The Large Error First (LEF) Scheduling Policy for Real-Time Control Systems

José Yépez, Pau Martí and Josep M. Fuertes
Dept. of Automatic Control and Computer Engineering
Technical University of Catalonia
Pau Gargallo 5, 08028 Barcelona, Spain
{jose.yepez, pau.marti, josep.m.fuertes}@.upc.es

Abstract- **Most of the real-time scheduling algorithms are based on "open-loop" strategies that do not take application demands into account. This precludes the scheduler to dynamically adjust task executions in order to optimize performance. To overcome this limitation, we have focused our work on scheduling techniques that are able to take scheduling decisions based on continuous feedback information of the performance delivered by each task. Focusing on control applications, we present an early specification of a novel scheduling technique: Large Error First (LEF). It uses feedback information from each controlled plant in order to assign priorities to each control task. For a given simulation set-up, comparing the performance of LEF versus open loop classical scheduling techniques, encouraging simulation results have been obtained.**

I. INTRODUCTION

Classical real-time scheduling algorithms depend on a priori characterization of the workload to provide performance guarantees in predictable environments. For example, Rate Monotonic (RM) and Earliest Deadline First (EDF) [4] require complete knowledge about the task set such a resources requirements, precedence constraints, resource contention, and future arrival times. Such algorithms work in "open-loop". That is, they are based on fixed parameters that are configured at system set-up. At run time, resources are accessed by all tasks following pre-established rules regardless the application requirements. Once the scheduling rules are created, they are not adjusted based on continuous feedback from the application.

"Open-loop" scheduling approaches work fine in predictable/static environments. Note however that they provide determinism assuming worst-case conditions. This led us to the following observations. First, pessimism implies a conservative allocation of execution time that may result in poor system performance in terms of CPU utilization. Second, environmental conditions may change introducing various degrees of unpredictability. In this case these scheduling algorithms may perform poorly in the sense that they do not have feedback mechanisms to monitor the application and tune the scheduling according to new and changeable environment dynamics. And third, although they had the monitoring capacity, they would not be able to adjust their operation because their tasks are based on fixed timing constraints such as (constant) periods and deadlines.

Looking at real-time control systems, the traditional approach is to treat control tasks as hard periodic real-time tasks with fixed periods and deadlines. This is a reasonable assumption for many control applications and open-loop scheduling policies can fulfill with most of the application performance specifications. Note however that this scheme fall into the type of systems we criticized in the previous paragraph.

Focusing on control applications and in order to overcome the problems outlined earlier, we have focused our work on scheduling techniques that are able to take scheduling decisions based on continuous feedback information of the performance delivered by each control task.

In this paper we present an early specification of a novel scheduling technique: Large Error First (LEF). It uses feedback information from each controlled plant in order to assign priorities to each control task. For a given simulation set-up, comparing the performance of LEF and classical scheduling techniques such as RM or EDF, encouraging simulation results have been obtained.

The rest of this paper is as follows. Section II surveys related work, thus adding further motivation for the work. Afterwards, the Large Error First scheduling technique is described in Section III. Section IV details the experimental case study and gives the simulations results. Finally, in section V, conclusions are discussed and future work is outlined.

II. RELATED WORK

Lately, significant effort has been made in the area of real-time and control systems research. As it has been shown in several works (see for example [2], [5], or [6]), approaches that combine real-time and control disciplines offer effective solutions for the analysis and design of both real-time control systems and feedback scheduling approaches.

In particular, [2] presents a scheduling architecture for real-time control tasks in which feedback information is used to adjust the workload of the processor and to optimize the overall control performance by simple rescaling of the task periods. The approach works fine if the sampling periods are chosen wisely, that is, for plants sampled reasonably fast. Our approach aims to take advantage of each plant dynamics in order to assign processor capacity to the tasks that more urgently need to issue the control signal. This may result in slow sampling for other control tasks in the system.

[5] presents a framework for adaptive real-time systems where feedback control scheduling algorithms are designed to satisfy the transient and steady state performance specifications of real-time systems. Note that although the framework uses the idea of feedback, it is not intended for control applications. Our scheduling technique is for control tasks.

The approach we present follows the work of [6], in which the authors stressed the need for new scheduling approaches able to optimize control performance according to the control application dynamics. The technique we present is based in the following paradigm: the highest priority is dynamically assigned to the control task with

largest error (information that is feed back to the scheduler from each controlled plant).

The goal of the work we present is to treat the scheduling policy as a fundamental part of the control application. Scheduling decisions are taken according to the application needs. The feedback information we use between the scheduler and each controlled plant is a function of the state of the plant. This gives us an easy and fast way to reassign priorities.

III. LARGER ERROR FIRST SCHEDULING

The novel paradigm we introduce for the scheduling of control tasks is called Large Error First (LEF). The policy is aimed to be dynamic and flexible. The adjustment of the produced schedule is based on continuous feedback of the performance of the systems being controlled.

A. LEF scheduling concept

The LEF scheduling algorithm is an online scheduling policy that assigns priorities as a function of the state of the controlled plants. The states of the controlled plants are the minimum number of variables that are needed to characterize the actual behavior of each plant.

Note that controllers, beyond meeting the system response characteristics (such as transient response and steady-state accuracy) and stability requirements [1], attempt to minimize the closed-loop system error (difference between the desired response of the system and the actual response of the system) for any given perturbations. In other words, controllers try to optimize control performance. As it was shown in [6], upon a perturbation arrival, the higher the frequency a controller is given, the better the control performance.

Treating the scheduling policy as a fundamental part of the control application means to assign to the scheduler the same objectives than controllers have. That is, to optimize control performance.

Therefore, to start with, the rule that we define for our scheduling policy is the following: at any given time, the plant with largest error, e_i (obtained from their state variables), will be assigned the highest priority. As a result, more CPU will be assigned to those control loops that more urgently need to issue the control signal in order to correct larger deviations from the plant expected behavior. More CPU assignment means higher task execution rates, implying, as discussed above, better control performance.

B. LEF scheduling operation

The basic LEF operation can be described as follows: after each task instance completion, the scheduler scans all the plants' states. If all the plants are in equilibrium (error $e_i=0$ for all plants), the actual schedule is kept. If some of the plants are in a transient state ($e_i>0$ for some of the plants), priorities are assigned according the policy discussed before and illustrated in Figure 1. Let us assume for Figure 1 that we have two responses of two plants, each one controlled by an independent task. At time t_k (coinciding with the completion of a task instance) the error of the response at each plant is measured (e1 and e2) and a highest priority is assigned to task2 because the response of plant2 is suffering a largest error than plant1.

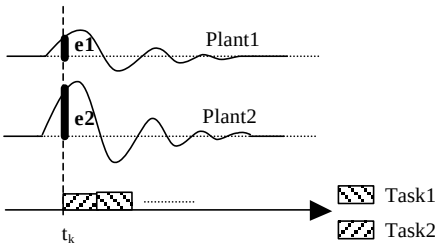

Fig. 1. LEF scheduling operation

Note that this basic operation does not guarantee any minimum execution rate for any of the control tasks, which would violate the basic principles of the sampling theory. To overcome this problem, several possibilities are being studied: a) to reassign priorities not after each task instant completion but after all tasks have executed once, b) to enforce a minimum rate for the less frequently executed tasks and c) combination of both.

However, for the purpose of this paper, the basic operation explained above is good enough to show the properties the LEF scheduling policy can provide for real-time control systems.

IV. SIMULATION RESULTS

To analyze the improvement that the LEF scheduling algorithm can deliver, we present in this section preliminary simulation results. The simulation set-up we have designed corresponds to the following real-time control system: we consider three inverted pendulums, each one controlled by an independent task. The three tasks will execute on a real-time kernel. The performance delivered by each task is obtained under different scheduling algorithms (such as EDF, RM and LEF) by simulation. Simulations have been performed using [3].

A. Plants and controller

From the linear time invariant state-space model that we used to model each inverted pendulum (each mounted on a motor driven cart), we can now through its state variables the cart position and speed and the pendulum angle (respect to the vertical position) and angular velocity. For the sake of clarity, for the LEF simulations we used as a relevant plant state only one variable, the pendulum angle. Therefore, the error used to assign priorities was defined as the difference between 0º and the angle (deviation of the inverted pendulum with respect to the vertical position).

Each control task instance sequentially samples the plant, executes the A-D conversion, computes of the signal control according to the control law, executes the D-A conversion and send the control signal.

Each inverted pendulum is controlled by a control task implementing a PID controller [1]. We heuristically tune the PID controller parameters for the three tasks to the same values. In this way, the PID tuning will not affect the simulations.

However, in order to avoid harmonic relationship among sampling periods, we chose different values for the sampling period of each task. The sampling periods were chosen according to the one of the rules of thumb [1] used for selecting such parameter. Thus, the sampling periods for each task are 10, 13 and 14 ms . Note that although different periods will give different performance figures for each pendulum, the difference is negligible if compared

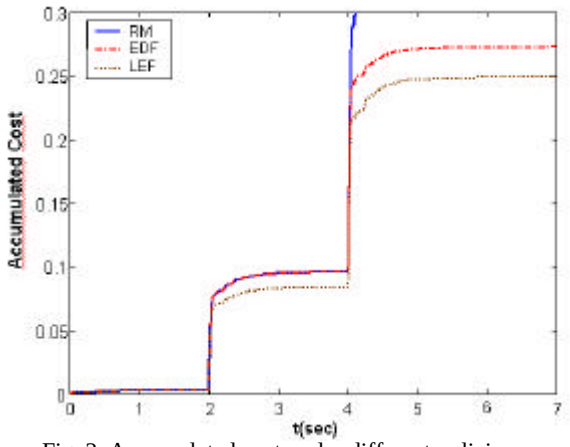

Fig. 2. Accumulated cost under different policies.

with the difference in performance due to the use of different scheduling policies, which is the objective of our simulations.

The performance of each pendulum is obtained by the following cost function J_i, which measures the error e_i of the pendulum weighted with time t.

$$J_i(t) = \int_0^t t e_i^2(t) dt$$

Note that t sets the duration of the evaluation period, which typically should go from the perturbation arrival time (0 in the integral) to the settling time. As higher values the cost function gets, the worst the control performance (because major deviations occur or because it takes more time for the inverted pendulum to recover from the perturbations).

B. Results

The simulation we run keeps the following time sequence: at time t=0, only the control task controlling the fist pendulum is released. After executing alone, at time t=2, another control task is released to control the second pendulum. Finally the third control task is released at time t=4. The three controllers run in parallel until t=7. Before the release time of each control task, the corresponding pendulum is in equilibrium. At release time of each control task, the pendulum suffers a perturbation (of equal magnitude for each pendulum). This simulation pattern is repeated for different scheduling algorithm: RM, EDF and LEF. During each experiment, the cost function presented earlier and the resulting schedule are recorded. The cost function evaluation period goes from the beginning of each simulation, t=0, to the simulation completion, at t=7. For each scheduling policy, the results we obtained are summarized in the following:

- **RM:** RM is a static scheduling algorithm in open loop, which assigns priorities to tasks according to their request rates. The cost function values for the three pendulums are shown in the Figure 2 (note that Figure 2 shows the accumulated cost function, ΣJ_i, during the evaluation period for each scheduling policy). Under RM the schedulability condition for our experiment is given by $U < 0.78$. Taking into account that task execution time is 4ms, until t=4, the processor utilization is 0.71 and the control performance is good. From t=4, the tree control tasks run in parallel and these consume 0.99 CPU. That is, task3 misses deadlines and the inverted pendulum1 falls down. This explains the fact that the cost function in Figure 2 goes to infinite. Note that under RM, the task is not schedulable.

- **EDF:** EDF is a dynamic algorithm in open loop, which assigns priorities to tasks according to their absolute deadlines. Under this scheduler the schedulability condition is given by $U < 1$. For our simulations, since $U < 1$, the task set is schedulable and the three pendulums can be controlled as it can be seen in Figure 2: the accumulated cost reaches a finite value, which means that the deviation caused by each perturbation that affected each of the three pendulums could be adequately corrected. The performance achieved by EDF is also given in Figure 2 in terms of the cost function, reaching a value of 0.2732 at the completion of the evaluation interval

- **LEF:** LEF is a dynamic scheduling algorithm in closed-loop, which adjusts the schedule based on continuous feedback of each control loop. Under this scheduler, in the simulation we obtain that the three inverted pendulums can be perfectly controlled. As it can be seen in Figure 2, the cost function of LEF goes below the cost function of EDF, meaning that for this particular simple simulation set-up, LEF performs better that EDF. Note that the final cost of LEF is 0.2490.

C. Discussion

The exact cost for each one of the three pendulums under each scheduling algorithm is summarized in the Table 1. RM fails in controlling the third pendulum. EDF and LEF are able to control the three pendulums. However LEF gives the best control in terms of the cost function.

Table 1. Cost for the three inverted pendulums under each different scheduling policy.

Scheduling	J_1	J_2	J_3
RM	0.0033	0.0930	8
EDF	0.0033	0.0930	0.1769
LEF	0.0033	0.0812	0.1645

This results shows that scheduling policies that take advantage of the application dynamics and are able to adjust the schedule accordingly can provide, in some specific scenarios, better performance in terms of the application (control performance in our case).

For open-loop scheduling approaches we identified (see Section 1) three main negative aspects: low real CPU utilization, the lack of feedback mechanism and poor adaptability to the application dynamics. Our strategy optimizes CPU utilization in the sense that control tasks are executed only when they are required. That is, they are executed when the controlled plants suffer perturbations. Otherwise, they execute with the slowest possible rate in order to give room to other tasks with higher priorities. In addition, our approach is based on the idea of feedback, which offers the possibility of taking at run time the

scheduling decisions according to the application dynamics. And finally, we plan that LEF schedule will use flexible timing constraints (see [6]) for the control tasks, which will allow a better both CPU utilization and control performance.

V. CONCLUSIONS

In this paper we have presented an early specification of a novel scheduling paradigm for the scheduling of control loops called Larger Error First (LEF). We have stressed that the main advantage of such new paradigm is the fact that it allows a dynamic adjustment of each closed loop operation according to the application characteristics, and concretely, according to the states of the controlled plants in the form of the system error.

Different open-loop and classic scheduling algorithms such as RM or EDF have been evaluated using a simple example in order to show the performance of LEF compared to the others. The results we have obtained are very promising and they motivate us to continue exploring the possibilities and characteristics of the proposed scheduling policy. Future work will require the exact specification of both the LEF operation and the task model for control tasks, as well as to define the scheduling algorithm and provide methods for the schedulability analysis.

REFERENCES

[1] K.J. Åström and B. Wittenmark. *Computer Controlled Systems. Third edition.* Prentice Hall. 1997.

[2] A. Cervin, J. Eker, B. Bernhardsson, K.-E. Årzén "Feedback-Feedforward Scheduling of Control Tasks", *Real-Time Systems*, 23:1, 2002.

[3] J. Eker, J. and A. Cervin. A Matlab toolbox for realtime and control systems codesign. In 6th International Conference on RealTime Computing Systems and Applications, 1999

[4] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, 20:1, 1973.

[5] C. Lu, J. Stankovic, G. Tao and S. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms", *Real-Time Systems*, 23:1, 2002.

[6] P. Marti, G. Fohler, K. Ramamritham, and J.M. Fuertes, "Improving Quality-of-Control using Flexible Timing Constraints: Metric and Scheduling Issues." In Real-Time Systems Symposium, Dec. 2002.

Real time embedded systems have time constraints linked with the output of the system. The scheduling algorithms are used to determine which task is going to execute when more than one task is available in the ready queue. The operating system must guarantee that each task is activated at its proper rate and meets its deadline. you may also like to read about. real time operating systems. Embedded operating systems.Â Offline Scheduling Algorithm. Offline scheduling algorithm selects a task to execute with reference to a predetermined schedule, which repeats itself after specific interval of time. For example, if we have three tasks Ta, Tb and Tc then Ta will always execute first, then Tb and after that Tc respectively. Online Scheduling Algorithm. Real-time systems are systems that carry real-time tasks. These tasks need to be performed immediately with a certain degree of urgency. In particular, these tasks are related to control of certain events (or) reacting to them. Real-time tasks can be classified as hard real-time tasks and soft real-time tasks. A hard real-time task must be performed at a specified time which could otherwise lead to huge losses. In soft real-time tasks, a specified deadline can be missed.Â In real-time systems, the scheduler is considered as the most important component which is typically a short-term task scheduler. The main focus of this scheduler is to reduce the response time associated with each of the associated processes instead of handling the deadline. Real-Ââ€Time Scheduling. Chenyang Lu. CSE 467S Embedded Compu5ng Systems. Readings. Ã˜ Single-Processor Scheduling: Hard Real-Time Computing Systems, by G. Buttazzo. q Chapter 4 Periodic Task Scheduling q Chapter 5 (5.1-5.4) Fixed Priority Servers q Chapter 7 (7.1-7.3) Resource Access Protocols. Ã˜ Optional further readings. q A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems, by Klein et al. q Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms, by Stankovic et al. Real-Ââ€Time Scheduling. Ã˜ What are the optimal scheduling a... Theory - a supposition or a system of ideas intended to explain something. Conspiracy Theory - a hypothesis that some covert but influential organization is responsible for a circumstance or event. This is a forum for free thinking and for discussing issues which have captured your imagination.Â 5. Titles and comments with an excess of caps lock, bold text, large fonts, text colors, exaggerated punctuation and other attention-seeking devices will be removed. 6. No memes use /r/ConspiracyMemes. Other image posts are subject to removal at moderators discretion.Â So is it just the real time PCR test? Or all of them? permalink. Large Error First (LEF). This scheduling algorithm uses feedback information from each controlled plant by the application in order to assign computing resources to control tasks. The LEF is used in simulation. By comparing the performance ofLEF versus classical openloop scheduling techniques, encouraging simulation results have been obtained. There are still many issues to be addressed with regard to LEF algorithm. This paper also shows these issues, with possible approaches that should be investigatedfurther.