlines are missed. For *hard* real-time systems, no deadline misses are tolerated. For *soft* real-time systems, it is acceptable for tasks to miss deadlines occasionally and tasks not finished by their deadlines are still completed, albeit with reduced values. *Firm* real-time systems (FRTS) also allow occasional deadline misses. But, unlike *soft* real-time systems, if an instance (job) of a task is not completed by its deadline, the instance is considered valueless and is discarded (or dropped) [4] by the system. Consider the cruise control in an automobile. Suppose the software fails to provide current velocity in time for the control algorithm to use. The control algorithm would then use the old value to compute necessary control signals rather than wait for the delayed velocity. The cruise control system can still function acceptably because the amount of velocity change between the last sample and the current sample is small. The delayed measurement is useless to the system and could be considered as being dropped. This paper focuses on FRTS.

Quantifying the "occasional deadline misses" is critical for evaluating the Quality of Service (QoS) of an FRTS. Previously used QoS metrics for FRTS can be grouped into two categories: 1) QoS metrics based on the *average* behavior of the system dropout (deadline miss) process, such as deadline miss ratio (or average dropout rate), effective processor utilization, and completion count [2], [3]. 2) QoS metrics based on the system dropout pattern within certain "windows." (A window refers to a fixed number of consecutive invocations (or jobs) of a task.) According to the relationship between adjacent windows, window-based QoS constraints can be further categorized as fixed-window constraints [23], [22], where adjacent windows do not overlap each other, and sliding-window constraints [8], [24], where the window "slides" along the time line. Both of the above two categories of QoS constraints have some shortcomings and limitations in describing the desired FRTS behaviors under certain circumstances, as we will show later.

This paper introduces a novel QoS constraint that is more general and flexible than existing QoS constraints for FRTS. The constraint is specified based on the Markov Chain (MC) process. We will show in Section 3.2 that MC-based QoS constraints can be used to describe all the QoS metrics or constraints discussed above. The power of the MC-based constraint lies in that it can incorporate the probabilistic behavior of the system. Moreover, for a wide range of control systems, this constraint can be directly related to the overall control system performance. In fact, it is the desired control system performance that determines the actual parameters of the MC-based constraint. We will illustrate this through an example networked feedback control system. Since our MC-based constraint is different from all constraints used in previous work, we present several alternative scheduling approaches and evaluate their effectiveness.

The rest of the paper is organized as follows: The next section presents related work and necessary notational conventions. In Section 3, we introduce the concept of the MC-based constraint, elaborate its features, and demonstrate the usefulness of MC-based constraints by using a Networked Control System as an example. Then, in Section 4, we present three new scheduling algorithms specifically designed for MC-based constraints and discuss how to compare

- *D. Liu and X.S. Hu are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {dliu, shu}@nd.edu.*
- *M.D. Lemmon is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: lemmon@nd.edu.*
- *Q. Ling is with Seagate Technology, Pittsburgh, PA 15237. E-mail: Qiang.Ling@seagate.com.*

Manuscript received 14 May 2004; revised 16 July 2005; accepted 30 Aug. 2005; published online 20 Jan. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0159-0504.

them against traditional scheduling algorithms. We then show our experimental results in Section 5. Conclusions and future work are in the final section.

2 Related Work and Notational Conventions

In this section, we review related work and define necessary notation.

2.1 QoS Metrics for FRTS

One often used QoS metric for FRTS is the deadline miss ratio or average dropout rate (as an instance of a task is dropped if it cannot be finished by its deadline), defined as the percentage of the number of deadline misses with respect to the number of jobs admitted to the system. Two other similar metrics are the effective processor utilization and completion count discussed in [2], [3]. The advantage of these metrics is that they can be estimated offline and used directly to guide the scheduler design. Several techniques have been proposed to evaluate the average dropout rate in the presence of task execution variations for different scheduling schemes [7], [21], [10], [11].

A drawback with these metrics is that they cannot directly express how deadline misses or dropouts are distributed. In some systems, such as those found in control and multimedia applications, system performance is sensitive to the dropout patterns or the distribution of dropouts. For example, if dropouts occur consecutively for several jobs of the same task, then the system performance may be totally unacceptable.

To overcome the shortcoming of the QoS metrics based on average dropout rate, a variety of window-based QoS constraints have been proposed for FRTS. Hamdaoui and Ramanathan introduced the (m,k)-$firm$ constraint in [8]. The (m,k)-$firm$ constraint specifies that tasks should meet at least m deadlines in any k consecutive invocations. Koren and Shasha proposed the *skip factor* [12] constraint, where a task with a skip factor of s is allowed to have one job skipped out of s consecutive jobs. In [4], Bernat et al. presented the following set of *firm* real-time constraints based on the desired deadline miss patterns.

1. $\binom{n}{m}$: In any window of m consecutive jobs of a task, at least n jobs must meet their deadlines.
2. $\langle {n \atop m} \rangle$: In any window of m consecutive jobs of a task, at least n consecutive jobs must meet their deadlines.
3. $\binom{\overline{n}}{m}$: In any window of m consecutive jobs of a task, no more than n deadlines are missed.
4. $\langle {\overline{n} \atop m} \rangle$: In any window of m consecutive jobs of a task, no more than n consecutive jobs miss their deadlines.

The authors of [4] claim that, with the above four constraints and combinations of them, it is possible to represent all real scenarios of the desired dropout patterns. However, as we will show in Section 3.2, there exist some situations where it is difficult (if not impossible) to represent the preferred dropout pattern by the above four constraints or any combinations of them, although the preferred dropout pattern is simple and straightforward.

The window-based QoS constraints provide a more comprehensive measure of deadline misses than the average dropout rate. The associated schedulability analysis is carried out assuming complete knowledge of the tasks is available. Unfortunately, many real-time systems must face the uncertainties inherent in executing software tasks as well as those arising from the environment. Simply using the worst-case parameters of the task, e.g., worst-case execution time, for schedulability analysis would result in overly expensive system implementations. Furthermore, it may be inadequate to use just two parameters in the QoS constraint (e.g., n and m in [4]) to specify the desired dropout patterns in a system where uncertainties in task parameters are to be considered.

2.2 Online Scheduling Algorithms for FRTS

To meet an MC-based constraint, previous scheduling methods (e.g., [4], [8], [12], [17], [18], [24]) are no longer applicable due to the following reasons:

1. These methods assume a deterministic dropout pattern, while MC-based constraints are inherently probabilistic.
2. Some dropout patterns described by MC-based constraint cannot be described by any existing constraints for FRTS.

We present several scheduling alternatives for this new QoS constraint. A common feature of these scheduling approaches is to use feedback information to adjust the schedule. They differ in terms of the feedback information being used and the actual schemes used in adjusting the schedule.

Using feedback information for scheduler design is not new. The original work can be traced back to [5] for scheduling in general-purpose operating systems. In recent years, using feedback information for scheduling has also seen an increase in the real-time system area. For example, in [9], the authors proposed the *Adaptive Earliest Deadline* (AED) priority assignment policy to stabilize the performance of the *Earliest Deadline First* (EDF) scheduling algorithm under overload situations. AED monitors the *HitRatio* (the fraction of the task instances that should meet their deadlines that have actually met their deadlines) to adjust task priorities. Brandt et al. presented a dynamic QoS manager (DQM) to change a task's QoS levels according to the sampled central processing unit (CPU) utilization or deadline misses [6]. Another group of work in the real-time system area can be categorized as applying feedback control theory to scheduling (e.g., [1], [16], [19], [20]). Based on feedback control theory, a controller is designed which takes a certain controlled variable as input and adjusts the manipulated variable. The controlled variable can be the deadline miss ratio or CPU utilization, while the manipulated variable can be the utilization allotted to each task or the task execution time estimation factor.

All of the above approaches (except [19]) target *soft* real-time systems where each task has multiple QoS levels. In this paper, we consider real-time tasks that do not have multiple QoS levels, but have *firm* real-time constraints. The tasks are periodic, but their execution times can vary between the best-case execution time (BCET) and worst-case execution time (WCET).

2.3 Notational Conventions

We consider a system consisting of periodic tasks. The ith task is denoted as τ_i. The period of τ_i is denoted by t_i and the deadline of τ_i is denoted by d_i. The execution time of τ_i is modeled by a random variable, C_i, where the probability of C_i taking the value of e_k is denoted by $P_{C_i}(e_k) = Pr(C_i = e_k)$, for $k = 1, 2, \ldots, K$. Each instance of a task is called a *job*. The jth job of τ_i is denoted as τ_{ij}. The execution time of τ_{ij}, denoted by c_{ij}, has the same probability distribution as C_i. That is, $P_{c_{ij}} = P_{C_i}$, for $j = 1, 2, \ldots$.

To simplify the notation, we assume here that $P_{C_i}(e_k)$ is a constant and is independent of other task requests. However, our work does not depend on this assumption. The distribution of C_i could be obtained from experiments or through profiling.

In this paper, we study systems in which a job is discarded (or dropped) if it is not completed by its deadline. This practice is adopted in many feedback control systems [13]. We use f_{ij} to denote the completion status of job τ_{ij}, i.e., $f_{ij} = 1$ if τ_{ij} is completed at or before its deadline and $f_{ij} = 0$ otherwise. Note that $f_{ij} = 0$ is due to either the late finish of τ_{ij} or the rejection of τ_{ij} before it is started. For simplicity, we refer to both cases as the jobs being dropped. The average dropout rate of task τ_i is denoted by θ_i.

We use $H_{ij}^{n_i}$ to denote the completion status of the n_i previous jobs of τ_{ij}. $H_{ij}^{n_i}$ is an n_i-bit binary string and $H_{ij}^{n_i} = \{f_{i(j-n_i)}, f_{i(j-n_i+1)}, \cdots, f_{i(j-2)}, f_{i(j-1)}\}$. We also refer to $H_{ij}^{n_i}$ as the n_i-bit execution pattern of τ_{ij}. For example, if the first and the third job of τ_1 miss their deadlines and the second job of τ_1 meets its deadline, then we have: $f_{11} = 0$, $f_{12} = 1$, $f_{13} = 0$, and $H_{14}^3 = \{010\}$. To make the definition complete, we let $f_{ij} = 0$ when $j \leq 0$. The 3-bit execution pattern of τ_{13}, H_{13}^3, is $\{001\}$.

3 The Markov Chain-Based QoS Constraint

Many real-time systems exhibit probabilistic behavior due to uncertainty inherent in the operating environment and the task parameters. In such systems, the pattern of jobs being completed or dropped may also be probabilistic. Since dropout patterns can have a significant impact on the overall system performance in certain applications and neither the dropout rate nor window-based constraints have the descriptive power to capture probabilistic behaviors, we propose in this section a Markov Chain (MC) based model to describe the desired stochastic job dropout behavior. We call this Markov Chain the MC-based constraint. The MC-based constraint associated with τ_i is denoted by $\mathcal{MC}_i$.

3.1 MC-Based Constraint: Definition and Examples

Given a real-time task τ, $\mathcal{MC}$ is a discrete stochastic process with M states, denoted as $X_{(m)}$ for $m = 1, 2, \cdots, M$, and $M \geq 2$. The transition probability from one state $X_{(j)}$ to another $X_{(k)}$ denotes either the probability of the next job being dropped or the probability of the next job being completed.

To show the dropout patterns associated with each state explicitly, each state is associated with a specific string of the job's completion status bits, f_{ij}, to represent the recent execution pattern. For example, if we use two bits to store the job completion status bits, i.e., we only use the execution status of the two most recent jobs, we can have at most four states ($M = 4$). Fig. 1a depicts the general MC process for

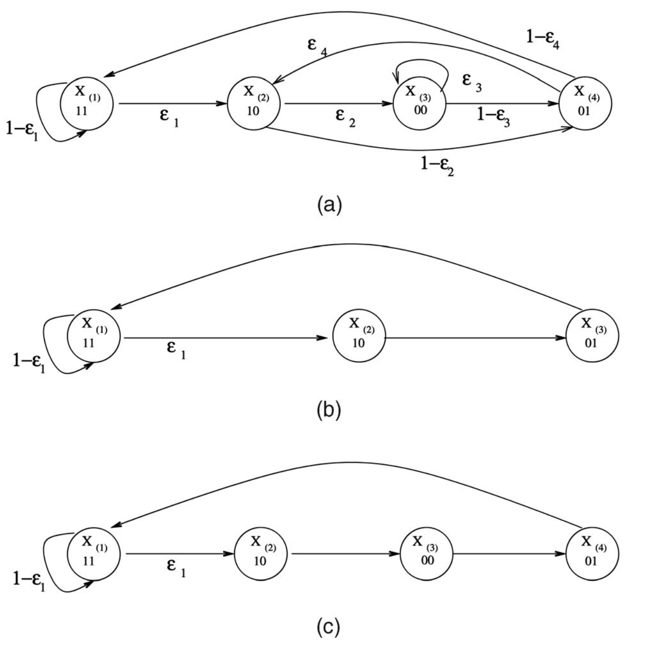


Fig. 1. Three Markov chains for the case where 2 bits are used for the execution pattern. (a) The general MC dropout process. (b) An example MC-based constraint, which is equivalent to $(2,3)$-$firm$ constraint. (c) An example MC-based constraint, which describes the optimal dropout process in the networked control system in Section 3.3.

the case of using two completion status bits for each state, where ε_1 to ε_4 are the probabilities of the next job being dropped at the respective states. The state which represents the execution pattern of one completed job followed by one dropped job can be written as $X_{(2)} = \{10\}$. (We use the rightmost bit to represent the completion status of the most recent job.) Note that each state can make a transition to at most two other states with nonzero transition probabilities. For instance, state $X_{(2)} = \{10\}$ can only make a transition to either state $X_{(3)} = \{00\}$ (if the next job is dropped) or state $X_{(4)} = \{01\}$ (if the next job is completed). This is because the execution patterns of consecutive jobs are related to each other. It is easy to see that the last $n-1$ bits of H_{ij}^n are the same as the first $n-1$ bits of $H_{i(j+1)}^n$.

Given the fact that a state can only make a transition to at most two other states based on the completion status of the next job, the dropout probability (i.e., dropout rate) at each state, denoted by ε_m, for $m = 1, 2, \cdots, M$, is sufficient to uniquely specify an MC-based constraint. Fig. 1b shows a specific example of an MC-based constraint which has three states and ε_1 represents the dropout probability at state $X_{(1)} = \{11\}$. It is not difficult to verify that this MC-based constraint is equivalent to the (m,k)-$firm$ constraint [8], where $m = 2$ and $k = 3$. Fig. 1c illustrates another MC-based constraint which cannot be described by any window-based constraint.

The completion status of a task, i.e., jobs being dropped or not, can be considered as a discrete stochastic process when the task execution times are probabilistic. An MC-based constraint thus specifies the desired stochastic dropout process. The length of the bit string associated with a state (i.e., the number of consecutive jobs observed)

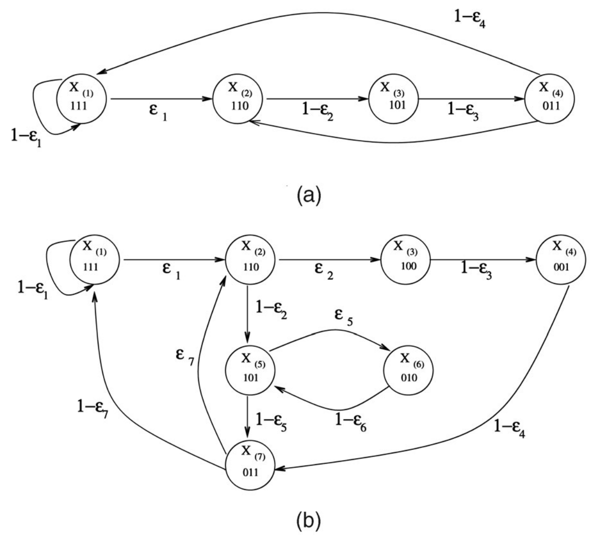


Fig. 2. (a) The MC-based constraint which is equivalent to the $\langle {2 \atop 4} \rangle$ window-based constraint [4]. (b) The MC-based constraint which is equivalent to the $\binom{2}{4}$ window-based constraint [4].

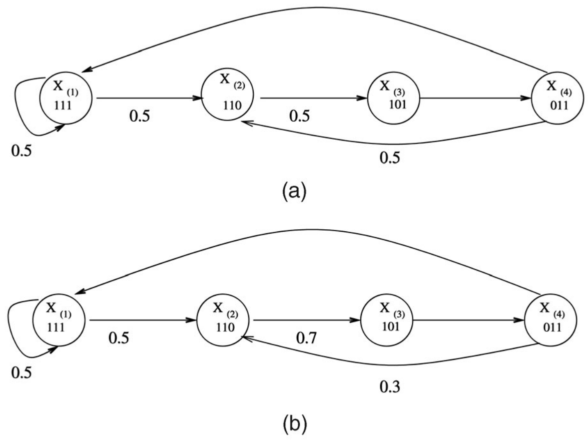


Fig. 3. Two MC-based constraints which satisfy the $\langle {2 \atop 4} \rangle$ window-based constraint.

determines the general structure (the maximum number of states and all possible transitions) of an MC-based constraint. For example, if we use 4 bits in each state of an MC-based constraint, the maximum number of states is $2^4 = 16$. In practice, the number of bits used is not large. Also, the number of states in an MC-based constraint is usually less than the maximum since the purpose of the constraint is to eliminate certain dropout patterns so that the system performance can be improved. In Fig. 2, we show two MC-based constraints corresponding to the $\langle {2 \atop 4} \rangle$ (in any window of four consecutive jobs of a task, at least two *consecutive* jobs must meet their deadlines) and $\binom{2}{4}$ (in any window of four consecutive jobs of a task, at least two jobs must meet their deadlines) window-based constraints defined in [4]. In these cases, each state in the MC-based constraint is associated with a 3 bit execution pattern.

The MC-based constraint has three distinctive features. First, it can readily capture dropout rate-based constraints. Second, it can be used to represent any window-based constraint. Last and more importantly, it can be used to *quantitatively* capture the desired performance of certain real-time system. We describe the first two features in the next subsection and the last one in Section 3.3.

3.2 Features of the MC-Based Constraint

Clearly, the MC-based constraint generalizes the dropout rate-based constraint. To see this, observe that an MC-based constraint can relate the dropout pattern with the average dropout rate-based constraint. There exists a one-to-many mapping from a dropout rate based constraint to the MC-based constraint. Each different mapping represents a different dropout pattern. For example, for a desired dropout rate of $\frac{1}{3}$, by setting $\varepsilon_1 = 1.0$ in Fig. 1b and setting $\varepsilon_1 = \frac{1}{3}$ in Fig. 1c, both MC-based constraints reflect the desired dropout rate. Similarly, given an MC-based constraint, one can readily compute the average dropout rate using the following procedure: First, compute the stationary state distributions, denoted as π_m for $m = 1, 2, \cdots, M$, based on the state transition matrix of the given MC-based constraint. Then, the average dropout rate θ can be calculated as:

$$\theta = \sum_{m=1}^{M} \pi_m \times \varepsilon_m. \tag{1}$$

MC-based constraints also generalize window-based constraints. Any window-based constraint can be represented as an MC-based constraint. A straightforward way of deriving an MC-based constraint that is equivalent to a window-based constraint is to enumerate all possible patterns allowed by the window-based constraint and summarize the patterns by a Markov Chain. We have shown that a $(2,3)$-$firm$ constraint can be represented by the MC-based constraint in Fig. 1b and a $\langle {2 \atop 4} \rangle$ constraint in [4] can be represented by the MC-based constraint in Fig. 2a. We should point out that the mapping from a window-based constraint to an MC-based constraint is not unique. The MC-based constraint is characterized by the state transition matrix (dropout possibilities at each state). Two MC-based constraints with different state transition matrices may both satisfy the same window-based constraint. For example, both MC-based constraints in Fig. 3 satisfy the window-based constraint where *at least* two *consecutive* jobs must meet their deadlines in any window of four *consecutive* jobs. However, their state transition matrices are not the same.

There exist dropout patterns which can only be represented by the MC-based constraint. An example of such a dropout pattern represented by an MC-based constraint is given in Fig. 1c. Observing the MC-based constraint given in Fig. 1c carefully, one can find that the desired dropout pattern really specifies the following:

1. In any window of four jobs, at least two of them must meet their deadlines.
2. If any job in a window misses its deadline, the next job has to be dropped by the system.

Note that it is impossible to describe the above desired dropout pattern by the constraints in [4] or by their combinations.

The MC-based constraint is more attractive because it provides much more descriptive power. One might argue

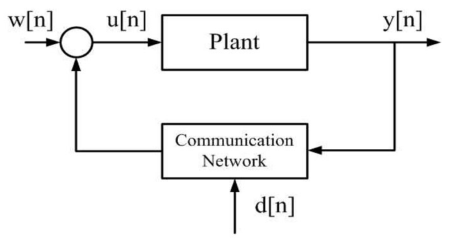


Fig. 4. NCS model.

that an MC-based constraint can be replaced by simply combining the dropout rate constraint and the window-based constraint. But, from the above examples of the MC-based constraint, one can see that a window-based constraint does not care about the actual values of the dropout probabilities, i.e., ε_m. On the other hand, an average dropout rate-based constraint cannot uniquely describe the dropout patterns and the dropout probabilities.

3.3 A Motivational Example

To demonstrate the value of the MC-based constraint, we briefly review a recent work [14] in networked control systems and show how an MC-based constraint is used to quantify the QoS of a networked control system. This characterization is important as it allows us to directly evaluate different MC-based constraints in terms of their impact on application performance.

A *networked control system* (NCS) is a control system whose feedback path is realized over a communication network. Feedback measurements may be occasionally dropped for one of two reasons: either the medium is unreliable or the network is overloaded. If we think of the communication network as a resource and packets to be transmitted as requests of the resource, packet dropouts are similar to job dropouts during processor execution.

Fig. 4 shows a generic NCS. This system consists of a plant and a communication network. The plant's input, $w[n]$, is a zero-mean white noise disturbance. The output, $y[n]$, is fed back through the communication network in a manner that is modeled by a *dropout process*, $\{d[n]\}$. The dropout process is a binary random process that can be described by an underlying Markov chain with transition matrix Q. If $d[n] = 1$, then there is no dropout and the plant's output $y[n]$ is successfully transmitted over the network. Therefore, the feedback signal equals $y[n]$ when $d[n] = 1$. If $d[n] = 0$, then $y[n]$ is dropped, (e.g., due to network congestion) and the controller simply reuses the past feedback signal $y[n-1]$. This particular control system tries to reject the input disturbance, w, at the output, y. The performance of this control system is therefore measured by the *output power* (computed as $\|y\|^2_{\mathcal{P}} = \mathbf{E}(\lim_{n\to\infty} \frac{1}{N}\sum_{n=0}^{N} y^2[n])$), where systems having smaller $\|y\|_{\mathcal{P}}$ perform better than systems with larger $\|y\|_{\mathcal{P}}$. The main result in [14] provides a systematic method for computing $\|y\|_{\mathcal{P}}$ as a function of the Markov chain's transition matrix Q.

Consider a specific plant described by the following difference equation:

$$y[n] + y[n-1] + 2y[n-2] = u[n-1] + 2u[n-1]. \quad (2)$$

This plant has the transfer function: $\frac{z+2}{z^2+z+2}$, which is open-loop unstable with a nonminimum phase zero. If the dropout process, $\{d[n]\}$, is independent and identically distributed (i.i.d.) (i.e., governed by the average dropout rate constraint), the system performance measured by the output power, $\|y\|^2_{\mathcal{P}}$, as a function of the average dropout rate is shown by the circled curve in Fig. 5 [14]. If the dropout process, $\{d[n]\}$, follows the $(2,3)$-firm guarantee dropout policy (i.e., the $(2,3)$-firm constraint, which is a sliding-window-based constraint), the system performance is depicted by the crossed curve in Fig. 5 [14]. Clearly, different dropout processes may lead to different performance, even when the dropout rate is the same.

A natural question to ask is what dropout process can maximize overall system performance (i.e., minimize output power) subject to a lower bound on the average dropout rate. This is precisely the problem studied in [14]. The solution to this problem, which is a Markov Chain, provides the statistical pattern of dropouts that degrade overall control system performance as little as possible for a given

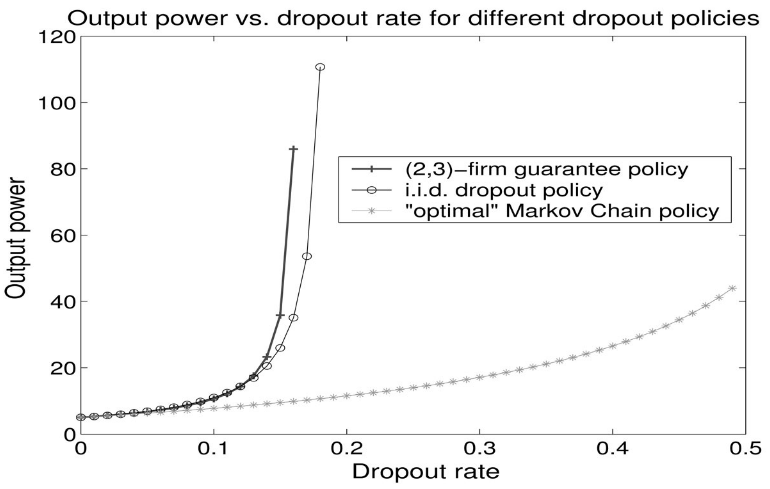


Fig. 5. Comparison of the three dropout processes.

TABLE 1
The Timing Parameters of an Example Task Set

$Task$	t_i	d_i	$(e_{i1}, P_{C_i}(e_{i1})$	$(e_{i2}, P_{C_i}(e_{i2})$
τ_1	5	5	(2, 0.75)	(5, 0.25)
τ_2	10	10	(4, 1.0)	

dropout rate. Thus, using this "optimal" dropout process as the QoS constraint should lead to superior system performance. The results in [14] reveal that this "optimal" dropout process for the plant described by (2) is simply the Markov Chain in Fig. 1c.

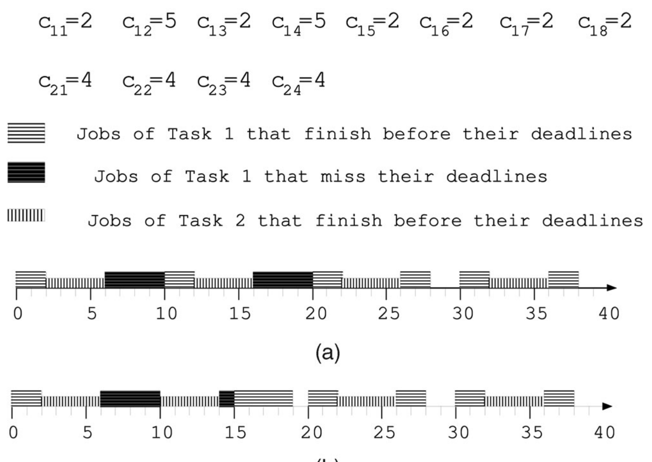
The "$*$" curve in Fig. 5 depicts the performance for this "optimal" dropout process. It is evident that the "optimal" dropout process performs much better than either the $(2,3)$-firm dropout process or i.i.d. dropout process. Note that systems driven by the $(2,3)$-firm dropout process and i.i.d. dropout process go unstable (output power larger than 100) at dropout rates in excess of 20 percent, while the optimal dropout process greatly extends the region of stability for the closed loop system.

Through the above example, one can see that the MC-based constraint provides more descriptive power and can capture desirable dropout processes that cannot be modeled by existing *firm* real-time constraints. Though the results shown in Fig. 5 are not representative of all networked control systems, they do underscore an important point, namely, that it can be dangerous to use ad hoc heuristics to specify QoS constraints in feedback control systems. The closed loop nature of these systems requires a much more flexible approach in characterizing QoS constraints. We believe that the MC-based constraint provides such flexibility. Note that the example control system is representative of a wide range of control applications, such as power-train control systems in automobiles and satellite control systems.

4 Design and Evaluation of Scheduling Approaches

Given a set of real-time tasks and MC-based constraints associated with some of the tasks, one could still use a scheduling algorithm designed without considering MC-based constraints, but such algorithms may not perform well. Consider a simple example where a task set contains two periodic tasks (τ_1 and τ_2) with the task parameters as given in Table 1. Let τ_1 be a task that sends a feedback signal to a plant and is associated with the MC-based constraint given in Fig. 1c. The period and deadline of τ_1 is five time units. Possible execution times of τ_1 are either two time units with a probability of 0.75 or five time units with a probability of 0.25. τ_2's period and deadline is 10 time units. All jobs of τ_2 take four time units.

Consider the task execution pattern in the first 40 time units. Assume that τ_{12} and τ_{14} each take five time units to finish while the rest of jobs of τ_1 task two time units. Suppose we use the nonpreemptive Earliest Deadline First (EDF) scheduling algorithm to schedule the task set and a job is dropped if it is not finished by its deadline. Fig. 6a depicts the resulting execution pattern, where the horizontally shaded boxes represent τ_1's jobs that finish on time,



c11=2 c12=5 c13=2 c14=5 c15=2 c16=2 c17=2 c18=2
c21=4 c22=4 c23=4 c24=4
Jobs of Task 1 that finish before their deadlines
Jobs of Task 1 that miss their deadlines
Jobs of Task 2 that finish before their deadlines
0 5 10 15 20 25 30 35 40
(a)
0 5 10 15 20 25 30 35 40
(b)

Fig. 6. Two different execution patterns for the tasks in Table 1. Jobs that miss their deadlines are colored black.

vertically shaded boxes represent τ_2's jobs that finish on time, and the black boxes represent τ_1's jobs that miss their deadlines. Note that τ_{12} and τ_{14} did not finish by their deadlines and, hence, were dropped. The execution pattern of τ_1 can be described as 10101111, where 1 means a job meets the deadline and 0 otherwise. If we assume that the execution pattern is repeated infinitely, then the average dropout rate of τ_1, θ_1, is equal to 0.25.

We could use some other algorithm to obtain a different execution pattern shown in Fig. 6b. In this execution sequence, τ_{12} and τ_{13} missed their deadlines. The execution pattern now is 10011111, but θ_1 is still 0.25. Task τ_2 in both cases always finishes by its deadline. The only difference between the two schedules is the dropout patterns of τ_1. In terms of meeting the MC-based constraint (see Fig. 1c), the execution pattern in Fig. 6b is clearly more desirable. If we compare the output power under the dropout processes indicated by the two execution patterns, we can also get that the power corresponding to the second pattern is much lower than that of the first one.

The above example reveals that there exist different dropout patterns that satisfy the same dropout rate and using existing scheduling algorithms may not help to satisfy the MC-based constraints because they are not trying to enforce an MC dropout process. So, it is worthwhile to investigate scheduling approaches for systems having MC-based constraints.

Recall from Section 3.1 that an MC-based constraint is specified by the dropout rate at each state. Each combination of dropout rate values results in one average dropout rate, θ. The average dropout rate θ of a task should not take any arbitrary value between 0 and 1 as it impacts both the system performance and the load that the task presents to the system. We use $\overline{\theta}$ and $\underline{\theta}$ to represent the upper and lower bound on θ. For example, the upper bound on θ of the MC-based constraint in Fig. 1c should be smaller than 0.5 because, when the dropout rate of a task is higher than 0.5, it is impossible to satisfy the MC-based constraint in Fig. 1c. The dropout probabilities as well as the θ bounds will be used by the scheduling approaches.

A good scheduler in term of meeting the MC-based constraint should have the resulting dropout process be as close to the MC constraint as possible. However, this cannot always be done without interfering with other tasks that share the same resource and may have other types of QoS constraints. Measuring the "goodness" of a scheduler in such a case is not a trivial matter. We will discuss how to evaluate the performance of a scheduler later in this section.

4.1 Three Online Scheduling Algorithms

Without loss of generality, we consider that there are two types of tasks in the *firm* real-time system: tasks with MC-based constraints (referred to as control tasks, such as those in an NCS) and tasks with average dropout rate constraints (referred to as noncontrol tasks). We are aiming at achieving better control performance without sacrificing the performance of the tasks with average dropout rate constraints by satisfying the MC-based constraints.

Because some tasks have probabilistic execution times, the actual dropout patterns of tasks are also probabilistic and change dynamically. In order to help a control task follow the MC-based constraint, an online scheduling algorithm should be a good choice because it is able to reassign the priority based on the recently observed dropout pattern. For example, if the recent execution pattern H_{ij}^2 is $\{01\}$ and the MC-based constraint is as given in Fig. 1c, then τ_{ij} should be assigned to a higher priority so that it has a higher chance to meet its deadline. In this way, the dropout process would follow the given MC-based constraint.

All three scheduling algorithms we propose in this paper are online algorithms. The common idea among them is to partition jobs dynamically into different groups. Some groups are given higher priorities than others, while jobs within each group are scheduled by some conventional scheduling algorithms. This is similar to the mandatory versus optional job partitioning used in [17], [18]. Instead of a two-way partition, we use three groups, *Must Finish* (MF), *Better Finish* (BF), and *Optional Finish* (OF). As the names indicate, the three groups have decreasing priorities. The jobs in the first two groups are scheduled by a priority-based scheduling algorithm such as the EDF scheduling algorithm or Rate Monotonic (RM) algorithm, while the tasks in the OF group can be scheduled by a randomized priority assignment similar to [9]. (Note that if a non-preemptive scheduling scheme is adopted, the jobs in the OF group can simply be discarded so that they will not block the execution of tasks in other groups. More sophisticated algorithms could be used in this case to predict if a job in the OF group can be completed without interfering with the jobs in the other groups. But, this is beyond the focus of this paper.)

In our scheduling algorithms, when the average dropout rate of a noncontrol task is higher than the given constraint, the jobs of this task are always put in the MF group. Otherwise, the jobs of this task are put in the BF group. Jobs of the control tasks are partitioned between the MF and OF groups. Our goal is to make the tasks' dropout processes follow the MC-based constraints as close as possible and to bound the average dropout rates by $\overline{\theta}$ and $\underline{\theta}$ through judiciously partitioning the jobs into different groups.

The job partitioning is performed during runtime so as to better respond to system variations. The three algorithms differ in their ways of partitioning the jobs of control tasks and are discussed in detail in the rest of the section.

4.1.1 MC Driven Algorithm (MDA)

The MDA directly uses MC-based constraints to partition jobs. That is, the dropout probabilities of the given MC-based constraint are used to decide whether a newly arrived job is put in the MF or OF group. Since the actual value of the dropout probability at a specific state depends on the value of the average dropout rate, we need to determine which average dropout rate should be used. Note that this average dropout rate reflects the desired load that the corresponding task should present to the system and should be bounded by $\overline{\theta}$ and $\underline{\theta}$. A reasonable choice is the estimated deadline miss ratio of the task when scheduling the task set by an algorithm optimal in terms of schedulability (e.g., RM or EDF).

Specifically, for a given task set and a chosen scheduling algorithm such as EDF or RM, we first apply offline an estimation algorithm such as the ones introduced in [7], [10], [11], [21] to obtain the average dropout rate for each task. (An alternative to using an estimation method is simulation.) Based on the average dropout rate of τ_i, we obtain the dropout probability ε_m at each state of $\mathcal{MC}_i$. Let the number of bits used to represent the states in $\mathcal{MC}_i$ be n_i. During runtime, the scheduler records the execution pattern of the most recent n_i jobs. Assuming the newly arrived job is τ_{ij}, the current execution pattern of τ_{ij} is specified by $H_{ij}^{n_i}$. If $H_{ij}^{n_i}$ corresponds to the mth state $X_{(m)}$ in $\mathcal{MC}_i$, τ_{ij} is put into the OF group with probability ε_m and put into the MF group with probability $1-\varepsilon_m$. This probabilistic grouping method can be implemented online using a random number generator.

The time complexity of this scheduling algorithm depends on the number of states in the MC-based constraint and the random number generation. Usually, the number of states is small. The efficiency of a random number generator depends on the desired quality; minimal standard generators require 1 to less than 10 multiplications and 5 to 10 additions [15].

4.1.2 Dropout-Rate Driven Algorithm (DDA)

The main difference between the MDA and DDA is that the latter retains, for each task, a "long" execution pattern instead of just the most recent execution pattern used to determine the previous state of the system dropout process. This long execution pattern contains the execution pattern of a task and is used to compute the *estimated* average dropout rate of τ_i, $\hat{\theta}_{ij}$, at the beginning of τ_i's jth period. Therefore, the DDA gets the average dropout rate from the feedback information instead of offline estimation, as is the case with the MDA. The length of this execution pattern, L, is a parameter set by the user.

During runtime, the DDA uses the long execution pattern for τ_i, represented by an L bit binary number, to compute $\hat{\theta}_{ij}$, i.e.,

$$\hat{\theta}_{ij} = 1 - \frac{\sum_{k=1}^{L} f_{j-k}}{L}, \tag{3}$$

where τ_{ij} is the newly arrived job. $H_{ij}^{n_i}$ is used to determine the current state of the dropout process. The algorithm then applies the following rules to decide in which group the newly arrived job is placed:

1. If $\hat{\theta}_{ij} > \overline{\theta}_i$, put τ_{ij} in the MF group.

2. If $\hat{\theta}_{ij} < \underline{\theta}_i$, put τ_{ij} in the OF group.
3. If $\underline{\theta}_i \leq \hat{\theta}_{ij} \leq \overline{\theta}_i$, find the dropout probability, i.e, ε_k corresponding to the current state. τ_{ij} is put in the OF group with this dropout probability. Otherwise, it is put in the MF group.

The rationale behind the DDA is to make the average dropout rate stay within the given bounds and the dropout process follow the MC-based constraint at $\hat{\theta}_{ij}$. DDA is less efficient than MDA in terms of memory usage since it must maintain an execution pattern usually much longer than MDA. However, DDA only needs a few more (less than 10) operations than MDA. Note that the execution pattern being monitored to compute $\hat{\theta}_{ij}$ should not be too long so that the recent dropout process is not overshadowed by the dropout process a long time ago. We have set L to 60, 80, 100, 120, 140, and 200 and found that the results of our scheduling algorithm did not vary much. In our experiments, we used $L = 100$.

4.1.3 *Feedback Driven Algorithm (FDA)*

The FDA uses a somewhat different way to partition jobs. It avoids the use of any random number generator while still trying to achieve the same goal as the other algorithms. Similarly to DDA, for each control task τ_i, FDA retains a "long" execution pattern to assist the computation of $\hat{\theta}_{ij}$. Furthermore, this execution pattern is used to compute the *estimated* dropout probability at each state of the MC-based constraint for τ_i.

During runtime, upon arrival of τ_{ij}, the FDA first computes $\hat{\theta}_{ij}$ according to (3) and determines the current state of the dropout process similar to DDA. Then, it computes the *estimated* dropout probability, denoted as $\hat{\varepsilon}_k$, at the current state $X_{(k)}$.

$$\hat{\varepsilon}_k = \frac{\text{Total number of transition from } X_{(k)} \text{ to } X_{(p)} \text{ in } H_{ij}^L}{\text{Total number of state } X_{(k)} \text{ in } H_{ij}^L}. \tag{4}$$

$X_{(p)}$ is the state which can be reached from $X_{(k)}$ if the next job is dropped. FDA also finds the desired dropout probability ε_k for the average dropout rate $\hat{\theta}_{ij}$ from the MC-based constraint specification as MDA does. The algorithm finally applies the following rules to decide in which group the newly arrived job is placed.

1. If $\hat{\theta}_{ij} > \overline{\theta}_i$, put τ_{ij} in the MF group.
2. If $\hat{\theta}_{ij} < \underline{\theta}_i$, put τ_{ij} in the OF group.
3. If $\underline{\theta}_i \leq \hat{\theta}_{ij} \leq \overline{\theta}_i$ and $\hat{\varepsilon}_k > \varepsilon_k$, τ_{ij} is put in the MF group. Otherwise, it is put in the OF group.

The FDA monitors the dropout probabilities as well as the average dropout rates and uses both to partition jobs. Its philosophy is similar to DDA, but it avoids the use of a random number generator by employing the observed dropout probabilities.

4.2 Evaluation of Scheduling Algorithms

We have proposed three scheduling algorithms to help meet MC-based constraints. One challenge we are still facing is how to measure the "goodness" of a scheduler in order to compare the performance of different schedulers. As we pointed out earlier, a good scheduler should produce a dropout process as close to the MC-based constraint as possible. Comparing the "closeness" of one stochastic process to another is not a trivial issue. It may seem that this problem bears similarities to the evaluation problem in the Hidden Markov Model (HMM) [27]. However, given that an MC-based constraint often contains transition probabilities equal to zero, applying the method for solving the evaluation problem of HMM often leads to meaningless results. Since an MC-based constraint must come from the specification of an application and its selection depends on some performance measure of the application, it is natural to evaluate the resulting stochastic process by the performance of the intended application. Without loss of generality, we resort to the control performance, as discussed in Section 3.3, to tackle this problem.

We have pointed out in Section 3.3 that the output power of a control system is a proper performance measurement when the task dropout process is modeled as a stochastic process. Given the same task specification, a scheduler essentially determines the task dropout process. So, one could use the output power to indicate the goodness of a scheduler. However, a low output power may not necessarily mean that the task dropout process is close to the MC-based constraint. This can be illustrated by observing the data shown in Fig. 5. For example, when the average dropout rate is 0.12, the dropout process corresponding to the (2, 3)-rule gives a power value of 15 and, when the average dropout rate is 0.35, the dropout process corresponding to the optimal MC constraint gives a power of value 20. Though the former leads to a lower power value, it requires a much lower average dropout rate and is far from the optimal dropout process. A lower average dropout rate demands more resources and may adversely effect the performance of other tasks. A good scheduler should lead to a dropout process that has lower output power for each control task and does not increase the dropout rates of noncontrol tasks.

Based on the above discussion, we propose the following method to compare two schedulers, A and B, when they are applied to a task set: If τ_i in the task set is a control task, we denote the output power resulted from applying A and B by p_i^A and p_i^B, respectively. If τ_j is a noncontrol task, the two measured average dropout rates resulted from applying A and B are denoted by θ_j^A and θ_j^B, respectively. We say that A performs better than B for this task set if the following conditions are satisfied:

- $\theta_j^A \leq \theta_j^B$ for every noncontrol task τ_j and
- $p_i^A \leq p_i^B$ for every control task τ_i.

Note that the above metrics do not impose a total order on different schedulers applied to a given task set. However, one can repeatedly perform the above comparison for many randomly generated task sets and use a scoring system to rank different schedulers. If a scheduler makes the dropout processes of control tasks closely follow the optimal dropout process specified by the MC-based constraint, it tends to receive a higher score since such a scheduler helps to achieve lower output power under the same dropout rate.

We propose a scoring system as follows: Let the number of task sets used for comparing the schedulers be N. If a total of n number of task sets satisfy the two conditions discussed above, scheduler A gets a score of $\frac{n}{N} \times 100\%$. This is similar to comparing alternatives according to multiple, competing criteria [25].

5 Experimental Results

In this section, we present experimental results to evaluate the performance of the scheduling algorithms proposed in the previous section.

5.1 Comparison with EDF

The EDF scheduling algorithm is one of the most commonly used scheduling algorithms in real-time systems. In this section, we compare our scheduling algorithms with nonpreemptive EDF and preemptive EDF. That is, given a collection of task sets, we applied the nonpreemptive and preemptive EDF algorithms to schedule each task set directly and obtained performance results. Then, we used the nonpreemptive/preemptive EDF algorithm to schedule the tasks within the MF and BF groups resulting from employing MDA, DDA, and FDA. Finally, we compared the results obtained from using MDA, DDA, and FDA to those obtained directly from EDF. Below, we first describe our experimental setup and then present the relevant data.

In our experiments, we randomly generated a large number of task sets, each of which contains 5, 10, or 20 tasks. In each task set, some of the tasks (control tasks) are associated with the MC-based constraint, which is the same as the one given in Fig. 1c. The other tasks (noncontrol tasks) simply use the average dropout rate as a QoS metric. The period of each task was randomly selected from a uniform distribution between 2 to 50 time units and the deadline of each task was set to be the same as its period. The execution time distribution of every task was also randomly generated. That is, we first randomly selected a number k_i to be the number of discrete values that τ_i's execution time can take and then generated k_i pairs of random numbers, one as the execution time and the other as the corresponding probability. After a task set was generated, we used the method in [11] to compute the average dropout rate of each task with an MC-based constraint. If the dropout rate of such a task was higher than 40 percent, we discarded the task set because the control system behavior under such a high dropout rate is usually unstable and the comparison of output power is meaningless. We also discarded task sets in which a control task had a dropout rate lower than 2 percent since, in this case, the dropout patterns no longer make any difference (see Fig. 5).

We have developed a simulation environment to simulate the task execution process according to the four schedulers, EDF, MDA, DDA, and FDA. For the DDA and FDA, the upper bound $\overline{\theta}$ could take various values between 40 percent to 50 percent, while the lower bound $\underline{\theta}$ could take values between 3 percent and 5 percent. We found that the trends of the algorithms remain the same for different bound values. Therefore, we will only show the results for $\overline{\theta} = 50\%$ and $\underline{\theta} = 5\%$. Note that, for dropout rates outside these bounds, the system performance is either unstable or insensitive to the dropout pattern variations (see Fig. 5). For a noncontrol task, the dropout rate constraint was set to be the dropout rate obtained from directly applying the EDF algorithm. A job of noncontrol task was put in either the MF or BF group depending on whether the current dropout rate is above or below the constraint.

TABLE 2
Simulation Output for Two Example Task Sets

Scheduler type	Dropout rate for task1 to task5 (in percentage) (Task5 is the only control task)					Output Power
EDF	20.73	10.54	35.93	28.82	26.27	2.337565e+04
MDA	20.47	10.91	35.01	28.35	5.64	6.817112e+00
DDA	20.59	10.98	35.12	28.43	3.28	6.084178e+00
FDA	17.75	9.25	31.56	22.52	47.54	4.592294e+01
EDF	13.95	21.00	5.88	47.17	24.73	3.460549e+01
MDA	14.96	20.04	5.93	41.30	3.69	6.052811e+00
DDA	15.00	20.12	5.99	42.70	0.63	5.248122e+00
FDA	14.88	19.57	5.83	38.22	8.57	1.168510e+01

For each task set, we simulated the execution of each scheduler to obtain the dropout pattern for each task up to one million jobs. We then employed a control system simulator (MATLAB Simulink) to determine the output power of the control tasks. For each task's dropout pattern, we ran the control system simulator three times and recorded the average power. Note that, since obtaining the output power is a statistical analysis process, this means the power value needs to be treated as having an error range. We use the bootstrapping method [26] to obtain the simulation error distribution. That is, first, we randomly generated 100 dropout patterns of different dropout rates. Second, each of these dropout patterns is fed to the simulator and we ran the simulation 10,000 times. Finally, we analyzed the histogram of the result power value of each of these dropout patterns to obtain the error distribution characteristics. We observed that the maximal standard deviation of the simulation results is around 10 percent of the mean value and the probability of a single simulation result being 10 percent larger (or small) than its mean value is less than 0.04. Therefore, in applying the scoring approach, we considered p^A to be different from p^B only if $|p^A - p^B| > \rho \times \min\{p^A, p^B\}$. In our experiments, we let ρ equal 0.2, which will give us a error probability around 0.0016. (By error probability, we refer to the probability of getting a wrong comparison result using the above comparison scheme.) If both p^A and p^B are greater than 100, we ignore the comparison result since a system with power greater than 100 is considered unstable in the control sense. We simulated a large number of task sets in order to collect the statistical behavior of the schedulers. The performances of the different scheduling algorithms are compared by the scoring approach discussed in Section 4.2.

As an example, Table 2 shows the actual simulation results for two task sets, each of which contains four noncontrol tasks and one control task. Rows 2-5 and 6-9 correspond to task sets 1 and 2, respectively. The dropout rates of all five tasks in each task set are given in columns 2-6, where column 6 is for the control task. The last column provides the average output power of the control system. If one examines the data for task set 1, it is easy to see that FDA performs better than EDF because it results in smaller output power and lower dropout rates for noncontrol tasks. In particular, for the control task, though the dropout rate has increased significantly for FDA compared to EDF (47.5 percent versus 26.3 percent), the output power has decreased greatly. This reveals that the

TABLE 3
Scheduler Scores Based on 300 Task Sets (in Percentages) for Tasks Set with Various Number of Tasks

Scheduling scheme	Number of tasks in a task set	EDF vs MDA	MDA vs EDF	EDF vs DDA	DDA vs EDF	EDF vs FDA	FDA vs EDF
non-preemptive	5	0.00	68.33	0.00	65.67	0.00	35.00
	10	0.00	56.00	0.00	49.33	0.00	83.67
	20	0.00	55.67	0.00	51.00	0.00	80.33
preemptive	5	0.00	15.67	0.00	13.67	0.00	28.00
	10	0.33	50.33	0.00	39.33	0.33	68.33
	20	0.00	48.67	0.00	45.33	0.00	56.67

dropout pattern for the FDA case is much closer to the optimal dropout pattern (refer to Fig. 5). Similarly, we can see that MDA and DDA result in smaller output power and similar dropout rates (deviation is smaller or less than 1 percent) for tasks with dropout rate constraints. Of course, in some cases, we may not be able to say one scheduler is definitely better than another. For instance, comparing FDA and EDF for task set 2, we see that the dropout rates for some tasks are decreased, while, for others, the rates are increased.

Table 3 summarizes the comparison results based on a total of 300 task sets. Each task set contains 5, 10, 20 tasks (see rows 2-7) with only one of them being a control task. Six groups of comparisons (see columns 2-7) are provided. By "EDF vs MDA," we mean that the data are based on testing if EDF is better than MDA. Recall that the scoring method presented in Section 4.2 only gives a partial order (in contrast to a total order) when comparing two schedulers applied to the same task set. Thus, we need both "EDF vs MDA" and "MDA vs EDF" to see which is better. The other columns have the same meaning. From Table 3, one can readily observe that, when the non-preemptive scheme is used, EDF never scores against MDA, DDA, and FDA, while the three new algorithms score between 35 percent to 84 percent. When the preemptive scheme is used, the results are similar. Thus, all three new algorithms outperform the EDF algorithm.

Table 4 shows similar information as Table 3 but the task sets now contain two control tasks (instead of one) among five tasks. The data again demonstrates that our algorithms outperform EDF. The better performance of MDA, DDA, and FDA is attributed to the fact that these algorithms have reduced the output power without penalizing the dropout rates of noncontrol tasks.

The fact that our proposed algorithms indeed improve the job dropout processes in terms of meeting the MC-based constraints can be further demonstrated by the plots in Fig. 7. Each point in the plots depicts the resulted control task's performance obtained by applying one particular scheduler to one task set. The optimal dropout process (i.e.,

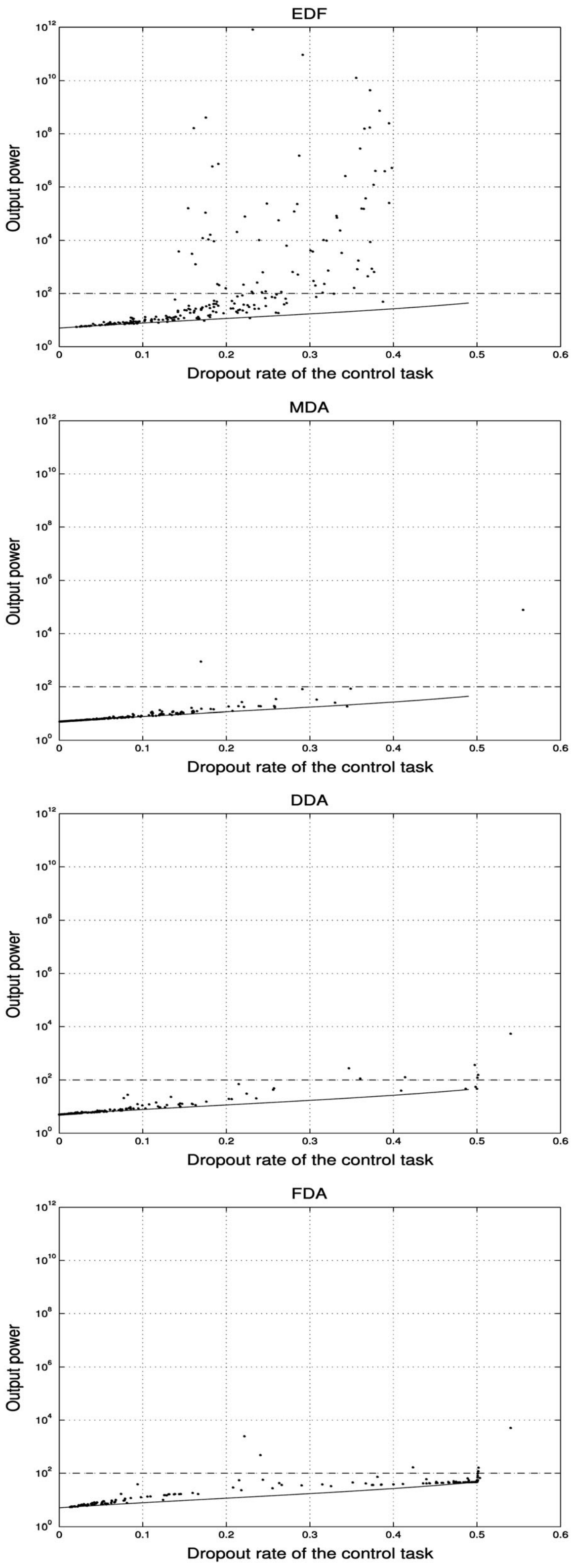


Fig. 7. Output power versus the dropout rate for applying four different schedulers to 300 task sets. Each task sets contains four noncontrol tasks and one control task. The nonpreemptive scheme is used. The optimal dropout process is also shown as the solid line.

TABLE 4
Scheduler Scores (in Percentages) for 100 Task Sets

EDF vs MDA	MDA vs EDF	EDF vs DDA	DDA vs EDF	EDF vs FDA	FDA vs EDF
1	46	1	40	1	31

Each task set contains two control tasks and three noncontrol tasks. The nonpreemptive scheme is used.

TABLE 5
Scheduler's Effect on the Dropout Rates of the Tasks without MC-Based Constraints

Scheduler Type	The number of tasks whose dropout rates are higher than 50 percent	The number of tasks whose dropout rates are higher than 30 percent
MDA	108	238
DDA	105	234
FDA	55	148

Task sets are the same as those for Fig. 7.

the MC-based constraint) corresponds to the curve below all the points in each plot. From the plots, one can readily see that the data corresponding to MDA, DDA, and FDA are close to the optimal dropout curve, while those of EDF in general are not. This is especially true when the dropout rate is high (say greater than 18 percent). A horizontal long-dashed line indicating power being equal to 100 is also shown in the plot. It is clear that EDF resulted in many more points above this line than the other algorithms. Since a control system is considered unstable when the power value is larger than 100, MDA, DDA, and FDA again outperform EDF greatly in this regard.

Fig. 7 also reveals that FDA tends to make tasks have higher dropout rates than MDA and DDA, but output power values of the tasks are still quite low. Thus, FDA seems to be more effective in reducing the output power (i.e., meeting the MC-based constraints) without increasing resource demands. The data in Table 5 also verify this observation. Table 5 shows that, when FDA is used, the number of tasks (without MC-based constraints) whose dropout rates are higher than 50 percent (30 percent) is smaller, compared with when MDA and DDA are used.

We have shown that our scheduling algorithms are better than EDF in a way that they are able to achieve the same or better average dropout rates for the noncontrol tasks while improving the control system performance at the same time for a large percentage of randomly generated task sets. The conclusion should not be surprising as the EDF algorithm does not pay any special attention to the dropout patterns of control tasks.

5.2 Comparison with Window-Based Scheduling Algorithms

Experimental results from the last subsection show that the EDF algorithm, being not mindful of dropout patterns, does not perform well in the presence of MC-based constraints. A natural question to ask is how the scheduling algorithms designed for the window-based constraints behave under the MC-based constraints. In this section, we will compare our scheduling algorithm with the scheduling algorithm introduced in [24] (referred to as WHSA). WHSA makes scheduling decisions based on dropout patterns and is one of the most general and powerful online algorithms for window-based constraints.

The experimental setup for comparing WHSA with our algorithms is the same as the one in the previous subsection. The MC-based constraint for a control task is given in Fig. 1c. In order to make the comparison fair, we need to choose the window-based constraints for WHSA carefully. For a noncontrol task τ_i, $\binom{n_i}{m_i}$ as defined in [4] is used, i.e., in any window of m_i consecutive jobs of a task, at least n_i jobs must meet their deadlines. We let $m_i = 100$ and $n_i = \lceil \varepsilon_i \times 100 \rceil$ (ε_i is the desired average dropout rate of τ_i).

For a control task, because there is no window-based constraint equivalent to the MC-based constraint used, we selected the two window-based constraints most similar to the MC-based constraint, i.e., $\binom{2}{4}$ (in any window of four consecutive jobs of a task, at least two jobs must meet their deadlines), and $\langle {2 \atop 4} \rangle$ (in any window of four consecutive jobs of a task, at least two *consecutive* jobs must meet their deadlines). Note that the partitioning method used by WHSA on noncontrol tasks is the same as the methods used by MDA, DDA, and FDA. The only difference between WHSA and our scheduling algorithms is the way in which the jobs of the control tasks are partitioned between different priority groups. The group assignment in WHSA is deterministic since there are no probabilistic parameters in the window-based constraint. (For more details of WHSA, refer to [24].)

We applied WHSA to the same task sets that are used in Table 3 and used EDF's results as a common base to compare WHSA and our scheduling algorithms. The results are shown in Table 6. There is one control task in each task set. We designate the WHSA using $\binom{2}{4}$ as the constraint as "WHSA1" and that for $\langle {2 \atop 4} \rangle$ as "WHSA2."

From Table 6, we can see that WHSA scores for only a small percentage of task sets against EDF, while MDA, DDA, and FDA do a much better job for the same task sets (as shown in Table 3). This shows that WHSA cannot adequately deal with the MC-based constraints, which are handled nicely by our algorithms.

Readers might question why we use the scoring mechanism to compare WHSA and our scheduling approaches with EDF separately. Recall that the goal of applying WHSA or our scheduling algorithms is to improve

TABLE 6
Scheduler Scores (in Percentages) of the Window-Based Scheduling Algorithm and EDF Based on the Same Task Sets Used in Table 3

Scheduling scheme	Number of tasks in a task set	EDF vs WHSA1	WHSA1 vs EDF	EDF vs WHSA2	WHSA2 vs EDF
non-preemptive	5	0.67	1.33	0.67	2.67
	10	0.33	1.00	0.33	2.33
	20	0.33	0.67	1.00	1.33
preemptive	5	0.00	2.33	0.00	2.67
	10	0.33	0.33	0.33	3.00
	20	0.00	0.00	0.67	2.67

Only one control task in each task set.

TABLE 7
Scheduler Scores Based on 300 Task Sets (in Percentages) for Tasks Set with Various Number of Tasks

Scheduling scheme	Number of tasks in a task set	WHSA1 vs MDA	MDA vs WHSA1	WHSA1 vs DDA	DDA vs WHSA1	WHSA1 vs FDA	FDA vs WHSA1
non-preemptive	5	0.00	0.00	0.00	1.33	0.00	11.33
	10	0.00	0.00	0.00	1.67	0.00	15.67
	20	0.33	0.67	0.33	1.67	0.33	8.33
preemptive	5	0.00	0.33	0.00	0.33	0.00	9.33
	10	0.67	0.33	0.00	2.00	0.00	13.33
	20	0.00	0.67	0.33	2.33	0.00	8.00

TABLE 8
Scheduler Scores Based on 300 Tasks Sets (in Percentages) for Tasks Set with Various Number of Tasks

Scheduling scheme	Number of tasks in a task set	WHSA2 vs MDA	MDA vs WHSA2	WHSA2 vs DDA	DDA vs WHSA2	WHSA2 vs FDA	FDA vs WHSA2
non-preemptive	5	3.00	0.00	0.00	0.00	0.00	4.67
	10	2.67	0.00	2.33	0.67	0.00	5.00
	20	1.33	2.67	2.67	1.67	0.33	3.33
preemptive	5	3.67	1.33	3.33	0.33	1.00	7.00
	10	4.00	2.67	2.00	1.00	2.67	4.33
	20	4.67	1.00	3.00	1.33	1.00	3.67

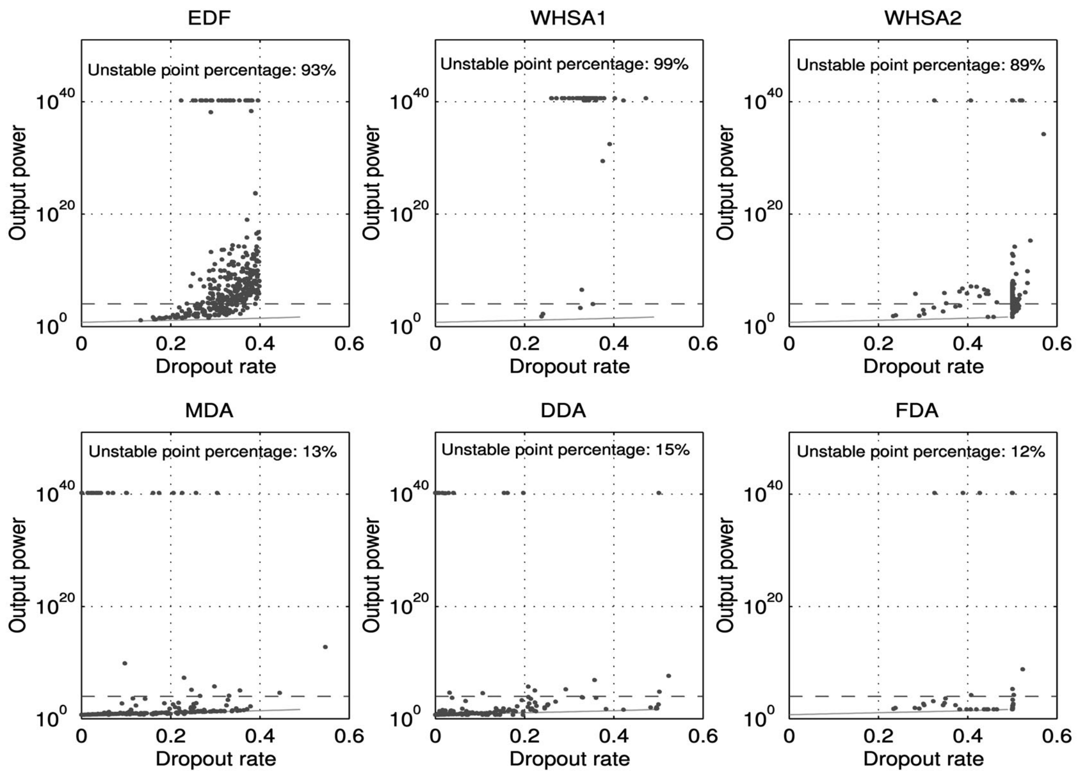


Fig. 8. Output power versus the dropout rate for applying four different schedulers to 300 task sets. Each task set contains nine noncontrol tasks and one control task. The nonpreemptive scheme is used. The optimal dropout process is also shown as the solid line. The horizontal long-dashed line in each plot indicates output power being equal to 100. The "unstable points" are those points whose output power exceeds 100.

the control task's performance without increasing the dropout rates of noncontrol tasks. In WHSA and our scheduling algorithms, the dropout rate constraints of noncontrol tasks are preset to the dropout rates obtained from applying the EDF algorithm. Thus, comparing WHSA and our scheduling algorithms with EDF separately demonstrates how well a scheduling algorithm fulfills the goal, which is the intent of the scoring mechanism.

Direct comparison of WHSA with our scheduling algorithms using the scoring mechanism can be done and the results are shown in Table 7 and Table 8. One can see that neither WHSA nor our algorithms are able to achieve high scores against each other. This can be attributed to the following facts: 1) Only when a scheduler results in a better performance for every task in the task set than another scheduler can the former score 1 point. 2) A scheduler that results in a worse control performance may unnecessarily decrease a noncontrol task's dropout rate (i.e., make a noncontrol task's dropout rate much lower than the dropout rate constraint). The reason that our algorithms do not score significantly higher than WHSA is due to the dropout rates of noncontrol tasks being unnecessarily lower in WHSA.

To examine the performance of WHSA further, we plot the control task's performance of WHSA and our scheduling algorithms in Fig. 8. The meaning of this plot is the same as that of Fig. 7. Since a good number of points overlap each other, some plots, i.e., the plots for WHSA1 and FDA, seem to have fewer points. However, each plot in Fig. 8 contains exactly 300 points. To help readers understand Fig. 8 better, we also show the "unstable point percentage" in each plot. The "unstable points" are those points whose output power exceed 100. (Because the control system is considered to be unstable when the output power is larger than 100.) One can observe that our scheduling algorithms perform much better than WHSA as our algorithms result in much fewer unstable points.

6 Conclusions

In this paper, we consider *firm* real-time tasks with probabilistic execution times (which are often found in control applications). We have introduced a novel QoS constraint which is based on the Markov chain model. The new QoS constraint generalizes both the dropout rate type of constraints and the window-based constraints. More importantly, it provides the flexibility needed to precisely specify the desired stochastic behavior of a system. This is particularly valuable for control systems where the system performance can be expressed as a function of the dropout process's transition matrix. A networked control system has been used to illustrate this point.

For the new QoS constraint, we show that traditional scheduling algorithms may not be adequate. We have presented three online algorithms which attempt to deal with the QoS constraint explicitly. The aim of these algorithms is to lead to job dropout processes as close to the QoS constraint as possible without impacting the utilization of the resource by other tasks. We have developed a simulation environment to help us evaluate different scheduling algorithms. The experimental results indicate that our scheduling algorithms indeed outperform scheduling algorithms that do not consider the QoS constraint. We also compare our algorithms with the algorithm in [24], which is designed to satisfy certain dropout patterns represented as window-based constraints. The result shows that the algorithm in [24] performs better than EDF, but it is far less effective than our algorithms.

As for future work, improvements can be made by incorporating various admission control schemes to help adjust the jobs in the different groups. We plan to investigate this and other directions.

Acknowledgments

This research is supported in part by the US National Science Foundation under grant numbers CNS-0410771, CCR-0208537, and ECS-0225265. Part of the work by X. Sharon Hu was carried out during her sabbatical visit to the Hong Kong University of Science and Technology.

References

[1] T.F. Abdelzaher, K. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 1, pp. 80-96, Jan. 2002.
[2] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang, "On the Competitiveness of On-Line Real-Time Task Scheduling," *Real-Time Systems,* vol. 4, pp. 125-144, 1992.
[3] S. Baruah, J. Haritsa, and N. Sharma, "Scheduling for Overload in Real-Time Systems," *IEEE Trans. Computers,* vol. 46, no. 9, pp. 1034-1039, Sept. 1997.
[4] G. Bernat, A. Burns, and A. Llamosi, "Weakly Hard Real-Time Systems," *IEEE Trans. Computers,* vol. 50, no. 4, pp. 308-321, Apr. 2001.
[5] P.R. Blevins and C.V. Ramamoorthy, "Aspects of a Dynamically Adaptive Operating Systems," *IEEE Trans. Computers,* vol. 25, no. 7, pp. 713-725, July 1976.
[6] S. Brandt, G. Nutt, T. Berk, and J. Mankovich, "A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage," *Proc. Real-Time Systems Symp.,* pp. 307-317, 1998.
[7] J.L. Diaz, D.F. Garcia, K. Kim, C. Lee, L. Lo Bello, J.M. Lopez, L.M. Sang, and O. Mirabella, "Stochastic Analysis of Periodic Real-Time Systems," *Proc. Real-Time Systems Symp.,* pp. 289-300, 2002.
[8] M. Hamdaoui and P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines," *IEEE Trans. Computers,* vol. 44, pp. 1443-1451, 1995.
[9] J.R. Haristsa, M. Livny, and M.J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems," *Proc. Real-Time Systems Symp.,* pp. 232-242, 1991.
[10] A. Kalavade and P. Mogh, "A Tool for Performance Estimation of Networked Embedded End-Systems," *Proc. Design Automation Conf.,* pp. 257-262, 1998.
[11] S. Manolache, P. Eles, and Z. Peng, "Memory and Time-Efficient Schedulability Analysis of Task Sets with Stochastic Execution Time," *Proc. 13th Euromicro Conf. Real-Time Systems,* pp. 19-26, 2001.
[12] G. Koren and D. Shasha, "Skip-Over: Algorithms and Complexity for Overloaded Systems that Allow Skips," *Proc. Real-Time Systems Symp.,* pp. 110-117, 1995.
[13] Q. Ling and M.D. Lemmon, "Robust Performance of Soft Real-Time Networked Control Systems with Data Dropouts," *Proc. IEEE Conf. Decision and Control,* vol. 2, pp. 1225-1230, 2002.
[14] Q. Ling and M.D. Lemmon, "Soft Real-Time Scheduling of Networked Control Systems with Dropouts Governed by a Markov Chain," *Proc. Am. Control Conf.,* 2003.
[15] S.K. Park and K.W. Miller, "Random Number Generators: Good Ones Are Hard to Find," *Comm. ACM,* vol. 31, no. 10, pp. 1192-1201, 1988.
[16] C. Lu, J.A. Stankovic, S.H. Son, and T. Gang, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *Real-Time Systems,* vol. 23, nos. 1-2, pp. 85-126, 2002.
[17] G. Quan and X. Hu, "Enhanced Fixed-Priority Scheduling with (m, k)-Firm Guarantee," *Proc. IEEE Real-Time Systems Symp.,* pp. 79-88, 2000.
[18] P. Ramanathan, "Overload Management in Real-Time Control Applications Using (m, k)-Firm Guarantee," *IEEE Trans. Parallel and Distributed Systems,* vol. 10, no. 6, pp. 549-559, June 1999.
[19] D.R. Sahoo, S. Swaminathan, R. Al-Omari, G. Manimaran, M. Salapaka, and A. Soomani, "Feedback Control for Real-Time Scheduling," *Proc. Am. Control Conf.,* vol. 2, pp. 1254-1259, 2002.
[20] J.A. Stankovic, C. Lu, S.H. Son, and T. Gang, "The Case for Feedback Control Real-Time Scheduling," *Proc. 11th Euromicro Conf. Real-Time Systems,* pp. 11-20, 1999.
[21] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.W.-S. Liu, "Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times," *Proc. Real-Time Technology and Applications Symp.,* pp. 164-173, 1995.
[22] R. West and C. Poellabauer, "Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams," *Proc. Real-Time Systems Symp.,* pp. 239-248, 2000.

[23] A.K. Mok and W. Wang, "Window-Constrained Real-Time Periodic Task Scheduling," *Proc. IEEE Real-Time Systems Symp.*, pp. 15-24, 2001.

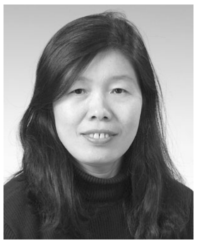
[24] G. Bernat and R. Cayssials, "Guaranteed On-Line Weakly-Hard Real-Time Systems," *Proc. IEEE Real-Time Systems Symp.*, pp. 25-34, 2001.

[25] T. Loukil, J. Teghem, and D. Tuyttens, "Solving Multi-Objective Production Scheduling Problems Using Metaheuristics," *European J. Operational Research*, 2003.

[26] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap.* Chapman and Hall, 1993.

[27] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, 1989.

Donglin Liu received the BS degree in electrical engineering from Tianjin University, China, in 1996. He received the MS degrees in computer science and engineering from the University of Notre Dame in 2005. His primary research interests are in the area of networked control system and real-time system scheduling algorithm design and evaluation.



Xiaobo Sharon Hu (S'85-M'89-SM'02) received the BS degree from Tianjin University, China, the MS degree from the Polytechnic Institute of New York, and the PhD degree from Purdue University, West Lafayette, Indiana. She is an associate professor in the Department of Computer Science and Engineering at the University of Notre Dame. She also worked at General Motors Research Laboratories as a senior research engineer and at Western Michigan University as an assistant professor. Her research interests include hardware-software codesign, real-time embedded systems, low-power system design, and design automation algorithms. She has published more than 90 papers in related areas. She has served on the program committees of a number of conferences, such as the Design Automation Conference, International Conference on Computer-Aided Design, Design, Automation and Test in Europe Conference, IEEE Real-Time Systems Symposium, etc. She was the program cochair of the International Symposium on Hardware-Software Codesign in 2001 and the general cochair of the same conference in 2002. She is a senior member of the IEEE.

Michael D. Lemmon received the BS degree in electrical engineering from Stanford University in 1979. He received the MS and PhD degrees in electrical and computer engineering from Carnegie-Mellon University in 1987 and 1990, respectively. He joined the University of Notre Dame in 1990, where he currently holds an appointment as a professor of electrical engineering. He served as program chair for the Fifth International Workshop on Hybrid Systems in 1997 and the 1999 International Symposium on Intelligent Control. He is a former associate editor of the *IEEE Transactions on Neural Networks* and the *IEEE Transactions on Control Systems Technology*. Dr. Lemmon's primary research interests are in the area of control theory, including hybrid control systems and supervisory control of discrete-event systems. He is currently studying decentralized control, detection, and estimation over ad hoc sensor-actuator networks.

Qiang Ling received the BS degree in automatic control from the University of Science and Technology of China in 1997, the ME degree in control theory and control engineering from Tsinghua University in 2000, and the PhD degree in electrical engineering from University of Notre Dame in 2005. He joined Seagate Technology, Pittsburgh, Pennsylvania, in 2005. His research interests include stochastic control, networked control systems, and quantized control systems.