Optimal Control for Real-Time Feedback Rate-Monotonic Schedulers*

Tolga Ayav[1] and Giancarlo Ferrari-Trecate[2]

[1] INRIA Rhône-Alpes ZIRST 655, Avenue de l'Europe,
38334 Montbonnot, St Ismier Cedex, France. Tel: +33 4 76 61 52 17. Fax: - 54 77.
[2] INRIA, Domaine de Voluceau Rocquencourt - B.P.105
78153, Le Chesnay Cedex, France. Tel: +33 1 39 63 59 21. Fax: - 57 86.
{Tolga.Ayav, Giancarlo.Ferrari-Trecate}@inria.fr

Abstract. This paper presents an optimal control scheme for a real-time feedback control rate-monotonic scheduling (FC-RMS) system. We consider two-version tasks composed of a mandatory and an optional part to be scheduled according to the FC-RMS. In FC-RMS, the controller provides a feedback strategy for deciding about the execution or rejection of the optional sub-tasks. By modeling the task execution times as random variables, we first find the statistical model of FC-RMS and then we design a pure optimal controller and an optimal controller with feedforward integral compensation. The comparison of these two schemes with common Proportional-Integral-Derivative (PID) controller highlights the benefit of the optimal scheme with integral compensation. The results are demonstrated through the real implementation of FC-RMS on RT-Linux.

1 Introduction

Real-Time (RT) dynamic scheduling algorithms traditionally fall into two categories: static and dynamic priority-driven. One major paradigm for the static priority-driven scheduling is Rate-Monotonic (RM). The main drawback of RM is that it considers WCETs, which results in systems having spare capacity under normal operation [1]. Dynamic priority driven scheduling can be further divided into two categories: algorithms that work in *resource sufficient* environments and algorithms that work in *resource insufficient* environments. In the first category, resources are sufficient in the sense that all the tasks are schedulable at any given time despite their unknown arrival times. Earliest-Deadline-First (EDF) has been proved to be an optimal dynamic scheduling algorithm in resource sufficient environments [1]. On the other hand, it may be impossible to guarantee that the resources are sufficient in unpredictable environments.

According to [2] and [3], the next generation of real-time systems will be more complex and capable of adaptivity as well as of meeting time constraints

* This work has been partially supported by the cooperation project between İzmir Institute of Technology and Institute Aéronautique et Spatial in 2003.

P. Yolum et al.(Eds.): ISCIS 2005, LNCS 3733, pp. 894–903, 2005.

for mission and safety critical functions. A challenge of the current research is, therefore, to use more adaptive techniques such as feedback control scheduling in order to handle the transient overload, to use the spare capacity existing with traditional algorithms and to enhance the fault tolerance of real-time systems [2] [3] [4].

Although feedback control scheduling has been considered so far in different works, many questions are still open. The most important one is how to design a controller in order to guarantee the stability of the overall system. Most of the contributions in the literature focused on the use of PID controllers. In this work, we consider a FC-RMS system with optimal and feedforward compensated optimal controllers. We first construct a novel optimal controller that relies on the estimated task statistics. We show that the optimal controller is not able to track a given setpoint, while it provides better performance, during the transient, than PID. Then, in order to eliminate the steady-state error, we enhance the optimal control action with a feedforward integral action.

2 Feedback Control Rate-Monotonic Scheduling Architecture

In this section, we present the architecture we consider that integrates feedback control and rate-monotonic scheduling. The model is developed for a generic control action.

2.1 Feedback Control Rate-Monotonic Scheduling

FC-RM scheduling features a feedback control loop that is invoked at every sampling interval. It is composed of a Monitor, a Controller and a Task Level

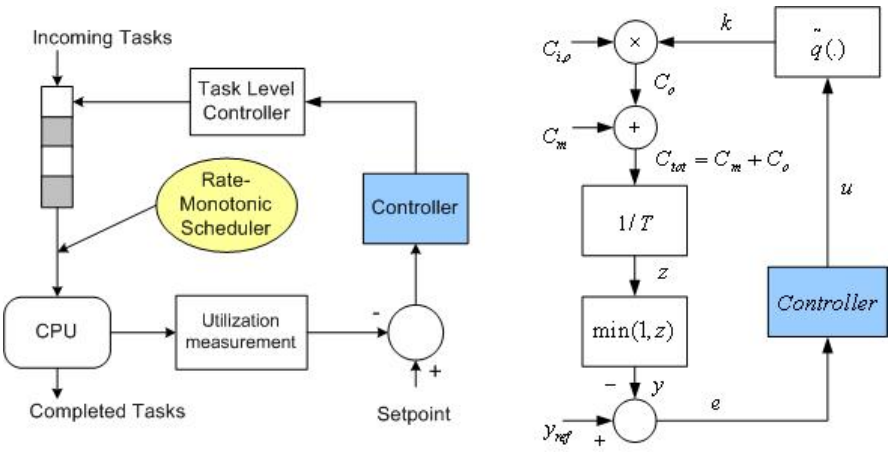

Fig. 1. Feedback Control Rate-Monotonic Scheduling. The left panel shows the schematic diagram and the right panel shows the mathematical model.

Controller (see figure 1). We choose as a controlled variable the CPU utilization (which is defined as the percentage of CPU busy time in a sampling period) since in hard real-time systems the miss ratio should be always kept zero [3]. In each sampling instance, the CPU utilization is monitored and the error is obtained by comparing it with the setpoint. The controller takes the error, produces a control value to minimize the error, i.e., to keep the utilization at a given set-point. The task level controller takes the output of the controller and adjusts the manipulated variable (the percentage or number of the optional sub-tasks to be executed) accordingly. In order to keep the overhead at an acceptable level, the controller output is updated at a frequency $1/T$, where T denotes the sampling period (that is equal to 100 ms in our experiments, see table 1).

We use the Rate-monotonic (RM) algorithm to schedule the mandatory sub-tasks since, under suitable assumptions (see [1]), it can guarantee that tasks will meet their deadlines. On the other hand, making decision on which optional tasks or how many optional tasks will be executed is a much more complex problem. Feedback control scheduling is a method for solving it.

We consider a set of tasks $Tasks = \{\tau_1, \tau_2, ..., \tau_q\}$ where each task is characterized by some parameters such as deadline d_i (this is considered as period T_i if the task is periodic), processing time C_i and priority p_i. Each task consists of two sub-tasks, the mandatory part M_i and the optional part O_i. The processing times of M_i and O_i are $C_{i,m}$ and $C_{i,o}$, respectively. Thus, $C_{i,m} + C_{i,o} = C_i$, $i \in \{1, \ldots, q\}$.

The evaluation of the CPU utilization in a time interval of length T can be represented by the scheme in figure 1 that we illustrate next. Let $t \in \mathbb{N}$ denote the discrete time index, i.e. the actual time $\tilde{t}$ can be computed as $\tilde{t} = Tt$. The total number N of the tasks at time t, is then given by $N = \sum_{i=1}^{q} \lceil T/T_i \rceil$. Note that $N_i = \lceil T/T_i \rceil$ is the number of instances of task τ_i in the period $[tT,\ (t+1)T]$ and $\lceil \cdot \rceil$ is the ceil function returning the lowest integer upper bound to the argument. The total execution time of all optional parts within a sampling period depends on the scheduling policy. If optional sub-tasks are scheduled according to Sieve method, each one of them is entirely executed or rejected [5]. Assume that the control action $u(t)$ is such that the optional sub-tasks of tasks $\tau_1, \tau_2, ..., \tau_k$ will be executed in the period $[tT,\ (t+1)T]$. Then, the total execution times of all mandatory instances and of all optional parts within a sampling period, under the assumption that $\tau_{1,o}, \tau_{2,o}, ..., \tau_{q,o}$ are ordered according to their priorities $p_1 > p_2 > ... > p_q$, are given by

$$C_m(t) = \sum_{i=1}^{q} C_{i,m}(t) \cdot \lceil T/T_i \rceil, \qquad C_o(t) = \sum_{i=1}^{k} C_{i,o}(t) \cdot \lceil T/T_i \rceil. \tag{1}$$

Clearly, in (1), $k = k(t)$ is a time-varying quantity. The signal $z(t)$, the CPU utilization $y(t)$ and the error signal $e(t)$ represented in figure 1 are given by

$$z(t) = \frac{C_m(t) + C_o(t)}{T}, \qquad y(t) = min\{1,\ z(t)\}, \qquad e(t) = y_{ref} - y(t). \tag{2}$$

The error signal is the difference between the requested utilization y_{ref} and the measured one, i.e. the controller takes the error, processes it and produces the value of $u(t)$ in the range of [0, 1]. Various control algorithms will be discussed in the next section. The output of the controller is then quantized by the block $\widetilde{q}(.)$ seen in figure 1. The quantizer function determines the number of optional sub-tasks to be executed by mapping [0, 1] to [0, q].

2.2 Statistical Characterization of the Signals

In order to account for the unpredictability of task durations, we model them as *independent* random variables with finite average and variance over which the scheduler has no control.

If there is a sufficient number of mandatory sub-tasks and a sufficient number of optional sub-tasks to be executed within a sampling period, the Central Limit Theorem (CLT) can be used [6] in order to approximate the distributions of C_m and C_o. In our setting, the total number of mandatory sub-tasks N_m is equal to N and the total number of optional sub-tasks to be executed is $N_o = \sum_{i=1}^{k} N_i$. We point out that, despite the fact that CLT is an asymptotic theorem, it provides good approximations even if N_m and N_o are low (see [6]). In our experiments, we have $N_m = 303$ and $N_o = 181$ as can be calculated from table 1. An important fact is that CLT holds even if the task durations are not uniformly distributed. Therefore, all our results hold for general distributions (with finite average and variance) associated to $C_{i,m}$ and $C_{i,o}$.

For a generic signal ξ, we denote with $f_\xi(\xi|\kappa)$ its Probability Density Function (PDF) for the choice of executing κ optional sub-tasks at time t. Let $g(\mu, \sigma^2)$ be the Gaussian distribution with mean μ and variance σ^2. Then, for a fixed k, CLT states that C_m and C_o can be represented by

$$C_m \sim f_{C_m}(C_m|k) = g(\mu_m, \sigma_m^2),\ \mu_m = \sum_{i=1}^{q}\sum_{j=1}^{N_i} \mu_{C_{j,m}},\ \sigma_m^2 = \sum_{i=1}^{q}\sum_{j=1}^{N_i} \sigma_{C_{j,m}}^2 \quad (3)$$

$$C_o \sim f_{C_o}(C_o|k) = g(\mu_o, \sigma_o^2),\ \mu_o = \sum_{i=1}^{k}\sum_{j=1}^{N_i} \mu_{C_{j,o}},\ \sigma_o^2 = \sum_{i=1}^{k}\sum_{j=1}^{N_i} \sigma_{C_{j,o}}^2. \quad (4)$$

According to the scheme reported in figure 1, C_{tot} is the sum of C_m and C_o. Thus, in view of the statistical independence of C_m and C_o, we have

$$C_{tot} \sim f_{C_{tot}}(C_{tot}|k) = g(\mu_{tot}, \sigma_{tot}^2) = g(\mu_m + \mu_o, \sigma_m^2 + \sigma_o^2). \quad (5)$$

The distribution function of z, defined in 2, is given by

$$z \sim f_z(z|k) = g(\mu_z, \sigma_z^2) = g\left(\frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}\right). \quad (6)$$

For the distribution of y, we have

$$y \sim f_y(y|k) = \begin{cases} f_z(y|k) & y < 1 \\ (1 - \int_{-\infty}^{1} f_z(z|k)dz)\delta(y-1) & y = 1 \\ 0 & y > 1 \end{cases} \quad (7)$$

Note that the block $\min(1, z)$ in figure 1 produces a Dirac delta function into the distributions of y and e (see [6]). In fact, (7) represents a truncated Gaussian distribution and the coefficient $\beta = 1 - \int_{-\infty}^{1} f_z(z|k)dz$ multiplying the δ function is significantly different from zero only if the average CPU utilization y is close to one (in fact, β represents the probability of CPU overload for a given k). On the other hand, when the setpoint is chosen sufficiently away from 1 and a stabilizing controller is used, the delta function disappears (see [7] for further details). This corresponds to remove the block $min(1, z)$ in figure 1. We highlight that this approximation is realistic since the set point should be chosen pretty far away from 1 in order to prevent undesirable saturation effects [3]. Hence, the PDFs of y and e can be approximated as

$$y \sim g\left(\frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}\right), \quad e \sim g\left(y_{ref} - \frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}\right). \tag{8}$$

The PDF of $u(t)$, $f_u(u|k)$ depends on the specific control scheme. Stankovic and Lu considered the use of PID control since it is a well-established technique in automatic control and in some cases is able to stabilize the scheduling system. For stability, it is of paramount importance to properly choose the controller parameters. Stankovic and Lu presented a method for tuning the PID parameters that relies on a deterministic scheduler model. Another approach to tune the PID parameters is given in [7], which presents the design of stabilizing PID controller based on the statistical framework presented in this paper.

On the other hand, the choice of more effective control schemes is still an important research issue. In the next section, we propose an optimal control scheme as an alternative to PID.

3 Optimal Control

Optimal control relies on the estimated task statistics. The block diagram of the optimal control rate-monotonic scheduling system is given in figure 2. The goal of the optimal control is to compute $u(t)$ that minimizes the variance of the error. For sake of clarity, it is assumed that there is no quantization in the system and the setpoint is far from 1, i.e., min(1,z) box is removed. Thus, the utilization $y(t)$ can be written as

$$y(t+1) = \frac{1}{T}(C_m(t) + C_{o,tot}(t)u(t)) \tag{9}$$

The cost function is the mean square error that, using equation 5.34 from [6], can be written as

$$J(t+1) = E[e^2(t+1)] = E^2[e(t+1)] + Var[e(t+1)]. \tag{10}$$

In (10), it holds that $E^2[e(t+1)] = (-\hat{y}(t+1|t) + y_{ref}(t+1))^2$ where $\hat{y}(t+1|t)$ is the optimal predictor of $y(t+1)$ on the basis of the information collected up to

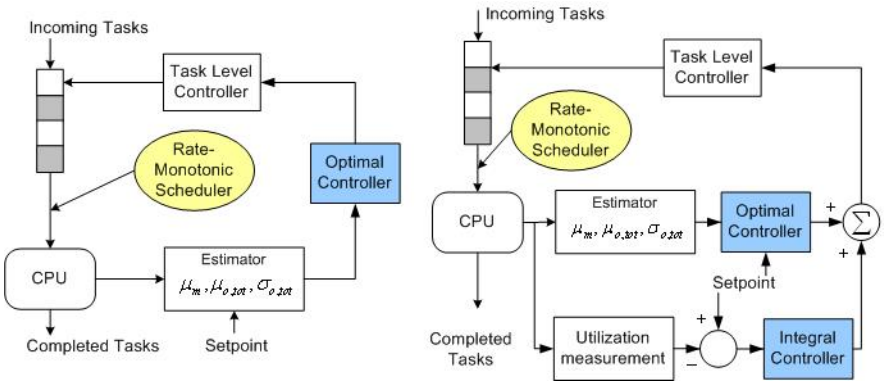

Fig. 2. Optimal Control Rate-Monotonic Scheduling. The left panel shows the system with pure optimal controller and the right panel shows the system with feedforward integral compensated optimal controller.

time t. According to certainty equivalence principle [8], the optimal prediction of $y(t)$ is given by $\hat{y}(t) = y_d(t+1)$ where $y(t) = y_d(t)+\nu(t)$. Here, $y_d(t)$ coincides with the optimal prediction of the deterministic part of $y(t)$ and $\nu(t)$ is a white noise. This principle will be applied to find $\hat{y}(t+1|t)$. First, $y(t+1)$ can be written as

$$\begin{aligned} y(t+1) &= \underbrace{\frac{1}{T}\mu_m(t) + \frac{1}{T}\mu_{o,tot}(t)u(t)}_{y_d(t)} + \underbrace{\frac{1}{T}\tilde{C_m}(t) + \frac{1}{T}\tilde{C_{o,tot}}(t)u(t)}_{\nu(t)} \qquad (11) \\ &= \end{aligned}$$

The uncorrelation between $\nu(t)$ and $\nu(t-\tau)$ $\forall\tau \geq 1$, can be easily proved from the following properties: 1) $u(t)$ depends only on the past history of C_m and C_o, i.e., from $C_m(t-\xi)$ and $C_o(t-\xi)$, $\xi \geq 1$; 2) $\tilde{C}_m$ and $\tilde{C}_o$ are zero mean random variables. Thus, $\nu(t)$ is a white noise with zero mean. By applying the certainty equivalence principle, the optimal predictor of $y(t+1)$ is given by

$$\hat{y}(t+1) = \frac{1}{T}(\mu_m(t) + \mu_{o,tot}(t)u(t)). \qquad (12)$$

On the other hand, $Var[e(t+1)]$ takes the following form:

$$Var[e(t+1)] = E[(e(t+1) - E[e(t+1)])^2] = \frac{1}{T^2}\sigma_m^2(t) + \frac{1}{T^2}\sigma_{o,tot}^2(t)u^2(t).$$

By using equations (12) and (13) in (10), the cost function $J(t+1)$ can be written as

$$J(t+1) = \left(y_{ref}(t+1) - \frac{1}{T}\mu_m(t) - \frac{1}{T}\mu_{o,tot}(t)u(t)\right)^2 + \frac{1}{T^2}\sigma_m^2(t) + \frac{1}{T^2}\sigma_{o,tot}^2(t)u^2(t). \qquad (13)$$

As a final step, $u^*(t)$ (the optimal value of $u(t)$, which minimizes $J(t+1)$) is found by imposing $\frac{dJ(t+1)}{du(t)} \mid_{u^*(t)} = 0$ thus obtaining

$$u^*(t) = \frac{-\mu_m(t)\mu_{o,tot}(t) + T\, y_{ref}(t+1)\mu_{o,tot}(t)}{\mu^2_{o,tot}(t) + \sigma^2_{o,tot}(t)} \tag{14}$$

Equation (14) highlights that if one can estimate the mean of all mandatory sub-tasks and the mean and variance of all optional sub-tasks within a sampling period, then $u^*(t)$ can be calculated in closed-form. When $u(t) = u^*(t)$ is used, the CPU utilization obeys to the dynamics

$$y(t+1) = \frac{1}{T}\left(\frac{\mu_m(t)\sigma^2_{o,tot}(t) + \mu^2_{o,tot}(t) T y_{ref}(t+1)}{\mu^2_{o,tot}(t) + \sigma^2_{o,tot}(t)}\right) \tag{15}$$

as $t \to \infty$. Note that even if the task statistics and the setpoint are constant in time, if $\sigma^2_{o,tot}(t) \neq 0$, one has, in general, that $y(t)$ does not converge to y_{ref} as $t \to \infty$. Moreover, incorrect estimations of the task statistics may further degrade the performance of the optimal controller.

4 Optimal Control with Feedforward Compensation

In order to avoid the bias produced by optimal control, we propose to add a feedforward integral control action. The block diagram of this scheme is given in figure 2.

In order to find a stabilizing controller, we first derive the state space form of the closed-loop system. Assume that $x \in \mathbb{R}$ is the state of the integral controller. Hence, controller equations are,

$$x(t+1) = x(t) + e(t)\,,\; u(t) = K_i x(t) + \left(\frac{-\mu_m(t)\mu_{o,tot}(t) + T\, y_{ref}(t+1)\mu_{o,tot}(t)}{\mu^2_{o,tot}(t) + \sigma^2_{o,tot}(t)}\right)$$

and the system equations are,

$$y(t) = \frac{1}{T}\left[C_m(t) + C_{o,tot}(t)\left(K_i x(t) + \frac{-\mu_m(t)\mu_{o,tot}(t) + T\, y_{ref}(t+1)\mu_{o,tot}(t)}{\mu^2_{o,tot}(t) + \sigma^2_{o,tot}(t)}\right)\right]$$

$$e(t) = -y(t) + y_{ref}(t).$$

The closed-loop system is therefore described by

$$\begin{aligned} x(t+1) = x(t) + y_{ref}(t) - \frac{1}{T}C_m(t) \\ - \frac{1}{T}C_{o,tot}(t)\left(K_i x(t) + \frac{-\mu_m(t)\mu_{o,tot}(t) + T\, y_{ref}(t+1)\mu_{o,tot}(t)}{\mu^2_{o,tot}(t) + \sigma^2_{o,tot}(t)}\right) \end{aligned} \tag{16}$$

By assuming that $C_{o,tot}$, C_m and y_{ref} are stationary signals, the mean state dynamics is

$$\mu_x(t+1) = \mu_x(t) + y_{ref} - \frac{1}{T}\mu_m - \frac{K_i}{T}\mu_{o,tot}\mu_x(t) - \frac{1}{T}\left(\frac{-\mu_m\mu^2_{o,tot} + T\, y_{ref}\mu^2_{o,tot}}{\mu^2_{o,tot} + \sigma^2_{o,tot}}\right)$$

where $\mu_x(t) = E[x(t)]$. A classical stability criterion for discrete time linear systems [9] guarantees that $\mu_x(t)$ is asymptotically stable around its equilibrium, if the following condition is fulfilled

$$\left|1 - \frac{K_i}{T}\mu_{o,tot}\right| < 1. \tag{17}$$

Then, formula (17) provides an explicit bound on the values of K_i guaranteeing the convergence of the average error to zero.

5 Implementation on RT-Linux and Experimental Results

We tested FC-RMS with PID, optimal and compensated optimal controllers using the RT-Linux system (a detailed description of FC-RMS with stabilizing PID controllers is provided in [7]).

RT-Linux has a priority-driven preemptive scheduler loaded as a kernel module [10]. In order to implement our FC-RMS method on RT-Linux 3.1, we modified the default scheduler so as to measure CPU utilization and other task statistics μ_m, $\mu_{o,tot}$ and $\sigma_{o,tot}$. A dummy application consisting of a periodic task set has also been created.

The CPU utilization y, the mean of all mandatory tasks μ_m, and the mean and variance of k optional sub-tasks within one sampling period are easy to reconstruct by resorting to the empirical estimators for the mean and variance of stationary stochastic processes [6]. However we have to estimate the mean and variance of all optional sub-tasks $\mu_{o,tot}$ and $\sigma^2_{o,tot}$ in order to use formula (14). Under the assumption that the mean and variance of each optional sub-task are constant, i.e., $\mu_{C_{j,o}} = \mu_c$ and $\sigma^2_{C_{j,o}} = \sigma_c$, $\forall j \in \{1, 2, \ldots, q\}$, one gets $\mu_{o,tot} = \mu_o . \frac{q}{k}$ and $\sigma^2_{o,tot} = \sigma^2_o . \frac{q}{k}$.

In the optimal controller, there is no parameter to tune. However it needs accurate estimates of the task statistics that may be difficult to obtain when the system is overloaded [3]. Another drawback is that the mean and variance of each optional sub-task are not constant in time, which results in incorrect estimations. These two pitfalls are overcome by using the feedforward integral controller.

In order to compare the performance of PID, optimal and compensated optimal controllers during transients, we used a step workload jumping from the nominal load L_{nom} to a maximum load L_{max} at a given time instant, as reported in table 1.

A comparison of figures 3.A, 3.B, and 3.C shows that pure optimal control provides the best transient characteristics in terms of settling time and overshoot. However, it cannot track the setpoint due to the bias shown in formula (15). The additional feedforward integral controller compensates this undesired phenomenon thus providing a better error tracking while preserving satisfactory performance in the transient.

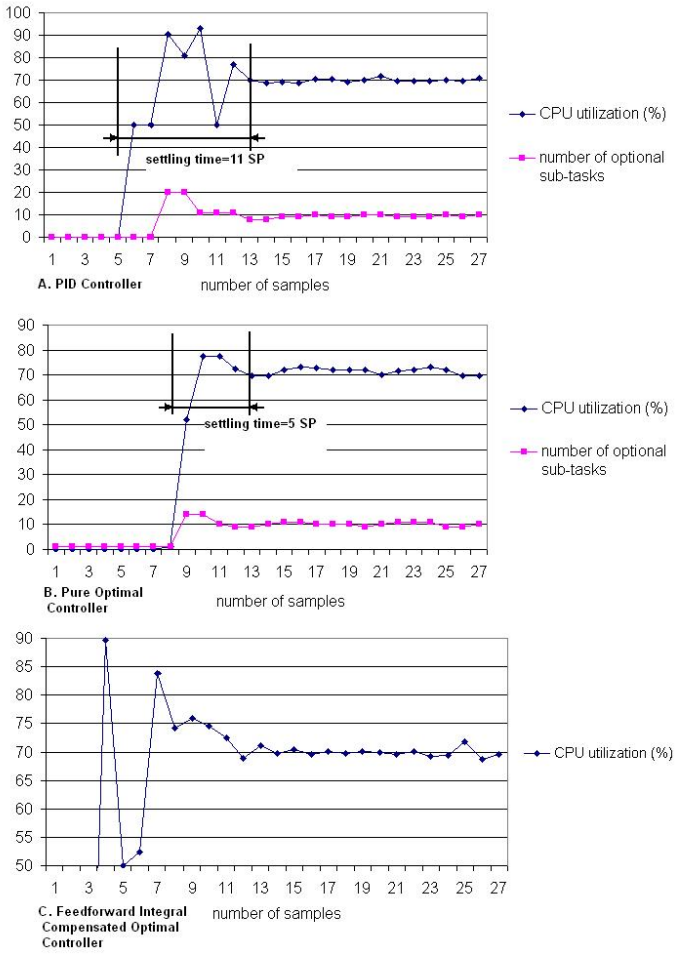

Fig. 3. Experimental results. A) CPU utilization with PID controller. $K_p = 0.1$, $K_i = 2.0$ and $K_d = 0.1$. B) CPU utilization with pure optimal controller. C) CPU utilization with feedforward integral compensated optimal controller.

Table 1. Parameters characterizing the workloads L_{nom} and L_{max}. $U[a, b]$ represents uniform distribution between a and b.

	L_{nom}	L_{max}
q	0	20
$C_{i,m}$ and $C_{i,o}$ (μs)	-	U[130, 180]
$T_{i\in\{1,\ldots,q\}}$ (ms)	-	{15,2,14,3,13,4,12,5,11,6,10,7,9,8,8,35,8,15,3,10}
T (ms)	100	100

6 Conclusions

In this paper, we presented a feedback control rate-monotonic scheduling system. By assuming that all tasks are periodic, independent and implemented with the two-version method, we first derived a statistical representation of the scheduling system. Then, we proposed a novel feedforward compensated optimal controller as an alternative to PID. The new controller achieves better performance (in terms of overshoot, settling time and error tracking) as shown in the experimental results obtained from a real implementation of FC-RMS on RT-Linux. We highlight that the developed statistical framework can be very useful for designing and analyzing feedback control scheduling systems based on other control strategies such as hybrid control. Performance comparisons of PID and optimal controllers on real applications with different types of tasks (such as aperiodic, dependent) and different implementation techniques (such as milestone and multiple-version) still require additional research, which may lead to major generalization of the proposed method.

References

1. Liu, C., Layland, J.: Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM **20** (1973) 40–61
2. Lu, C., Stankovic, J., G.Tao, Son, S.: Design and evaluation of a feedback control EDF scheduling algorithm. In: Proc. 20^{th} IEEE Real-Time Systems Symposium, Phoenix, Arizona (1999) 56–67
3. Lu, C., Stankovic, J., G.T., Son, S.: Feedback control real-time scheduling: Framework, modeling, and algorithms. Real-Time Systems Journal. Special Issue on Control Theoretical Approach to Real-Time Computing **23** (2002) 85–126
4. Lawrence, D., Guan, J., Mehta, S., Welchr, L.: Adaptive scheduling via feedback control for dynamic real-time systems. In: 20^{th} International Performance, Computing and Communications Conference. (2001)
5. Liu, J., Shih, W.K., Lin, K.J., Bettati, R., Chung, J.Y.: Imprecise computations. Proceedings of IEEE **82** (1994) 83–94
6. Papoulis, A.: Probability, Random Variables and Stochastic Processes. McGraw-Hill International Editions (1987)
7. Ayav, T., Ferrari-Trecate, G., Yılmaz, S.: Stability properties of adaptive real-time feedback scheduling: A statistical approach. In: 12^{th} Real-Time Embedded Systems Conference, Paris (2004) 259–277
8. Filatov, N.M., Unbehauen, H.: Adaptive Dual Control. Springer Verlag, Heidelberg (2004)
9. Franklin, G., Powell, J., Emami-Naeini, A.: Feedback Control of Dynamic Systems. Prentice-Hall International (2002)
10. Barabanov, M.: A Linux-based Real-Time Operating System. PhD thesis, New Mexico Institute of Mining and Technology (1997)