Performance Specifications and Metrics for Adaptive Real-Time Systems*

Chenyang Lu John A. Stankovic Tarek F. Abdelzaher
Gang Tao* Sang H. Son Michael Marley
Department of Computer Science **Department of Electrical Engineering*
University of Virginia, Charlottesville, VA22903
e-mail: {cl7v, stankovic, zaher, gt9s, son, mem5w}@ virginia.edu

Abstract

While early research on real-time computing was concerned with guaranteeing avoidance of undesirable effects such as overload and deadline misses, adaptive real-time systems are designed to handle such effects dynamically. Various research efforts have addressed the characterization and improvement of the dynamic behavior of real-time systems. However, to the authors' knowledge, no unified framework exists for designing adaptive, real-time software systems based on specifications of desired dynamic behavior. We propose such a framework based on control theory. Using control theory a designer can (i) specify the desired behavior in terms of a set of performance metrics that can be mapped to a dynamic response of the control system, (ii) establish an underlying control model of the real-time systems, and (iii) design a resource scheduler using feedback control design methods to guarantee runtime satisfaction of the specs. This is in contrast to more ad hoc techniques. We also show that simply using long term average performance metrics is not sufficient in designing controllers. We then develop a new algorithm based on two PID controllers that meet both the transient and steady state performance requirements.

1. Introduction

While traditional real-time systems such as process control systems typically work in closed and highly predictable environments, a new category of soft real-time applications executing in open and unpredictable environments is rapidly growing in recent years [24]. Examples include online trading, e-commerce, multimedia, and agile manufacturing. Performance guarantees are required in these types of applications. Failure to meet performance guarantees may result in loss of customers, financial damage, or liability violations. Adaptive real-time systems [1-8, 12, 13, 17-19, 20, 23, 26] have been developed as a promising approach to achieve performance guarantees in unpredictable environments. While early research on real-time computing was concerned with guaranteeing complete avoidance of undesirable effects such as overload and deadline misses, adaptive real-time systems are designed to handle such effects dynamically. Dynamic (transient and steady-state) behavior of adaptive real-time systems upon load or resource changes has received special attention in recent years. Transient behavior of an adaptive system describes the responsiveness and efficiency of QoS adaptation in reacting to changes in run-time conditions, and steady-state behavior describes a system's long-term performance after its transient response settles. Several research efforts addressed the characterization and improvement of dynamic behavior of real-time systems. For example, Rosu et. al. [20] proposed a set of performance metrics to capture the transient responsiveness and the steady-sate impact of adaptations. In [8], Brandt, et. al. evaluated a dynamic QoS manager by measuring the transient performance of applications in response to QoS adaptations. However, to the authors' knowledge, no unified framework exists to date for designing an adaptive system from performance specifications of desired dynamic response. In this paper we propose such a system design framework that maps QoS control in adaptive real-time systems to control theory. Our framework works as follows.

1) The system designer specifies the desired dynamic behavior with existing transient and steady-state performance metrics. This step requires a mapping from the existing metrics of adaptive real-time

* Supported in part by NSF grants CCR-9901706 and EIA-9900895, and contract IJRP-9803-6 from the Ministry of Information and Communication of Korea.

systems to the dynamic responses of control systems in control theory.

2) The system designer establishes a mathematical model of the system for the purposes of feedback control.

3) Based on the performance specs and system model from step 1) and 2), the system designer applies existing mathematical techniques in control theory [11][14] to design QoS adaptation algorithms with *analytic* guarantees on the desired transient and steady-state behavior at run-time. This step is similar to the process that a control engineer uses to design a controller for a feedback-control system to achieve desired dynamic responses.

In this paper, we define a category of performance metrics for transient response that can be readily mapped to dynamic response specifications of control systems (section 2), which is the first step in the framework. In the second step, we use the mathematical model we proposed in [17] for adaptive real-time systems. This model allows designing a controller using any of several standard techniques borrowed from control theory textbooks [11]. The second and third steps have been described in previous literature and thus will not be elaborated in this paper. The challenge in applying these steps in practice to computing systems lies in designing a scheduling algorithm that is capable of faithfully implementing the feedback controller. This algorithm is the link between the abstract control-theoretical formulation and the specifics of resource scheduling in a real-time system. In this paper, we show that designing the scheduling algorithm is not trivial. In particular, we show in Section 3 that an existing control-theory based scheduling algorithm, FC-EDF, is in fact incapable of implementing the theoretical controller. Testing this algorithm in the context of the proposed framework reveals that while it achieves the desired steady state response, it cannot produce the desired transient behavior. Instead, it leads to an unstable system.

In section 4, we present a new scheduling algorithm FC-EDF[2] that features two controllers that achieves the performance specs of both the steady-sate and transient behavior. After the discussion of the related work in section 5, the paper concludes with a summary and future work in section 6.

2. Performance specs and metrics

It is difficult to characterize the performance of adaptive real-time systems for two main reasons. First, it is often impossible to predict the exact run-time workload of such systems. Therefore, the system developer needs to characterize *how much* and *in what form* the system should be stressed. Second, traditional metrics such as the average miss-ratio cannot capture the transient behavior of the system in response to load variations. Recently (e.g., [20]), a set of metrics was proposed to characterize both transient and steady-state behavior of an adaptive system. In this section, we extend these metrics and show that the set of metrics can be mapped to dynamic responses of control systems, which enable the use of control theory techniques in QoS adaptation designs. The performance specs consist of a set of *miss-ratio profiles*[1], in response to a set of representative *load profile*s adapted from control theory [11].

2.1. Load Profile

For real-time systems operating in unpredictable environments, system load is not known *a priori*. However, system performance can be specified under a set of representative load profiles borrowed from control theory; namely, the *step load* and the *ramp load*. Similar types of signals have been widely used to generate canonical system responses in control theory. In the context of real-time systems, the step load represents the worst-case load variation, and the ramp load represents a nominal form of load variation. We define the system *load* as the resource requirement in percentage of the system capacity (i.e., the load corresponding to the full system capacity is 100%, and an overload is a system load that is higher than 100%). A *load profile* $L(t)$ is the system load as a function of time. In practice, a certain level of load can be translated into system-specific load parameters. For example, a 500% system load can be translated to the request rate of 8,000 request/sec in a specific web server (assuming a fixed requested file type/size distribution). For a process control system whose workload is composed of periodic tasks, the load can be expressed as the requested CPU utilization. The load profiles are defined as follows.

- *Step-load* $SL(t)$: a load profile that instantaneously jumps from a nominal load L_{nom} to load L_{max} and stays constant after the jump. In real systems, load variations typically occur gradually over a finite amount of time. Gradual load changes are easier to control and adapt to than sudden load changes. The step-load is represented with a tuple $SL(L_{nom}, L_{max})$.

- *Ramp-load* $RL(t)$: a load profile that increases linearly from the nominal load to a specific level

1 The miss-ratio profile can be generalized to other metrics such as resource utilization, response time, throughput, and value-cognizant metrics.

of overload during a time interval. Compared with the step load, the ramp signal represents a less severe and more realistic load variation scenario. The ramp-load *RL(t)* is described with a tuple $RL(L_{nom}, L_{max}, T)$, where L_{nom} is the original load, L_{max} is the new load, and T is the time it takes the load to increase from L_{nom} to L_{max}.

In practice the load profiles are application-specific based on the load characteristics and system requirements. It is usually necessary to specify and test the system performance under a series of load profiles with different types/parameters. The load profile is an *abstraction* of the workload, and there can be many possible instantiations of the same load profile. The instantiation of a load profile should incorporate the knowledge of the workload, and therefore the load profile should be viewed as an enhancement to existing benchmarks.

2.2. Miss-Ratio Profile

We now characterize the system performance in terms of deadline miss-ratio in response to a specific load profile. The *miss-ratio function MR(t)* is defined as the number of deadline misses divided by the number of task submissions in a time window *(t-MW, t)*, where *MW* is an application specific parameter called the *miss-ratio window*. Note that when *MW* is small, *MR(t)* approximates the instantaneous system performance at time *t*. In contrast, the *average miss-ratio* M_a, a traditional metric for real-time systems, is defined as the total number of deadline misses divided by the total number of task instances throughout the run-time (or in a much larger time window than *MW*). M_a represents the average system performance over a long time period. The average miss-ratio alone can be an inadequate metric in characterizing the dynamics of the system performance (as demonstrated in sections 3 and 4). Based on the miss-ratio function, the system performance can be characterized with the miss-ratio profile, a set of characteristics of *MR(t)* in response to a load profile *L(t)*. From the control theory point of view, a real-time system transits from the *steady state* to the *transient state* when load variation causes *MR(t)* to deviate significantly from its current value. After a time interval in the transient state, the system may settle down to a new steady state. For real-time systems, the steady-state can be defined as a state when *MR(t)* is bounded by a constant called *steady-state miss-ratio bound SMB*, i.e., $MR(t) \leq SMB$. *SMB* depends on the tolerance of the application to the deadline miss-ratio (e.g., *SMB*=0 for a system which requires no deadline misses in steady state). The transient state represents a state with degraded performance upon overload, while the steady state represents a state when the system performance is satisfactory. A miss-ratio profile describes the system performance in both transient state and steady state as follows.

- *Stability*: A system is said to be *stable*, in response to a step or ramp load profile, if the system output converges to zero as time goes to infinity, i.e., the system is said to be stable if $\lim_{t\to\infty} MR(t) = 0$ under any step or ramp loads[2]. P*ractical stability* is defined as the system's ability to return to a given small miss-ratio after the system experiences a step load or a ramp load profile, i.e., the system is said to be practically stable if $\lim_{t\to\infty} MR(t) \leq SMB$ for some small constant $0 < SMB < 1$. Note that the existence of the above limit excludes excessive oscillatory behavior. We should also note that, by definition, the practical stability of a system depends on the choice of the settling miss-ratio bound *SMB*. A system may be practically stable under one *SMB* value, but may not be practically stable under a different (smaller) *SMB* value. A stable system can always recover from the specified overload conditions and resume satisfactory performance; while an unstable system may fail to recover from an overload. Therefore, stability is an important requirement for real-time systems. For convenience of presentation, we will use the term stability to refer to practical stability in the rest of this paper.

- *Transient-state response* describes the responsiveness and efficiency of QoS adaptation in reacting to changes in run-time conditions.
  - *Overshoot* M_o: The highest miss-ratio in the transient state. Overshoot represents the worst-case transient performance of a system in response to the load profile. Overshoot is an important metric because a high transient miss-ratio can cause system failure in many systems such as robots and media streaming [8].
  - S*ettling time* T_s: The time it takes the system miss-ratio to enter a steady state after a load profile occurs. The settling time represents how fast the system can recover from overload. This metric has also been called reaction time or recovery time [20].

[2] Usual stability concepts from control theory such as Lyapunov stability and bounded-input and bounded-output stability [9], may not be applicable to real-time systems, because either the system states are not defined or miss-ratio is always bounded in [0, 1].

- *Steady-state miss-ratio* M_s: Average miss-ratio in the steady state. M_s characterizes how well the system recovers after adaptation.

- *Sensitivity* S_p: Relative change of the steady-state miss-ratio with respect to the relative change of a system parameter *p*. For example, sensitivity with respect to the task execution time S_{et} represents how significantly the change in the task execution time affects the system miss-ratio. Sensitivity describes the robustness of the system with regard to workload or system variation.

With the load profile and miss-ratio profile, we establish a mapping from metrics of adaptive real-time systems to dynamic response of control systems. This mapping enables system designers to apply established control theory techniques to achieve stability, and meet transient and steady-state specs.

	SL(0, 200%)	**RL(100%, 400%, 60 sec)**
M_o	< 50%	< 50%
T_s	< 60sec	< 90sec
M_s	= 0%	= 0%
S_{et}	= 0%	= 0%
MW	2.4sec	
SMB	0%	

Table 1. Specs of a real-time system

2.3. Specs of a real-time system

As an example of the control-based design framework, we first define the transient and steady-state specs of a real-time system. We then show that an existing scheduling algorithm FC-EDF cannot satisfy the specs due to problems with its controller design. In section 4, we present a new scheduling algorithm with a improved controller that can satisfy the specs.

Table 1 illustrates the specs for a real-time system to be designed. The miss-ratio window of interest is *MW* = 2.4 sec. *SMB* = 0 means that *MR(t)* stays at 0% the system is in steady state. The transient and steady-state performance require that, (1) The system should always recover from a step load of 200% of the system capacity (SL(0, 200%)) and from a ramp load that grows from 100% to 400% of the system capacity within a minute (RL(100%, 400%, 60 sec)); (2) (Settling-time requirement) The system should resume zero deadline miss-ratio within 60 sec after the occurrence of the step load and within 90 sec after the occurrence of the ramp load; (3) (Overshoot requirement) The highest miss-ratio during the transient state should be lower than 50%; (4) (Steady-state miss-ratio requirement) The system should maintain zero miss-ratio after it settles down from overload; (5) (Sensitivity requirement) The system should maintain 0% miss-ratio in steady state regardless of the actual execution times of the tasks. With a control-theoretical system model such as that described in [17], one can design a controller that achieves the above specs on transient and steady state response. A standard controller used in industry today is the PID controller. The PID controller design problem is elaborated in standard control textbooks such as [11], and is therefore not described in this paper. Once the PID controller is designed, the challenge is to translate it into a feedback scheduler. The only scheduler known to the authors, designed to implement a PID controller, is FC-EDF. In this paper we show the inadequacy of FC-EDF and propose an improved scheduler design that allows satisfying both transient and steady state response metrics such as those presented above. With the new scheduler in place, a wealth of existing control-theoretical design techniques can be leveraged to build dynamic real-time systems with guaranteed transient and steady-state response.

3. FC-EDF revisited

In this section, we study the performance of the FC-EDF [17] in the above proposed framework. It was shown that FC-EDF rendered satisfactory performance in term of the classical (steady state) metrics such as the average deadline miss-ratio (M_a) [17]. However, a FC-EDF may fail to meet the specs in Table 1, due to control saturation or system modeling errors, as shown by experiment results (see Section 3.3). This motivates us to modify FC-EDF so that better system performance could be achieved in the proposed metrics (see Section 4).

3.1. Overview of the FC-EDF

FC-EDF is a scheduling algorithm that integrates PID (proportional-integral-derivative) feedback control with an EDF scheduler [5]. The design and evaluation based on traditional metrics were presented in [17]. The FC-EDF scheduler (Figure 1) is composed of a PID controller, a Service Level Controller (SLC), an Admission Controller (AC) and a basic EDF scheduler. The EDF scheduler schedules the accepted tasks according to the EDF policy. FC-EDF features a feedback control loop as follows.

- The system deadline miss-ratio *MR(t)* is periodically monitored (with a sampling period *SP*) and fed back to the PID controller. Note that monitored *MR(t)* is defined in the time window *(t-SP, t)*, where SP can be different from the miss-ratio window *MW* in the specs (Table 1).
- The PID controller compares the current *MR(t)* with the miss-ratio set point MR_s to get an error *E(t)*, and maps it to the required change in the total

requested CPU utilization $\Delta U(t)$ according to the PID control function,

$$\Delta U(t) = KP \bullet (E(t) + KI \bullet \Sigma_{IW} E(\tau) + KD \bullet (E(t)-E(t-DW))/DW)$$

where KI, KP, and KD are controller parameters, and DW is the size of the derivative time window.

- SLC and AC change the total requested CPU utilization of the system by estimated $\Delta U(t)$ based on the *estimated* task execution times. In particular, SLC changes the QoS levels of admitted tasks to adjust the load in the system, and AC adapts admission/rejection decisions to affect the load flowing into the system.

Controller design was based on a mathematical model presented in [17].

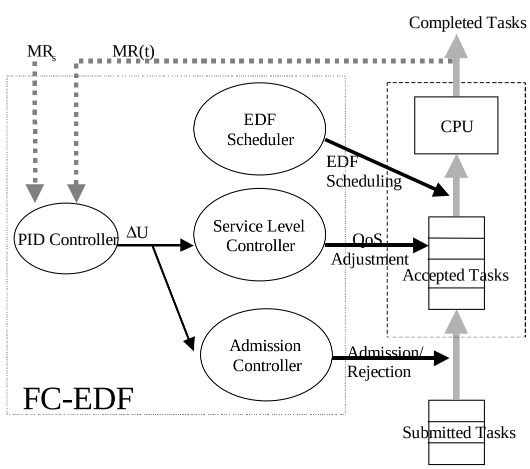

Figure 1 FC-EDF Scheduling Algorithm

3.2. Testing Configurations

The workload generator generates a random set of periodic tasks to a real-time system simulator [17] according to the specified load profiles. Each periodic task is characterized by a period P (the deadline of each task instance equals its period), a starting time, a finishing time, a set of worst-case execution times $\{WCET_i \mid i = 0,1\}$, a set of best-case execution times $\{BCET_i \mid i = 0,1\}$, and a set of estimated execution times $\{EET_i \mid i = 0,1\}$. $WCET_i$, $BCET_i$, and EET_i are the worst-case, best-case, and estimated execution times of QoS level i, respectively. Only the *estimated* execution time of each task is known to the scheduler. The *actual* execution time of each task is unknown and can deviate significantly from the estimation. Tasks' periods are harmonious with the least common multiple LCM_p=2400 time-units. Regarding each time unit as equivalent to one millisecond, we have LCM_p=2.4 sec. For the step-load SL(0, 200%), tasks arrive simultaneously at the beginning of each experiment, and each task's actual execution time equals the worst-case execution time (unknown to the scheduler). The total system load (requested CPU utilization) is close to 200%. The ramp-load RL(100%, 400%, 60 sec) is instantiated with a set of tasks (with a total load close to 100%) arrives at time 0, and a new task arrives every 0.64 sec afterwards until the system load reaches 400% by 60 sec.

Set Point	1%
SP	2.4 sec
KP	0.25
KI	0.1
KD	0

Table 2. FC-EDF configuration

During each run in the experiment, a miss-ratio analyzer (note the miss-ratio analyzer is part of the testing environment and different from the monitor of the FC–EDF) measured the miss-ratio $MR(t)$ and derived the miss-ratio profile. The sampling period of the analyzer is MW=2.4 sec as required by the specs. A sampled trace of the CPU utilization $U(t)$ is also measured at the same frequency in the experiments for reference. AC was turned off in the experiments and SLC is the only actuator in the experiments. The configuration of the tested FC-EDF is listed in Table 2. Note that M_s = 1% ≠ 0 to avoid CPU underutilization as explained in [17] and later in this section. Load SL(0, 200%) was used to stress the system. The experiment consisted of 20 repeated runs and each run lasted 20 minutes.

3.3. Instability of FC-EDF

In the experiments, FC-EDF performed well on average miss-ratio, which equals (with 90% confidence interval) 0.9802%(±0.0128%). However, experiments also demonstrated that FC-EDF was *unstable* and therefore cannot satisfy the specs. For example, in a typical case, plotted in Figure 2, the system entered a transient state and $MR(t)$ overshot to 50.8% in response to the step load. FC-EDF then decreased task QoS levels to reduce the system load. However, although $MR(t)$ = 0% most of the time after 16.8 sec, the system also suffered from cyclic deadline misses with $MR(t)$=4.5%. In control theory, this behavior is called a *limit cycle*. It is caused by saturation of the controller. The miss-ratio feedback control loop suffers a *control saturation* problem because the feedback is geared to measure overload only, but not underutilization. When the system is overloaded, the measured $MR(t)$ is representative of the degree of overload. However, it is insensitive to the degree of underutilization when the system is underutilized ($MR(t)$ becomes identically 0%). Thus, systems with 70% utilization and 5%

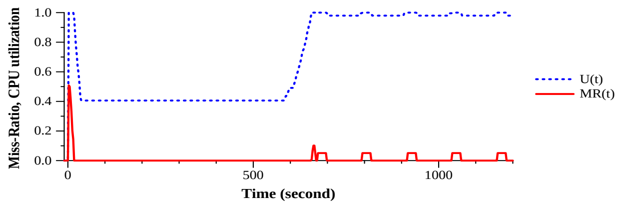

Figure 2. Miss-Ratio MR(t) and Utilization U(t) of FC-EDF (Miss-Ratio Control; Set Point: 1%; (KP, KI)=(0.25, 0.1))

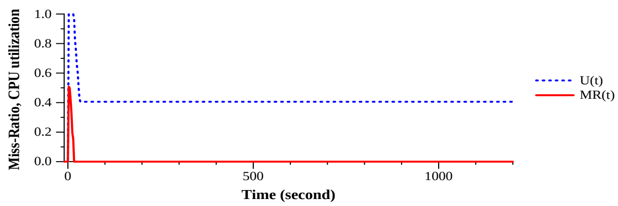

Figure 3. Miss-Ratio MR(t) and Utilization U(t) of FC-EDF (Miss-Ratio Control; Set Point: 0%; (KP, KI)=(0.25, 0.1))

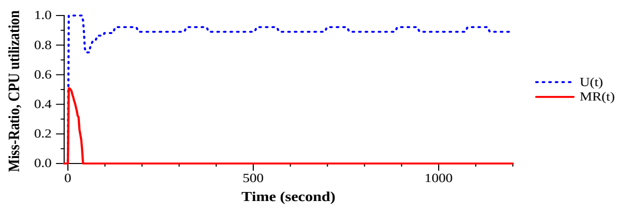

Figure 4. Miss-Ratio MR(t) and Utilization U(t) of FC-EDF (Utilization Control; Set Point: 90%; (KP, KI)=(0.25, 0.1))

utilization will appear the same to the controller. The control saturation problem prevents a 0% set point because it can underutilize CPU and reduce system throughput [17]. When the system has a 0% miss ratio, but extremely low utilization, a feedback control scheduler with a zero set point will treat it as the correct state (as illustrated in Figure 3). Small set point values cause the controller to oscillate between underutilized and overloaded system states.

It is therefore desired to apply a feedback control of the CPU utilization in addition to the miss-ratio. In the utilization control scheme, the system monitors the CPU utilization and dynamically adjusts the task QoS levels. With a set point lower than the utilization bound of the system, the system can achieve a 0% steady state miss-ratio. Utilization control has been applied successfully to computing systems such as web server QoS [11] and multimedia player [8]. Interestingly, utilization control suffers from the complementary control saturation problem. It saturates when the system becomes overloaded as the CPU utilization becomes identically 100%. A consequence of the utilization-control saturation can be a longer settling time when the server is heavily overloaded. For example, the utilization control illustrated in Figure 4 has a longer settling time than the miss-ratio control with the same control parameters as in Figures 2 and 3. Another problem with utilization control is the difficulty in computing the schedulable utilization bound. Although existing real-time scheduling theory (such as periodic rate-monotonic/EDF analysis [16], and aperiodic analysis [6]) has derived utilization bounds under different workload assumptions, it is still difficult to decide the utilization bound in many complex systems with unpredictable workload. Below, we propose a scheme that combines the advantages of both miss ratio control and utilization control while avoiding their limitations.

4. FC-EDF^2 scheduling algorithm

Because a FC-EDF may not satisfy the desired specs due to implementation constraint (control saturation) or system modeling errors, we now propose a new scheduling algorithm FC-EDF2 which is able to meet desired specs such as stability and transient performance.

4.1. The FC-EDF^2

In this section, we present the design of FC-EDF^2 (Figure 5), which can achieve stability and the desired transient and steady-state specs. FC-EDF^2 achieved stability by integrating miss-ratio control and utilization control. The intuition behind this combination lies in noting that the saturation conditions of the two control loops are mutually exclusive. Since control saturation is the main reason for instability of FC-EDF, the hybrid control scheme is expected to fix this problem. In this hybrid control scheme, both the miss-ratio $MR(t)$ and the CPU utilization $U(t)$ are monitored. At each sampling point, $MR(t)$ and $U(t)$ are fed back to two separate PID controllers, the miss-ratio controller (with a set-point MR_s) and the utilization controller (with a set point U_s), respectively. Each controller then computes its control signal independently. The control signal of the utilization control $\Delta U_u(t)$ is compared with the miss-ratio control signal $\Delta U_m(t)$, and the smaller control

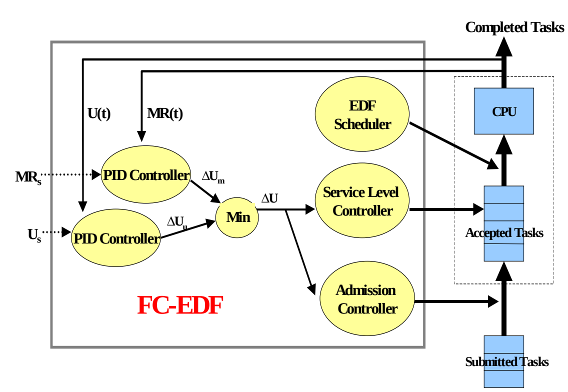

Figure 5. FC-EDF^2 scheduling algorithm

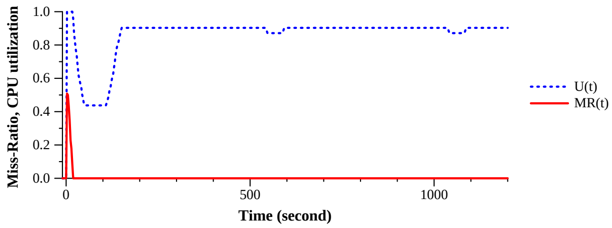

Figure 6. Miss-Ratio MR(t) and Utilization U(t) of FC-EDF2 (Hybrid Control; (KP, KI)=(0.25, 0.1))

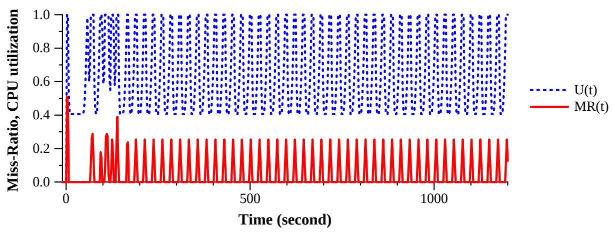

Figure 7. Miss-Ratio MR(t) and Utilization U(t) of FC-EDF2 (Hybrid control; (KP, KI)=(1.0, 0.9))

signal $\Delta U(t)=min(\Delta U_u(t), \Delta U_m(t))$ is sent to actuators (AC and SLC). The rationale of the hybrid scheme is to combine the advantages of the two controls.

First, FC-EDF^2 can achieve stability by canceling control saturation via dynamically switching controllers. When the system enters the saturation range of utilization control ($U(t)$>100%), the miss-ratio control will dominate and thus achieve faster control (i.e., a shorter settling time). On the other hand, when the system enters the output saturation range of the miss-ratio control ($U(t) < UB$), the utilization control will take over and achieve zero deadline miss-ratio. Since $UB \leq 100\%$, there is no overlap between the saturation ranges of the utilization control and the miss-ratio control. Consequently, the hybrid control always keeps the control alive. In addition, we used two techniques, the *min* operator and *integrator anti-windup* technique [11] to avoid oscillation during mode switching between the two controls. The *min* operator on the two control signals achieved smooth transition between the two controls. The integrator anti-windup requires a (miss-ratio or utilization) controller to turn off the integration of errors once it loses in the min operation (i.e., its signal is larger than the other control signal).

The second advantage of the hybrid control (with the min operator) is that it does not require an *accurate* utilization bound to guarantee satisfactory performance. Suppose the system utilization bound $UB(t)$ is approximate or varies at run time, the utilization control is effective when the actual utilization bound $UB(t) \geq U_s$. However, when the $UB(t)<U_s$, the miss-ratio control dominates (due to the large and/or reoccurred deadline misses) and bounds the performance of the scheduler.

Following our control-theory-based framework, our next step is to establish an analytical model for the control system. For example, preliminary results on the modeling of FC-EDF and a web server have been presented in [17] and [4], respectively. Based on the performance specs and a system model, we can use existing control design methods [11][14] to calculate the values of the control parameters that can satisfy the specs based on the new metrics. The design process followed standard control design method [11] and its full presentation is out of scope of this paper. Instead we present the experimental results that demonstrate that FC-EDF^2 can be tuned to satisfy the specs.

4.2. Control Gains

Experiments were run for different combinations of control gains (*KP*, *KI*). Gain ranges were obtained first by tuning a controller theoretically [11] using the system model described in [17] then by choosing a range around the computed gain value to demonstrate performance trends as the gain changes. The derivative control gain *KD*=0 in all the experiments because it is not suitable for systems with high noise. MR_s = 5%, and U_s = 90%. Both the step-load SL(0, 200%) and ramp-load RL(100%, 400%, 60 sec) were used in these experiments; 20 runs were made for each combinations of (*KP*, *KI*, load-profile). Each run lasted for 20 min. A fixed sampling period *SP*=2.4 sec is used in all the runs of experiments of this section. All the presented data (except for stability) is the average value (with 90% confidence interval) of 20 runs. Note that steady-state miss-ratio is relevant only when *SMB* > 0. When *SMB*=0% (as in our specs), the steady-state miss-ratio M_s is always 0% when the system is stable; and M_s is undefined when the system is unstable. Sensitivity was not measured in the experiments and will be part of our future work.

4.2.1. Stability

As defined in section 2.2, a real-time system is stable if it can always settle down to a steady state in finite time. According to the specs of Table 1, the system is in steady-state if $MR(t)$=0%. A system unstable if there exists a run in which the system fails to enter and stay in the steady state. For example, an unstable case of FC-EDF^2 (with (KP, KI) = (1.0, 0.9)) is illustrated in Figure 7. The system never settled to a steady state with $MR(t)$=0% after a step load SL(0, 200%) arrived. The system performance oscillated at a high frequency, which was an indication of frequent changes in task QoS levels. QoS oscillation at high frequency can be undesirable. For example, an unstable video player may frequently change its frame size and never settle down to a fixed frame size and desired frame rate. In contrast, Figure 6 illustrates that FC-EDF^2 achieved stability in response to the same load when (KP, KI) = (0.25, 0.1). The system settled to 0% miss-ratio by 19.2 sec when the miss-ratio control dominated the control in the beginning, and the system maintained a 0% steady-state miss-ratio afterwards because the utilization control took over. Table 3 shows the stability region of control gains of FC-EDF^2. The system became unstable when large control gains caused the controller to overreact to performance variations. All the unstable cases occurred in response to the step load SL(0, 200%), which indicated that step loads is more difficult to handle than ramp loads.

(KI,KP)	**0.025**	**0.05**	**0.1**	**0.25**	**0.5**	**0.75**	**1.0**
0.1	S	S	S	S	S	U	U
0.5	S	S	S	S	U	U	U
0.9	S	S	S	U	U	U	U

Table 3. Stability of FC-EDF^2 (S: stable, U: unstable)

4.2.2. Settling-Time (T_s)

The settling time describes the agility of the system in response to overload. The settling times of FC-EDF^2 in response to the step-load SL(0, 200%) and ramp-load RL(0, 400%, 60 sec) are illustrated in Figures 8 and 9, respectively. When the control gains were small, the settling-time decreased as the control gains increased. This is because the controllers of FC-EDF^2 can change the system load by a higher magnitude for each sampling period if their control gains are higher. However, as the control gains became large, the settling time started to increase due to excessive oscillations. The settling time eventually increased to infinity when the control gains hit the stability boundary (Table 3) and the system became unstable. The control gains had a significant impact on the settling time. For example, when (*KP*, *KI*)=(0.025, 0.5), the settling time in response to load RL(100%, 400%, 60 sec) is 133.5600(±7.6826) sec, while the settling time in response to the same load is only 28.8000(±0.3011) sec when (*KP*, *KI*)=(0.25, 0.5). Note although the (*KP*, *KI*)=(0.75, 0.5) has a even shorter settling time of 10.4400(±2.1973) sec in response of RL(100%, 400%, 60 sec), the control gains caused instability in response to SL(0, 200%). Similarly, the maximum settling time in response to load SL(0, 200%) is 603.6000(±85.3359) sec (when (*KP*, *KI*)=(0.025, 0.9)), which is more than 50 times of the minimum settling time of 12.0000(±0.0000) sec in response of the same load (when (*KP*, *KI*)=(0.5, 0.1)).

4.2.3. Overshoot (M_o)

The overshoot in response to RL(100%, 400%, 60 sec) and SL(0, 200%) is illustrated in Figures 10(a) and 10(b), respectively. In the ramp-load case, the system overshoot decreased as the controller gains increased. This is because the scheduler with small control gains adapted the system load slowly and consequently allowed the miss-ratio to increase significantly. However, when the control gains became too high, overshoot started to increase due to excessive oscillation. The control gains significantly affected the

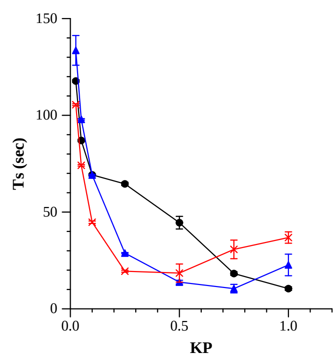

Figure 8 Settling Time vs. Control Gains
Load: RL(100%, 400%, 60sec)

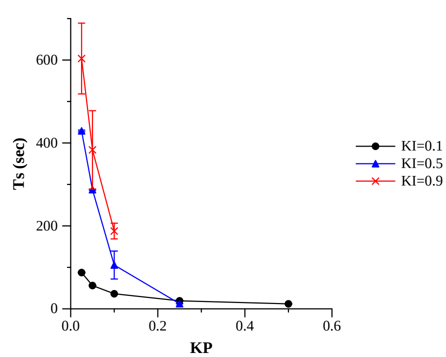

Figure 9 Settling Time vs. Control Gains
Load: SL(0, 200%)

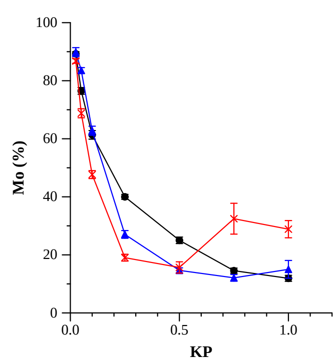

Figure 10(a) Overshoot vs. Control Gains
Load: RL(100%, 400%, 60sec)

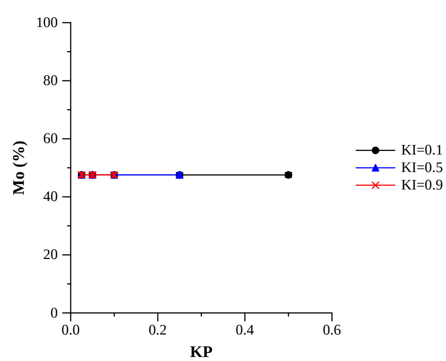

Figure 10(b) Overshoot vs. Control Gains
Load: SL(0, 200%)

system overshoot. When the control gains were (0.025, 0.5), the system overshot to 89.8583(±1.5399)% in response to load RL(100%, 400%, 60sec), i.e., most of task instances missed their deadlines during the worst sampling period. In contrast, in response to the same load, only 25.0147(±1.1180)% of the task instances missed their deadlines in the worst sampling period when (*KP*, *KI*)=(0.5, 0.1). Different from the ramp-load case, Figure 10b showed that the control gains had no impact on the overshoot for the step-load. It was 47.5395(±0.8110)% for all control gains. This is because the step-load jumped instantaneously and gave the system no time to react before overshoot occurred.

4.2.4. Average miss-ratio (M_a)

Although average miss-ratio is not part of the performance specs, they are also presented in Figures 11a and 11b for reference. The average miss-ratio is defined as the number of deadline misses divided by total number of task instances in a run. The average miss-ratio demonstrated much smaller differences when control gains varied, which indicates that average miss-ratio failed to capture the significant difference in stability and transient response of different control gains. For example, although the system response was extremely slow (with a settling time of 603.6(±85.3359) sec) when (*KP*, *KI*)=(0.025, 0.9) in response to SL(0, 200%); its average miss-ratio was only 1.5231(±0.0928)%. The miss-ratio of the system overshot to 89.8583(±1.5399)% in response to RL(100%, 400%, 60sec) when (*KP*, *KI*)=(0.025, 0.5), its average miss-ratio was only 5.9901(±0.4724)%.

4.2.5. Summary: control gains

In this section, we have verified that three set of control gains, (0.25, 0.1), (0.5, 0.1), and (0.25, 0.5), can satisfy the specs in Table 1 when SP=2.4 sec. Among these settings, (0.25, 0.1) is farthest away from the unstable region of the control gains (Table 3) and therefore are the most desirable values.

4.3. Sampling Period

In this section, we present experiments on different sampling periods (SP). *(KP, KI)* were fixed at the optimal control gains (0.25, 0.1) in all the experiments. Due to space limitations of this paper, we only summarize the results in the following.

- **Stability**: The system is stable when 2.4 ≤ SP ≤ 7.2 (sec). Note that the system was unstable when SP = 0.6 or 1.2 sec. This result is unusual because control system performance usually improves as *SP* decreases according to digital control theory [14]. This anomaly is due to the periodicity of the simulated real-time system. Because all the task periods are harmonious with LCM_p=2.4 sec, the system completes a natural cycle at each time-point divisible by LCM_p (i.e., same number of tasks enter the system every LCM_p). However, when $SP<LCM_p$, the controller in FC-EDF[2] monitors different number of tasks in different sampling periods, and the measured miss-ratio becomes jittery due to the different number of tasks in different sampling periods. Note that this effect may affect performance for periodic tasks periods with non-harmonious periods or aperiodic tasks. These types of workloads will be investigated in our future research.

- **Settling-time (T_s)**: Among the stable sampling periods, the settling time in response to both loads increased significantly as *SP* increased. This is because the controllers with smaller sampling period monitored and responded to load variations at a higher rate and thus settled down to zero miss-ratio faster.

- **Overshoot (M_o)**: When the system was stable, the system achieved significantly lower overshoot in response of the ramp-load as *SP* decreased. This is also because the controllers adapted faster with a smaller sampling period. M_o keeps constant in response to the step-load for all tested stable sampling periods. This is also due to the periodicity of the task arrivals. Since $SP>LCM_p$ for all the stable sampling periods, none of the sampling period was able to prevent overshoot that

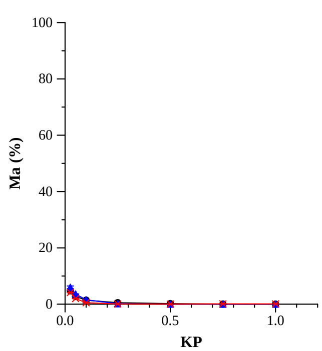

Figure 11(a) Average Miss-Ratio vs. Control Gains
Load: RL(100%, 400%, 60sec)

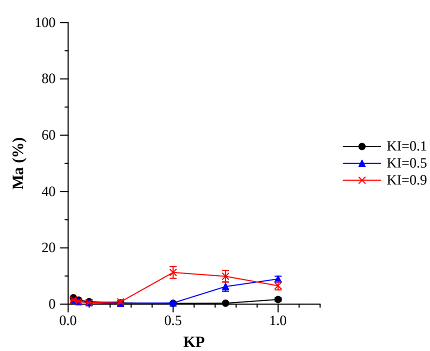

Figure 11(b) Average Miss-Ratio vs. Control Gains
Load: SL(0, 200%)

occurred before the first sampling point.

Among all the tested sampling period values, only *SP*=2.4 sec can satisfy the performance specs in Table 1 when (*KP*, *KI*) = (0.25, 0.1).

4.4. FC-EDF^2 summary

In section 4, we presented a scheduling algorithm FC-EDF^2 that fixed the instability problem of FC-EDF with a hybrid controller. Experimental results verified that the new scheduler can achieve the specs in terms of stability, transient and steady-state performance with correct parameters. To remain focused on scheduler design and performance evaluation, we omitted details regarding system modeling and controller tuning from the discussion presented in this section. For completeness, modeling of a real-time system as a controlled process is described in [17]. Several controller design techniques can be found in [11].

5. Related Work

Transient and steady-state performance of adaptive real-time systems have received special attention in recent years. For example, Brandt et. al. [8] evaluated a dynamic QoS manager by measuring the transient performance of applications in response to QoS adaptations. Dynbench [27] included *observed real-time QoS metrics* based on a time series of measurement, which is similar to the miss-ratio function *MR(t)* presented in this paper. Rosu et. al. [20] proposed a set of performance metrics to capture the transient responsiveness of adaptations and its impact on applications. The paper proposed metrics that are similar to the settling time and steady-state error.

The general framework of stability, transient response and steady-state performance in control theory [11] has been mostly applied in mechanical and electrical systems. There are several works that applied control theory to computing systems. For example, several works [17][21][22] proposed to integrate control system design with real-time scheduling. In [25], a feedback-based scheduling scheme was used to adjust CPU allocation based on application-dependent progress monitors. [1][23][26] utilized flexible timing constraints for performance adaptation in real-time control systems. [3][9][12][15] presented QoS control architectures for multimedia and communication systems. In [2], a PI controller was applied in resource allocation to achieve web server QoS. However, to the authors' knowledge, no unified framework has been proposed that integrates the specs of dynamic responses with the design of QoS control for a computing system. Such a framework is presented in this paper. A scheduling algorithm is proposed to map theoretical controller design to a scheduler implementation.

6. Conclusions and Future Work

In this paper, we propose a control-theory-based framework. This framework enable system designers to specify the performance specs (stability, transient and steady-state performance), and apply existing control methods to analytically design an adaptive real-time system to achieve the specs. We extended existing performance metrics of adaptive real-time systems [20] and demonstrate that the metrics can be mapped to the dynamic responses of a control system. Such mapping provides a foundation for further control design of QoS adaptation in real-time systems. The paper also identifies stability as a critical requirement for adaptive real-time systems. The framework is applied to the design of an adaptive real-time scheduling algorithm to satisfy a set of performance specs. An existing scheduling algorithm FC-EDF was discovered to be unstable due to problems with its control design. We presented a new scheduling algorithm FC-EDF^2 that achieved stability with a hybrid controller that achieved stability. Experiments verified that FC-EDF^2 could achieve the performance specs. In the future, we will apply the framework to more complex systems such an adaptive real-time web server.

Acknowledgements

The authors would like to thank their shepard Daniela Rosu and anonymous reviewers for their valuable suggestions to improve the paper.

7. Reference

[1] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automatic flight control," *IEEE Real-Time Technology and Applications Symposium*, June 1997.

[2] T. F. Abdelzaher and N. Bhatti, "Adaptive content delivery for web Server QoS," *International Workshop on Quality of Service*, June 1999.

[3] T. F. Abdelzaher and K. G. Shin, "End-host Architecture for QoS-Adaptive Communication," *IEEE Real-Time Technology and Applications Symposium*, Denver, Colorado, June 1998.

[4] T. F. Abdelzaher and C. Lu, "Modeling and Performance Control of Internet Servers," *39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

[5] T. F. Abdelzaher, "An Automoated Profiling Subsystem for QoS-Aware Services," *IEEE Real-Time Technology and Applications Symposium*, Washington D.C., June 2000.

[6] T. F. Abdelzaher, "A Schedulable Utilization Bound for Aperiodic Tasks," *University of Virginia, Technical Report CS-2000-21*, August 2000.

[7] G. Beccari, et. al., "Rate Modulation of Soft Real-Time Tasks in Autonomous Robot Control Systems," *EuroMicro Conference on Real-Time Systems*, June 1999.

[8] S. Brandt and G. Nutt, “A dynamic quality of service middleware agent for mediating application resource usage,” *19th IEEE Real-Time Systems Symposium*, December 1998.
[9] S. Cen, "A Software Feedback Toolkit and its Application In Adaptive Multimedia Systems," *Ph.D. Thesis,* Oregon Graduate Institute, October 1997.
[10] J. Eker: "Flexible Embedded Control Systems-Design and Implementation." PhD-thesis, Lund Institute of Technology, December 1999.
[11] G. F. Franklin, J. D. Powell and A. Emami-Naeini, *Feedback Control of Dynamic Systems (3rd Ed.)*, Addison-Wesley, 1994.
[12] D. Hull, A. Shankar, K. Nahrstedt, and J. W. S. Liu, “ An end-to-end QoS model and management architecture,” *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*, December 1997.
[13] M. Humphrey, S. Brandt, G. Nutt, and T. Berk, “The DQM architecture: middleware for application-centered QoS management,” *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services,* December 1997.
[14] B. C. Kuo, Digital Control Systems (2nd Ed.), Oxford University Press, 1992.
[15] B. Li, D. Xu, K. Nahrstedt, J. W. S. Liu, “End-to-End QoS Support for Adaptive Applications Over the Internet”, *SPIE International Symposium on Voice, Video and Data Communications*, November 1998.
[16] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,” *JACM*, 20(1), 1973.
[17] C. Lu, J. A. Stankovic, G. Tao and S. H. Son, “Design and Evaluation of a Feedback Control EDF Scheduling Algorithm,” *20th IEEE Real-Time Systems Symposium*, December 1999.
[18] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, “Practical solutions for QoS-based resource allocation problems,” *19th IEEE Real-Time Systems Symposium*, December 1998.
[19] D. Rosu, K. Schwan, and S. Yalamanchili, “FARA–a framework for adaptive resource allocation in complex real-time systems,” *IEEE Real-Time Technology and Applications Symposium*, June 1998.
[20] D. Rosu, K. Schwan, S. Yalamanchili and R. Jha, "On Adaptive Resource Allocation for Complex Real-Time Applications," *18th IEEE Real-Time Systems Symposium,* Dec., 1997.
[21] M. Ryu and S. Hong, “Toward Automatic Synthesis of Schedulable Real-Time Controllers”, *Integrated Computer-Aided Engineering*, 5(3) 261-277, 1998.
[22] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, “On Task Schedulability in Real-Time Control Systems,” *17th IEEE Real-Time Systems Symposium*, December 1996.
[23] K. G. Shin and C. L. Meissner, “Adaptation and Graceful Degradation of Control System Performance by Task Reallocation and Period Adjustment,” *EuroMicro Conference on Real-Time Systems*, June 1999.
[24] J. A. Stankovic, C. Lu, S. H. Son, and G. Tao, “The Case for Feedback Control Real-Time Scheduling,” *EuroMicro Conference on Real-Time Systems*, June 1999.
[25] D. C. Steere, et. al., "A Feedback-driven Proportion Allocator for Real-Rate Scheduling," *3rd Symposium on Operating Systems Design and Implementation*, Feb 1999.
[26] L. R. Welch, B. Shirazi and B. Ravindran, “Adaptive Resource Management for Scalable, Dependable Real-time Systems: Middleware Services and Applications to Shipboard Computing Systems,” *IEEE Real-time Technology and Applications Symposium*, June 1998.
[27] L. R. Welch and B. A. Shirazi, "A Dynamic Real-time Benchmark for Assessment of QoS and Resource Management Technology," *IEEE Real-time Technology and Applications Symposium*, June 1999.