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

Abstract

Many real-time systems have firm real-time requirements which allow occasional deadline violations but discard any tasks that are not finished by their deadlines. To measure the goodness 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 constraint. However, for certain applications, these metrics may not be adequate measures of system performance. This paper introduces a novel QoS constraint for networked feedback control systems. 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 compare these approaches.

1 Introduction

Real-time systems are usually classifi ed as being *hard*, *soft* and *firm*. 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 fi nished by their deadline are still completed albeit with a reduced value. Firm real-time systems also allow occasional deadline misses. But, unlike soft real-time systems, if a task is not completed by its deadline, the task is considered valueless and is discarded (or dropped) [4]. Examples of fi rm real-time systems can be found in many control and multimedia applications.

Quantifying the "occasional deadline misses" is critical for evaluating the Quality of Service (QoS) of a fi rm real-time system (FRTS). One often used QoS metric for FRTS is the deadline miss ratio or average dropout rate (as a task is dropped if it cannot be fi nished by its deadline), defi ned as the percentage of the average number of deadline misses with respect to the number of tasks 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. A problem with these metrics is that they cannot express the information about how the deadline misses or dropouts are distributed. In some systems, such as those found in control and multimedia applications, the system performance is very sensitive to the dropout patterns. For example, if dropouts occur consecutively for several invocations of the same task, then the system performance may be totally unacceptable.

To overcome the shortcoming of the dropout rate based metrics, a variety of window-based QoS constraints have been proposed for FRTS. A window refers to a fixed number of consecutive invocations or jobs of a task. Hamdaoui and Ramanathan introduced the (m, k)-firm constraint in [9]. 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* [13] constraint, where a task with a skip factor of s is allowed to have one invocation skipped out of s consecutive ones. In [4], Bernat, et. al., proposed a set of firm constraints based on the desired deadline miss patterns. They suggested, for example using the following constraints to characterize the dropout pattern: (i) meeting at least n consecutive deadlines in a window of m consecutive invocations, (ii) missing less than n consecutive deadlines in a window of m consecutive invocations.

The window-based QoS constraints provide a more comprehensive measure of deadline misses than the average dropout rate. The associated schedulability analyses are carried out assuming the complete knowledge of the tasks is available. Unfortunately, many real-time systems must face uncertainties inherent in executing software tasks themselves as well as those arising from the environment. Simply using the worst case parameters would result in overly expensive system implementations. Furthermore, it may be inadequate to use just two parameters (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.

This paper introduces a novel QoS constraint for FRTS that is more general and flexible than the window-based constraints mentioned above. The constraint is specified based on the Markov chain (MC) process. It will become clear later that the model 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 control systems, this constraint can be directly related to the overall control system's 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.

To meet an MC-based constraint, previous scheduling results (e.g., [4, 9, 13, 18, 19]) are no longer applicable since these methods assume a deterministic dropout pattern. We present several scheduling alternatives for the 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 monitored variables 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 general-purpose operating systems scheduling. In recent years, using feedback information for scheduling has also seen an increase in the real-time system area. For example, in [10], the authors proposed *Adaptive Earliest Deadline* (AED) priority assignment policy to stablize the performance of *Earliest Deadline First* (EDF) under overload situations. Brandt and his colleagues presented a dynamic QoS manager (DQM) to change task QoS levels according to the sampled central processing unit (CPU) utilization or deadline misses [6]. Another group of works in the real-time system area can be categorized as applying feedback control theory to scheduling (e.g., [1, 17, 20, 21]). Similar to [6], these works (except [20]) target soft real-time systems where each task has multiple versions (or QoS levels) with different values.

In our work, real-time tasks do not have multiple versions but have firm deadline constraints. They are periodic but their execution times can vary between the best case execution time (BCET) and worst case execution time (WCET). Since our MC-based QoS constraint is different from those used in all the previous work, we present several alternative scheduling approaches and evaluate their effectiveness.

2 Preliminaries

We consider a system consisting of periodic tasks. The i-th task is denoted as τ_i. Each instance of a task is called a *job*. The jth job of τ_i is denoted as τ_{ij}. 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 represented by a random variable C_i. Without loss of generality, we assume that C_i is a discrete random variable which takes values $c_{i1}, c_{i2}, \ldots, c_{ik_i}$ with probability $p_{i1}, p_{i2}, \ldots, p_{ik_i}$, respectively. The jth job of task τ_i, τ_{ij}, requires C_{ij} time to finish. The probability that C_{ij} equals to c_{ih} is p_{ih}. The distribution of C_i could be obtained from experimental data or through profiling.

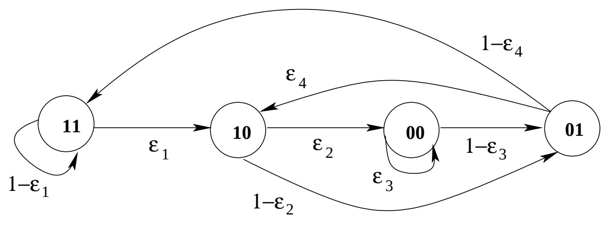
In this paper, we study systems in which a task is discarded (or dropped) if it is not completed by its deadline. This practice is adopted in many feedback control systems [14]. We use f_{ij} to denote the completion status of job τ_{ij}, i.e., $f_{ij} = 1$ if τ_{ij} is completed at or before its deadline 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 even started. For simplicity, we refer to both cases as the jobs being dropped. The average dropout rate of task τ_i is denoted by $\overline{\epsilon}(i)$.

3 A Markov Chain Based QoS

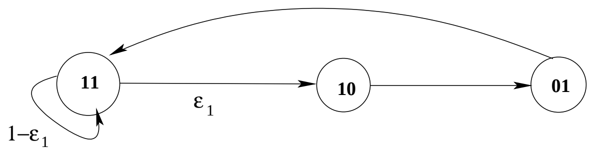
Many real-time systems demonstrate probabilistic behavior due to uncertainty inherent in the operating environment and the task parameters. The patterns of jobs being completed or dropped can have a significant impact on the overall system performance. To capture the dependency of QoS on both job dropout patterns and system probabilistic behavior, we propose using a Markov chain (MC) to describe the desired stochastic job dropout behavior associated with task τ_i. We call this the MC constraint of τ_i and denote it by $\mathcal{MC}_i$.

Specifically, $\mathcal{MC}_i$ is a discrete stochastic process with two or more states and the transition probability from one state to another denotes either the probability of the next job being dropped or the probability of the next job being completed. Each state is denoted by a specific string of job's completion status bits f_{ij} to represent the recent execution history pattern. For example, if we use two bits to represent the string of the job completion status bits (i.e., we only use the execution status of the two most recent jobs), the state which represents the history pattern of one completed job followed by one dropped job can be written as $\{10\}$. (We use the rightmost bit to represent the completion status of the most recent job). Note that each state can transition to at most two other states with non-zero transition probabilities. For instance, state $\{10\}$ can only transition to either state $\{00\}$ (if the next job is dropped) or state $\{01\}$ (if the next job is completed). Figure 1(a) depicts the general MC process for 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. Figure 1(b) shows a specific example of an MC constraint, which has three states and ϵ_1 represents the dropout probability at state $\{11\}$. It is not difficult to verify that this is equivalent to the (m, k) constraint [9] where $m = 2$ and $k = 3$.

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 constraint then specifies the desired stochas-



(a) The general MC dropout process.



(b) An example MC constraint.

Figure 1: Two Markov chains for the case where two bits are used for the history pattern.

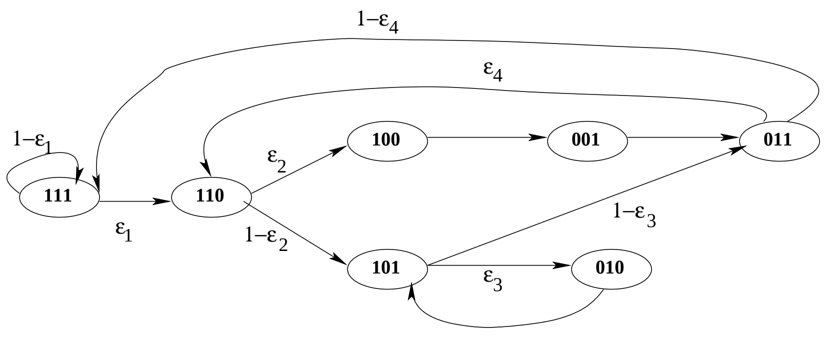


Figure 2: An example MC constraint. It is equivalent to the constraint that at least 2 consecutive jobs must meet the deadline in any window of 4 consecutive jobs.

tic dropout process. The length of the bit string associated with a state (i.e., the number of consecutive jobs observed) determines the general structure (the maximum number of states and all possible transitions) of a MC constraint. For example, if we use 4 bits in each state of a MC constraint, the maximum number of states is 2^4. In practice, the number of bits used is not large. Figure 1 has shown an example MC constraint. As another example, we show in Figure 2 the MC constraint corresponding to the window-based constraint that *at least* 2 *consecutive* jobs must meet the deadline in any window of 4 consecutive jobs. Again, ϵ_i for $i = 1, 2, 3, 4$ is a dropout probability. (Note that the mapping from a window-based constraint to an MC constraint is not unique.)

Given an MC constraint, one can readily compute the average dropout rate. Thus, an MC constraint generalizes the dropout rate constraint. Moreover, a window-based constraint can be represented as an MC constraint as illustrated in Figure 1 and 2. However, what makes the MC-based constraint attractive is that it provides much more descriptive power than both of them. One might argue that a MC constraint can be replaced by simply combining the dropout rate constraint and a window-based constraint. But, from the above two MC examples, one can see that a window-based constraint does not care about the actual values of the dropout probabilities, i.e., ϵ_i. On the other hand, an average dropout rate cannot uniquely define the dropout probabilities ϵ_i. Therefore, the MC-based constraint facilitates a more precise description of the allowed stochastic dropout patterns.

Besides having more descriptive power, the MC-based constraint can also quantitatively relate the dropout probabilities to the performance of a real-time control system. This assertion is based on recent results [15] characterizing the performance of *networked control systems* (NCS) in the presence of data dropouts satisfying the MC constraint. This characterization is important for it allows us to directly evaluate different MC constraints in terms of their impact on application performance.

A *networked control system* is a control system whose feedback path is realized over a communication network. Feedback measurements are occasionally dropped for one of two reasons; either the medium is unreliable or else the measurement is purposely dropped to stay within an allocated transmission rate. The transmission rate allocation is provided by the network as a means of congestion control. The results in [15] focus on a so-called discrete-time *generalized regulator problem* [8] in which unity gain feedback is used to reject a white noise disturbance injected into the system. The lefthand block diagram in Figure 3 shows the NCS under consideration. This system consists of a loop function $L(z)$ whose input, $w[n]$, is a zero-mean white noise disturbance. The output $y[n]$ is fed back through the network in a manner that is controlled by a *dropout process*, $\{d[n]\}$. The dropout process is a binary random process that is generated by an underlying Markov chain with transition matrix Q. Since this particular control system tries to reject disturbances, the system's performance is characterized by the output signal power, (denoted as $\|y\|_{\mathcal{P}} = \mathrm{Trace}\mathbf{E}\left[y[n]y^T[n]\right]$). The main result in [15] provides a systematic method of computing $\|y\|_{\mathcal{P}}$ as a function of the dropout process' transition matrix.

The result in [15] was framed as an optimization problem whose solution is a dropout process that maximizes overall system performance (i.e. minimizes output signal power) subject to a lower bound on the average dropout rate. In other words, the solution to this problem provides the statistical pattern of dropouts that degrade overall control system performance as little as possible for a given dropout rate. The "optimal" dropout process represents the best control system performance that we can obtain for a fixed transmission rate constraint. By directly comparing the performance of other QoS constraints against this optimal dropout process' performance, we obtain an objective means of evaluating different QoS constraints.

Such a comparison is found in the right-hand plot in Figure 3. This figure is taken from [15] and it compares the output signal power (performance) of a specific control system under three different types of dropout processes. In this example, the open loop system was unstable with transfer function $L(z) = \frac{z+2}{z^2+z+2}$. Assuming unity gain feedback, the closed loop transfer function becomes $z^{-1} + 2z^{-2}$ which is asymptotically stable. In other words, this is a networked control system in which dropouts cause us to switch between a stable and unstable system configuration. The output signal power for this system was evaluated for three different types of dropout processes; the "optimal" dropout process, a $(2,3)$-firm guarantee process, and process in which dropouts are independent and identically distributed (i.i.d.). The graph in Figure 3 plots the system's output power as a function of the average dropout rate.

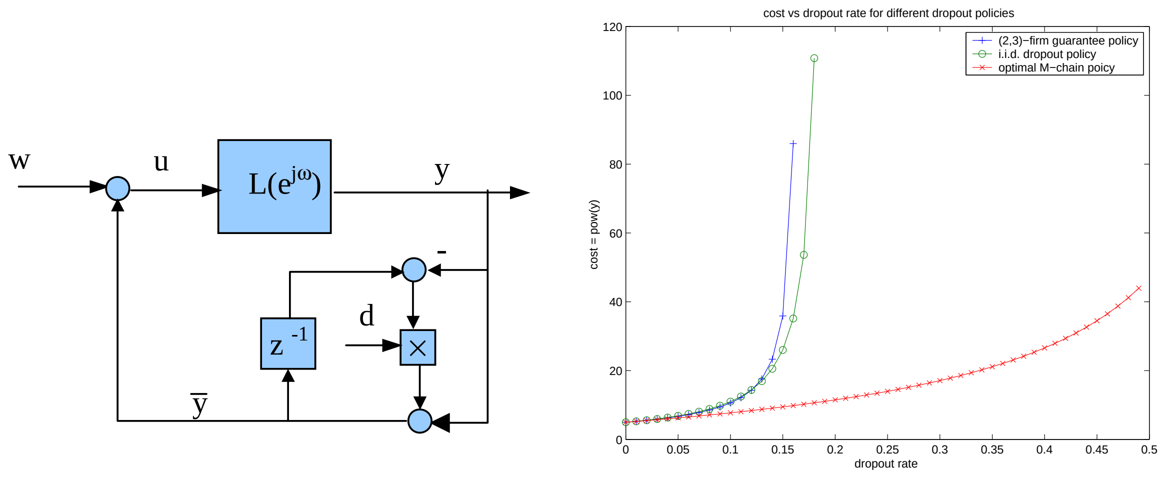


Figure 3: Left - NCS model: Right - comparison of the three dropout models

The results in Figure 3 show that the "optimal" policy performs much better than either the $(2,3)$-rule or i.i.d. dropouts. Note that systems driven by the $(2,3)$-rule and i.i.d. dropout processes go unstable at dropout rates in excess of 20%. It is important to note that the optimal dropout process greatly extends the region of stability for the closed loop system. What is, however, more surprising is the form of the "optimal" dropout policy. For this particular example (see [15] for details), the optimal dropout process always requires that if a single dropout occurs, the next measurement must be dropped as well. In other words for this particular system the optimal dropout policy is to drop two measurements in a row. This optimal dropout policy can be represented by the MC constraint given in Figure 4. (The $(2, 3)$-rule can be described by the MC constraint given in Figure 1). The performance comparison of the two dropout policies runs directly counter to the

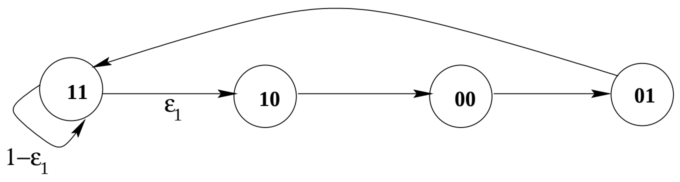


Figure 4: The Markov chain represents the optimal dropout process for the NCS model in Figure 3.

intuition inherent in the (m, k)-rule where one tries to minimize the number of consecutive drops. In fact, for this particular example, the (m, k)-rule does not perform any better than the i.i.d. dropout process.

The results shown in Figure 3 are not representative of all networked control systems, but 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 require a much more flexible approach in characterizing QoS constraints. We believe that the MC constraint provides such flexibility.

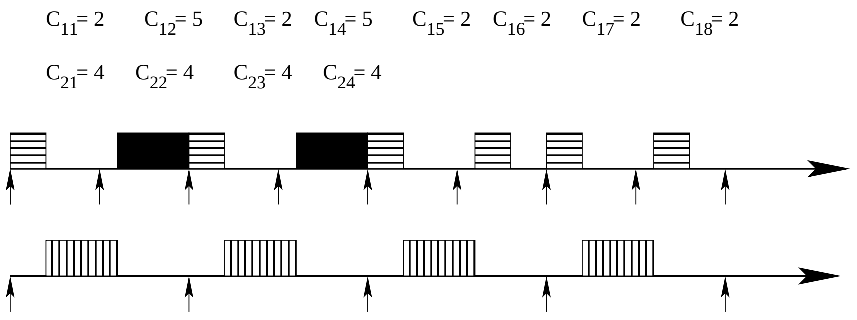
4 Design and Evaluation of Scheduling Approaches

Given a set of real-time tasks and MC constraints associated with some of the tasks, one could still use a scheduling algorithm designed without considering MC 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. Assume that τ_1 is a task implementing the controller for a feedback control system and is associated with the MC constraint given in Figure 4.

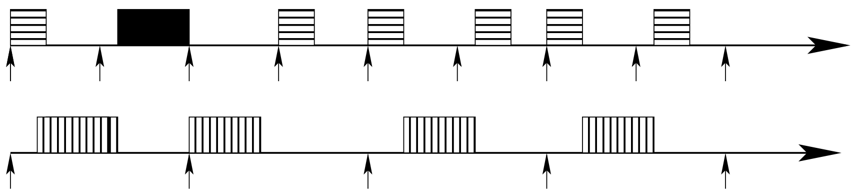
Table 1: The timing parameters of a simple task set.

τ_i	t_i	d_i	(c_{i1}, p_{i1})	(c_{i2}, p_{i2})
τ_1	5	5	(2, 0.75)	(5, 0.25)
τ_2	10	10	(4, 1.0)	

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. Figure 5(a) depicts the resulting execution pattern. If we assume that the execution pattern is repeated infinitely, then the average dropout rate of τ_1, $\overline{\epsilon}(1)$, is equal to 0.25. The execution pattern of τ_1 can be described as 10101111 where 1 means a job meets the deadline and 0 otherwise. We could use some other algorithm to obtain a different execution pattern shown in Figure 5(b). Note that the execution pattern now is 10011111, but $\overline{\epsilon}(1)$ is still 0.25. Task τ_2 in both cases always finishes by its deadline. The only difference between the two systems is the dropout patterns of τ_1. In terms of meeting the MC constraint, the execution pattern in Figure 5(b) is clearly more desirable.



(a) An execution pattern for the task set in Table 1. The second and fourth job of task1 miss their deadlines.



(b) An execution pattern for the task set in Table 1. The second job of task1 misses its deadline and the thrid job of task1 is dropped.

Figure 5: Two different execution patterns for the tasks in Table1. Jobs that miss their deadlines are colored black.

The above example reveals that simply using existing scheduling algorithms may not help to satisfy MC constraints and that it is worthwhile to investigate scheduling approaches for systems having MC constraints. Before we present our new scheduling approaches, we first summarize how an MC constraint is specified. An MC constraint is given by the transition probabilities which are functions of the average dropout rate $\overline{\epsilon}$. Since each state can only transition to one of two states depending whether the next job is dropout or not, only the dropout probability at each state is needed to uniquely specify a MC constraint. For example, suppose we use two bits to represent the most recent dropout history. For the MC constraint in Figure 4, we have $\epsilon_1 = \overline{\epsilon}/(2+3\overline{\epsilon}), \epsilon_2 = 1, \epsilon_3 = 0, \epsilon_4 = 0$ (where ϵ_1 to ϵ_4 are the parameters in the general MC process in Figure 1(a)). In general, the dropout probabilities cannot be expressed as a closed form formula of $\overline{\epsilon}$ but are given in a table. The average dropout rate $\overline{\epsilon}$ 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 presented to the system. We use $\overline{\epsilon}^u$ and $\overline{\epsilon}^l$ to represent the upper and lower bound on $\overline{\epsilon}$. The dropout probabilities as well as the $\overline{\epsilon}$ bounds will be used by the scheduling approaches.

A good scheduler in term of meeting the MC constraint should have the resulting dropout process be as close to the MC constraint as possible. However, 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

All three scheduling algorithms are online algorithms. The common idea of them is to partition jobs dynamically into different groups. Some groups are given higher priorities than others. This is similar to the mandatory v.s. optional job partitioning used in [18, 19]. Instead of two-way partition, we use three groups as *Must Finish* (MF), *Better Finish* (BF) and *Optional Finish* (OF). As the names indicate, the three groups have decreasing priorities. The tasks in the fi rst two groups are executed by using a priority-based scheduling algorithm such as EDF or RM while the tasks in the OF group can be scheduled by using a randomized priority assignment similar to [10]. A hard real-time job is always put in the MF group while the BF group contains tasks that demand best effort. Jobs with MC constraints are partitioned between MF and OF group. By judiciously partitioning the jobs into different groups, our goal is to make the job dropout processes follow the MC constraints as much as possible and to bound the average dropout rate by $\overline{\epsilon}^u$ and $\overline{\epsilon}^l$. The three algorithms differ in their ways of partitioning the jobs and are discussed in detail in the rest of the section.

4.1.1 MC Driven Algorithm (MDA)

The MDA algorithm directly uses MC constraints to partition jobs. That is, the dropout probabilities of the given Markov chain 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 specifi c 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 $\overline{\epsilon}^u$ and $\overline{\epsilon}^l$. 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).

Specifi cally, for a given task set and a chosen scheduling algorithm such as EDF or RM, we fi rst apply *offline* an estimation algorithm such as the ones introduced in [7, 11, 12, 22] to obtain the average dropout rate for each task. (An alternative to using an estimation method is simulation.) Based on the dropout rate for τ_i, we obtain the dropout probability at each state of $\mathcal{MC}_i$. Let the number of bits used to represent the state in $\mathcal{MC}_i$ be n_i. During run time, the scheduler records the dropout status of the most recent n_i jobs. Assuming the newly arrived job is τ_{ij}, the current state of the system is specifi ed by $X(i) = \{f_{i(j-n_i)} f_{i(j-n_i-1)} \cdots f_{i(j-2)} f_{i(j-1)}\}$. If $X(i)$ corresponds to the k-th state in $\mathcal{MC}_i$, the dropout probability ϵ_k is readily obtained. Then, τ_{ij} is put in the OF group with probability ϵ_k. Otherwise, it is put in the MF group.

The complexity of this scheduling algorithm depends on the lengths of the most recent execution patterns to be kept (i.e., n_i's) and the random number generation. Usually the value of n_i is small, and modern random number generators are quite effi cient as shown in [16].

4.1.2 Dropout rate Driven Algorithm (DDA)

The main difference between the MDA algorithm and DDA algorithm is that the latter retains for each task a “long” history execution pattern instead of just the most recent execution pattern used to determine the previous state of the system dropout process. This execution pattern feeds back a task's dropout status history to the scheduler, which uses the feedbakc information to compute the estimated average dropout rate, $\tilde{\epsilon}$, at the current time instant. The length of this history execution pattern, m, is a parameter set by the user. The DDA algorithm also does not require the computation of the estimated deadline miss ratio as the MDA algorithm (though the ratio may be used as the upper bound on $\overline{\epsilon}$ if desired).

During run time, the DDA algorithm uses the long history execution pattern for τ_i, represented by an m bit binary number, to compute $\tilde{\epsilon}$(i), i.e., $\tilde{\epsilon}(i) = \sum_{k=1}^{m}(1 - f_{j-k})/m$ where τ_{ij} is the newly arrived job. The bits f_{j-1} to f_{j-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 $\tilde{\epsilon}(i) > \overline{\epsilon}^u(i)$, put τ_{ij} in the MF group.
2. If $\tilde{\epsilon}(i) < \overline{\epsilon}^l(i)$, put τ_{ij} in the OF group.
3. If $\overline{\epsilon}^l(i) \leq \tilde{\epsilon}(i) \leq \overline{\epsilon}^u(i)$, find the dropout probability corresponding to the current state and the $\tilde{\epsilon}$(i) value. τ_{ij} is put in the OF group with this dropout probability. Otherwise, it is put in the MF group.

The rational behind the DDA algorithm is to make the average dropout rate stay within the given bounds and the dropout process follow the MC constraint for $\tilde{\epsilon}$ within the given bounds. DDA is less efficient than MDA since the history patterns maintained by DDA are usually much longer than those by MDA. However, the length of the history patterns should not be too big so that the recent dropout process is not overshadowed by the dropout process long time ago. In our experiments, we found that setting m to 100 is reasonable.

4.1.3 Feedback Driven Algorithm (FDA)

The FDA algorithm uses a somewhat different way to partition jobs from the above two. It avoids the use of a random number generator while still tries to achieve the same goal as the other algorithms. Similar to DDA, for each task τ_i, FDA retains a "long" history execution pattern to assist the computation of $\tilde{\epsilon}(i)$. Furthermore, this history pattern is used to compute the estimated dropout probability at each state, $\tilde{\epsilon}_k(i)$, in the MC constraint for τ_i.

During run time, upon arrival of τ_{ij}, the FDA algorithm first computes $\tilde{\epsilon}$(i) and determines the current state of the dropout process similar to DDA. Then it computes $\tilde{\epsilon}_k(i)$ for the current state. It also finds the desired dropout probability $\epsilon_k(i)$ for the average dropout rate $\tilde{\epsilon}$ from the MC constraint specification as MDA. The algorithm finally applies the following rules to decide in which group the newly arrived job is placed.

1. If $\tilde{\epsilon}(i) > \overline{\epsilon}^u(i)$, put τ_{ij} in the MF group.
2. If $\tilde{\epsilon}(i) < \overline{\epsilon}^l(i)$, put τ_{ij} in the OF group.

3. If $\overline{\epsilon}^l(i) \leq \tilde{\epsilon}(i) \leq \overline{\epsilon}^u(i)$ and $\tilde{\epsilon}_k(i) > \epsilon_k(i)$, τ_{ij} is put in the MF group. Otherwise, it is put in the OF group.

The FDA algorithm monitors the dropout probabilities as well as the average dropout rate and uses both to determine job partition. Its philosophy is similar to DDA except that it avoids the use of a random number generator by employing the observed dropout probabilities. Thus, FDA is more effi cient than DDA, but it does require some more memory during run time than MDA.

4.2 Evaluation of scheduling algorithms

We have proposed three scheduling algorithms to help meet MC 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 constraint as possible. However, it is diffi cult to quantify the "closeness" of one stochastic process to another. We resort to the control system performance to tackle this problem.

We have pointed out in Section 3 that the output signal 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 specifi cation, a scheduler essentially determines the task dropout process. So one could use the output signal power to indicate the goodness of a scheduler. However, a low output signal power may not necessarily mean that the task dropout process is close to the MC constraint. This can be illustrated by observing the data shown in the right-hand plot of Figure 3. 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 would demand more resource utilization and may adversely effect the performance of other tasks. Thus, a good scheduler should lead to a dropout process that has lower output signal power for each task with an MC constraint and does not increase the dropout rate of tasks without MC constraint.

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 has an MC constraint, we denote the output signal power resulted from applying A and B by $p_A(i)$ and $p_B(i)$, respectively. If τ_j only has a dropout rate constraint (or is a hard deadline task), the two measured average dropout rates resulted from applying A and B are denoted by $\tilde{\epsilon}_A(j)$ and $\tilde{\epsilon}_B(j)$, respectively. We say that A performs better than B for this task set if the following is true

- $\tilde{\epsilon}_A(j) \leq \tilde{\epsilon}_B(j)$ for every τ_j having only a dropout rate constraint, and
- $p_A(i) \leq p_B(i)$ for every τ_i having an MC constraint.

Note that this metric does not impose a total order on different schedulers applied to a given task set. However, one can repeatly perform the above for many task sets and use a scoring system to rank different schedulers. If a scheduler makes the dropout processes of tasks with MC constraints follow closely the optimal dropout process specified by the MC constraint, it tends to receive a higher score since such a scheduler helps to achieve lower output signal power given the same dropout rate.

5 Experimental Results

In this section, we present some experimental results to evaluate the performance of the scheduling algorithms proposed in the previous section. We chose the nonpreemptive EDF algorithm as our comparison target for its simplicity. That is, we applied the nonpreemptive EDF algorithm to schedule each task set directly and obtained performance results. Then, we used the nonpreemptive EDF algorithm to schedule the tasks within the MF and BF groups which are resulted from employing our MDA, DDA and FDA algorithms. 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 five tasks. In each task set, some of the tasks are associated with an MC constraint which is the same as the one given in Figure 4. The other 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 15 time unit, 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 the 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 [12] to compute the average dropout rate of each task with an MC constraint. If the dropout rate of such a task was higher than 40%, we discarded the task set because the control system behavior under such a high dropout rate is usually unstable and the comparison of output signal power is meaningless. We also discarded a task set if a task with an MC constraint had a dropout rate lower than 2% since in this case the dropout patterns no longer make any difference (see the right-hand plot in Figure 3).

We have developed a simulation environment to simulate both the task execution process as well as the four schedulers, EDF, MDA, DDA and FDA. For the DDA and FDA algorithms, the upper bound $\overline{\epsilon}^u$ was set to be 50% and the lower bound $\overline{\epsilon}^l$ was set to be 5%. Note that for dropout rates outside these bound, the system performance is either unstable or insensitive to the dropout pattern variations (see Figure 3). For a task with only the dropout rate constraint, the rate constraint was set to be the dropout rate obtained from directly applying the EDF algorithm. A job of such a task was put in either the MF or BF group depending on the current dropout rate is above or below the constraint. For each task set, we ran the simulation under 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 signal power of the tasks with an MC constraint. For each task's dropout pattern, we ran the control system simulator three times and recorded the average power to compensate for simulation errors.

As an example, Table 2 shows the actual simulation results for two different tasks. Rows 2-5 and 6-9 correspond to task set 1 and 2, respectively. The dropout rates of all five tasks in each task set are given in column 2-6, where column 6 is for the task with an MC constraint. The last column provides the average output signal power. If one examines the data for task set 1, it is easy to see that FDA performs better than EDF because it resulted smaller output signal power and lower dropout rates for tasks with dropout rate constraints. In particular, for the task with the MC constraint, though the dropout rate has increased significantly for FDA compared to EDF (47.5% v.s. 26.3%), the output signal power had decreased greatly. This reveals that the dropout pattern for the FDA case is much closer to the optimal dropout pattern (refer

Table 2: Simulation output for two example task sets

Scheduler type	Dropout rate for task 1 to task 5 (in percentage)					Signal 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

to Figure 3). Of course, in some case, 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 are increased.

To deal with the challenge due to comparing multiple dimensional data, we conducted simulation for a large number of task sets in order to collect the statistical behavior of the schedulers. We compared the performance of different scheduling algorithms by the scoring approach discussed in Section 4.2. Since obtaining the output signal power is a statistical analysis process, the power value needs to be treated as having an error range. Therefore in applying the scoring approach, we considered p_A to be different from p_B only if $|p_A - p_B| > \rho \min\{p_A, p_B\}$ (where A and B are two different schedulers). In our experiments, we set ρ to be 0.5. If both p_A and p_B are greater than 100, we ignore the comparison result since a system with power great than 100 is considered unstable in the control sense.

Table 3: Comparison of scheduler scores for 246 task sets. Each task set contains one task with the MC constraint.

Dropout Rate Accuracy	EDF to MDA	EDF to DDA	EDF to FDA	MDA to EDF	DDA to EDF	FDA to EDF
0.001	0	0	0	44	40	57
0.01	0	0	0	147	150	83

In Table 3, the comparison results based on a total of 246 task sets are summarized. In comparing the dropout rates for tasks without MC constraints, we considered two cases where the dropout rates were accurate up to the 3nd and 2rd digits after the decimal point. The two cases are given in the second and third row in Table 3, respectively. Six types of comparisons are provided. By "EDF to 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 (as compared to a total order) when comparing two schedulers applied to the same task set. Thus, we need both "EDF to MDA" and"MDA to EDF" to see which gets a higher score. The other columns have the same meaning. From the table, one can readily conclude that our new algorithms outperform the EDF algorithm. If we use less accurate representation (the 3rd row) for the dropout rates, our new algorithms score even better. This reveals that for a number of task sets all four algorithms resulted in very similar dropout rates for the tasks without MC constraints. Table 4 shows similar information as

Table 4: Comparison of scheduler scores for 68 task sets. Each task set contains two tasks with MC constraints.

EDF vs MDA	EDF vs DDA	EDF vs FDA	MDA vs EDF	DDA vs EDF	FDA vs EDF
1	1	1	27	24	19

Table 3 but the task sets used to generate the data contain two tasks with MC constraints (instead of one). The accuracy for the dropout rate is set to two digit after the decimal point. The data again demonstrates that our algorithms outperform EDF. The better performance of MDA, DDA and FDA indicates that these algorithms have reduced the output signal power without penalizing the dropout rates of tasks without MC constraints. This is achieved by making the job dropout processes behave close to the optimal dropout process specifi ed by the MC constraint.

The fact that our proposed algorithms indeed improve the job dropout processes in terms of meeting the MC constraints can also be demonstrated by the plots in Figure 6. Each point in the plots depicts the data obtained by applying one type of scheduler to one task set. The optimal dropout process (i.e., the MC 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 EDF in

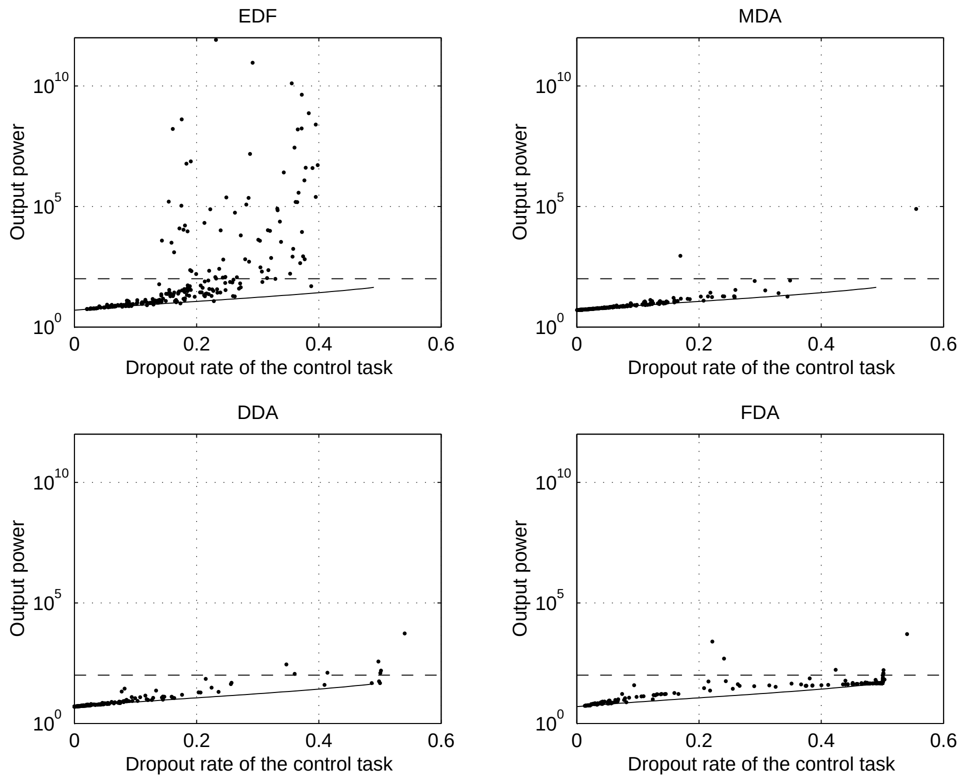


Figure 6: Output signal power v.s. the dropout rate for applying four different schedulers to 246 task sets. The optimal dropout process is also shown.

general is not. This is especially true when the dropout rate is high. A horizontal long-dashed line indicating power being equal to 100 is also shown in the plot. It is clear that EDF resulted 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 regards.

6 Conclusions

In this paper, we consider fi rm 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 (m, k) type of 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.

The scheduling algorithms discussed in the paper are relatively primitive. Improvements can be made by incorporating various admission control schemes to help adjust the jobs in the different groups. We plan to investigate this direction in our future work.

References

[1] T.F. Abdelzaher, K. Shin and N. Bhatti "Performance guarantees for Web server end-systems: a control-theoretical approach," *IEEE Trans. on Parallel & Distributed Sys.*, Vol.13, No.1, 2002, pp. 80-96.

[2] S. Baruah, *el. al.*, "On the competitiveness of on-line real-time task scheduling," *Real-Time Systems*, Vol. 4, 1992, pp. 125-144.

[3] S. Baruah and J. Haritsa, and N. Sharma, "Scheduling for overload in real-time systems," *IEEE Transactions on computers*, Vol. 46, No. 9, 1997, pp. 1034-1039.

[4] G. Bernat, A. Burns and A. Llamosi "Weakly hard real-time systems," *IEEE Trans. on Computers*, Vol. 50, No. 4, 2001, pp. 308-321.

[5] P.R. Blevins and C.V. Ramamoorthy, "Aspects of a dynamically adaptive operating systems," *IEEE Transactions on computers*, Vol. 25, No. 7, pp. 713-725, 1976.

[6] S. Brandt, G. Nutt, T. Berk and J. Mankovich, "A Dynamic quality of service middleware agent for mediating application resource usage," *Real-Time Systems Symposium*, pp. 307-317, 1998.

[7] J.L. Diaz, *et. al.*, "Stochastic analysis of periodic real-time systems," *Real-Time Systems Symposium*, 2002, pp. 289-300.

[8] M. Green and D.J.N Limebeer, *Linear Robust Contro l*, Prentice-Hall, 1995.

[9] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-fi rm deadlines," *IEEE Trans. on Computers*, 1995, 44, pp. 1443-1451.

[10] J.R. Haritsa, M. Livny and M.J. Carey, “Earliest deadline scheduling for real-time database systems,” *Real-Time Systems Symposium*, 1991, pp. 232-242.

[11] A. Kalavade and P. Mogh, “A tool for performance estimation of networked embedded end-systems,” *Proceedings of Design Automation Conference*, 1998, pp. 257-262.

[12] S.Manolache, P.Eles and Z. Peng, “Memory and time-effi cient schedulability analysis of task sets with stochastic execution time,” *13th Euromicro Conf. on Real-Time Systems*, 2001, pp. 19-26.

[13] G. Koren and D. Shasha, “Skip-over: Algorithms and complexity for overloaded systems that allow skips,” *Real-Time Systems Symposium*, 1995, pp. 110-117.

[14] Q. Ling and M.D. Lemmon, “Robust performance of soft real-time networked control systems with data dropouts,” *IEEE Conf. on Decision and Control*, December 2002.

[15] Q. Ling and M.D. Lemmon, “Soft real-time scheduling of networked control systems with dropouts governed by a Markov chain,” *American Control Conference*, June 2003.

[16] M. Matsumoto and T. Nishimura, “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator,” *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, No. 1, 1998, pp. 3-30.

[17] 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, No.1-2, July-Sept. 2002, pp.85-126.

[18] G. Quan and X. Hu, “Enhanced Fixed-Priority Scheduling with (m,k)-Firm Guarantee,” *IEEE Real-Time Systems Symposium*, 2000, pp. 79–88.

[19] P. Ramanathan, “Overload management in real-time control applications using (m,k)-fi rm guarantee,” *IEEE Transactions on Parallel and Distributed Systems*, 1999, 10(6), pp. 549-559.

[20] D.R. Sahoo, S. Swaminathan, R. Al-Omari, M. Salapaka, G. Manimaran and A. Soomani, “Feedback control for real-time scheduling,” *American Control Conf.*, Vol. 2, pp. 1254-9, 2002.

[21] J.A. Stankovic, C. Lu, S.H. Son and T. Gang, “The case for feedback control real-time scheduling,” *Proceedings of 11th Euromicro Conference on Real-Time Systems*, 1999, pp.11-20.

[22] T.-S. Tia, *et. al.* “Probabilistic performance guarantee for real-time tasks with varying computation times”, *Real-Time Technology and Applications Symposium*, 1995, pp. 164-173.