

Simulation study on control performance and resource utilization for resource-constrained control systems

Camilo Lozoya¹, Pau Martí¹, Manel Velasco¹ and Josep M. Fuertes¹

¹ Automatic Control Department, Technical University of Catalonia
Pau Gargallo 5, 08028
Barcelona, Spain

Research Report: ESAII-RR-09-01

March 2009

Abstract

Control performance optimization and the efficient use of the available resources, are two key elements in the design and implementation of multitasking real-time control systems. Based on this, two main approaches have been used in order to provide solutions: feedback scheduling (FS) and event-based scheduling (ES). The first one focuses in the optimization of control performance achieved by all tasks using the available resources. The later one has as a main goal the optimization in the use of resources while ensuring system stability. This report presents a control performance and resource efficiency evaluation of existing FS and ES methods. The methods have been selected due their representative characteristics within this two tendencies. The results obtained show that both ES and FS methods provides considerably and similar control performance improvements when there is not any constraints on resources. However when the user of resources is considered in the evaluation, ES methods provides superior results, since they are able to reduce the use of resources with less performance degradation.

Keywords: Embedded systems, real-time control systems, feedback scheduling, event-driven control, performance evaluation

1 Introduction

Real-time computer controlled systems are usually implemented in resource constrained platforms with the intention to achieve the best possible performance. As a result, the design of networked and embedded control systems faces two main challenges: improve control performance and make an efficient use of system resources. In a multitasking real-time control system there are several tasks running concurrently that need to share a critical and common resource such as processor capacity, communication bandwidth, battery life, etc. In order to make these systems cost-effective, it is mandatory to optimize both aspects: control performance and resource utilization [1].

Traditionally, real-time control systems are implemented using hard-real time periodic tasks [2], however this approach fails at minimizing resource utilization and maximizing control performance. In recent years, the control and real-time communities have shown a renewed interest on deriving novel methods for efficient implementation of real-time control systems. According to a taxonomy analysis on these methods [3], two main tendencies have been identified: feedback scheduling (FS) and event-based scheduling (ES). The main difference between them is that feedback scheduling look at the problem of optimizing control performance by fully exploiting the available resources. On the other hand, event-based scheduling considers the minimization in the use of resource while providing an acceptable control performance or at least ensuring system stability.

Feedback scheduling assumes periodic sampling, this means that the controller is implemented by tasks that are activated periodically. So in essence the implementation of feedback scheduling on real-time control systems, can be analyzed as the use of performance optimization methods in time-triggered control systems. On contrary, in event-based scheduling, there are event conditions (usually asynchronous) that determine the occurrence of discrete events that trigger control updates. In this case event-based scheduling can be considered as the design of resource optimization methods for event-driven control systems.

This report presents an evaluation analysis of representative FS and ES methods used in real-time control systems, considering two main criteria:

the control performance and the resource optimization. The contribution of this work is to provide valuable elements of analysis, through the results obtained from the evaluation, in order to discuss the key elements of each optimization approach considering selected FS and ES methods.

The rest of the report is structured as follows. Section 2 describes the characteristics of FS and ES approaches. Section 3 summarizes the evaluated methods. Section 4 details the simulation set-up and the customization of the evaluated methods. Section 5 presents the simulation results and Section 6 concludes the report.

2 Preliminaries on FS and ES

In this work we consider the case where there is a control system with several control loops competing for a common resource: processor capacity. Each control loop contains a controller which executes a control task τ_i , there is one plant for each control task, and there are several control tasks being executed in the same processor. Each control loop can be described by the evolution of the plant dynamics over time. Each plant can be modelled using the following continuous time state-space representation:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

where x represents the system state, u is the input, y denotes the system output, A, B and C describes the evolution of the system. Now, given the feedback matrix L , let

$$u_i = Lx(a_i) = Lx_i \tag{2}$$

be the control updates computed from a sample at time a_i by a discrete feedback controller.

The control updates provided by the controller to the plant, can be triggered on a time basis or considering the system evolution. Therefore, two approaches can be used in the design of the control system.

- In a time-triggered control system the activation of the control tasks is conducted according to a discretization in the time domain, providing equidistant activation instants, i.e. constant sampling period. In the periodic sampling we have $a_{i+1} = a_i + T$, where T is the period of the controller.

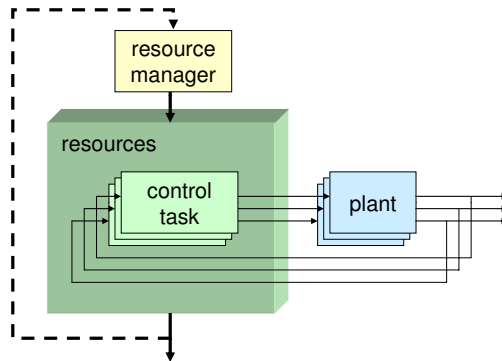


Figure 1: Feedback scheduling scheme

- In an event-driven control system the activation of control tasks depends on the system trajectory according to the space domain. If the trajectory crosses a defined threshold (or boundary) a control action is activated. In this case the activation instants are not constant and neither equidistant. In event-driven systems, control task activations occur at a sequenced $\{a_i\}$ that are not necessarily periodic.

2.1 Feedback Scheduling

Feedback scheduling methods are implemented on time-triggered control systems and refer to the problem of sampling period selection for real-time control tasks that compete for limited computing resources. Its goal is to optimize the aggregated control performance achieved by all tasks by using the available resources.

The logics of the standard feedback scheduling architecture are shown in Fig. 1. Several control tasks in charge of controlling plants compete for limited resources (processor capacity). The resource manager implements the solution provided by each FS method, that is, it dictates how resources are assigned to each control task in order to optimize overall control performance.

Considering a control system where several control loops share the same processor, now let the number of tasks on the system be n , and let each task execute with a sampling interval h_i (task period) and have the execution time C_i . Each task is associated with a cost function $J_i(h_i)$, which gives the control cost as a function on sampling interval h_i . The following optimization

problem is formulated:

$$\text{minimize } J = \sum_{i=1}^n J_i(h_i), \quad (3)$$

subject to the constraint

$$\sum_{i=1}^n \frac{C_i}{h_i} \leq U_{ref} \quad (4)$$

where U_{ref} is the desired utilization level.

The optimization problem is constrained by two key aspects. The set of optimal sampling periods must guarantee closed loop stability and task set schedulability. Stability is either guaranteed by the formulation of the optimization problem, or it is not explicitly imposed in the formulation but analyzed after solving the optimization problem. Task set schedulability is often imposed by resource utilization tests.

2.2 Event-based Scheduling

Event-based scheduling methods are implemented on event-driven control systems. These control systems provide interesting benefits like reducing resource utilization while providing similar control performance to the case of periodic controllers [12]. In event-driven control systems, the controller is activated upon some condition on the system status and not periodically. The condition called even-condition or execution rule, mandates to take a new control action when the system state variables (or measured signals) have deviated sufficiently from the set-point. Figure 2, presents a graphical representation of an event-based system, the x-y plane is defined by two state-variables (or two measured signals), as it can be observed there is a boundary which allows the system to move without the need of a control action. The event-condition or execution rule is activated when the variables crosses or touch the boundary, therefore the controller only acts when needed. Unfortunately event-driven control lacks of a mature system theory that prevent, for example, to estimate computational load required by these controllers. However recent works on event-driven ([11],[8],[9],[13],[14]) control approaches have focused not only on control aspects but also on real-time aspects.

The event condition specifies the rules that triggers the execution controller jobs. Generally speaking, the execution rule ensures that the system state does not deviate significantly from the expected value, i.e. a given

Approach	Trigger	When	Dynamics	Exec. Rule
Static approach	TT			
Off-line FS	TT	Off		
On-line RA FS	TT	On	kernel	
On-line RCA FS-Inst.	TT	On	kernel/plant	
On-line RCA FS-FH	TT	On	kernel/plant	
ES SS ER	ET	On	kernel/plant	state
ES MS ER	ET	On	kernel/plant	meas. state
ES MS ER-Opt.	ET	On	kernel/plant	meas. state

Table 1: Key features of evaluated FS and ES methods.

where $0 < \eta \leq 1$ specify the relative size of the boundaries, in this case the boundary shape is a circle. On the other hand, if the event condition, which triggers a control task, is a function of the measured state, then its definition is as follows:

$$e_k(t)^T M e_k(t) = \eta x_k^T M x_k, \quad (8)$$

where $0 < \eta \leq 1$ specify the relative size of the boundaries, and M defines the shape of the boundaries. On [9], M is obtained from the system dynamics creating an ellipse shape.

3 Selected Methods

The following subsections present the evaluated methods, and characterizes them according to the following aspects. Although not exhaustive, they are representative of the existing research (see [3] for more details).

- Triggering paradigm: time-triggered (TT) for feedback scheduling or event-triggered (ET) for event-based scheduling.
- When to solve the optimization problem: off-line or on-line.
- Which kind of dynamics, is accounted for in the optimization problem: resource (kernel) and/or plant.
- Execution rule: event condition can be defined as a function of the system state or the measured state (applies only for ES methods).

3.1 Static Approach

This is the only approach that does not belong to the class of feedback scheduling nor event based scheduling methods but it is here included for comparative purposes. It implements the traditional approach to real-time implementation of computer controlled systems. That is, each control task is assigned off-line an *arbitrary* sampling period (time-triggered) selected according to well established procedures [16], taking also into account task set utilization.

3.2 Off-line FS (feedback scheduling)

The off-line FS is represented by the work by [4] which can be considered one of the seminal papers on sampling period selection subject to control performance optimization for real-time control systems. An off-line optimization is performed in order to reduce control cost, once the optimal periods are calculated, the control tasks are scheduled under EDF (Earliest Deadline First).

3.3 On-line RA (resource aware) FS (feedback scheduling)

The method presented by [5] is the first one that uses the term *feedback scheduler*. The key aspect is to on-line adjust sampling periods considering the dynamics of the processor load. Looking at the outer loop of Fig. 1, the resource manager is the feedback scheduler that, having available the system workload from the real-time kernel, i.e. *resource aware* (RA) and given a utilization set-point, keeps the desired utilization by modifying workload via on-line sampling period selection, while optimizing the total control performance.

3.4 On-line RCA (resource and control aware) FS (feedback scheduling)

A step further is to optimize total control performance by on-line adjusting sampling periods according to both kernel workload and plants' dynamics, i.e., *resource and control aware* (RCA). The intuitive idea behind this kind of approaches is to provide more processing capacity to control tasks whose plants are experiencing severe transients due to e.g. perturbations or noise.

Representative methods are given for example by [6] and [7]. Both approaches assume that the feedback scheduler has available the state of all

controlled plants when solving the optimization procedure. Depending on how states are treated in the optimization, two flavors can be distinguished:

- *Instantaneous* (Inst.): the current state is the only information of the plants that is considered in the optimization procedure. This is the approach adopted by [6].
- *Finite horizon* (FH): the current state is the initial condition for predicting over a finite horizon the future plants' dynamics. This is the approach adopted by [7].

3.5 ES (event-based scheduling) SS (system state) ER (execution rule)

This method is based on event-driven control and is represented by the work by [8], where a self-triggered control approach is implemented to ensure robust control performance. The key idea is that at each job activation, the software tasks select themselves the next job release time according to a specific execution rule. In this approach the event condition or execution rule is defined as a function of the system state.

3.6 ES (event-based scheduling) MS (measured state) ER (execution rule)

This method is based on event-driven control and it is represented by the work by [9]. Also a self-triggered approach is implemented. In this case the event condition or execution rule is defined as a function of the measured state. An enhancement of this approach can be obtained from a model proposed by [10] where an optimization algorithm can be executed to obtain optimal parameters according to the plant dynamics and the cost function.

3.7 Summary of key features

Table 1 presents a summary of the key features of each method under evaluation. Columns refer to a) the triggering paradigm (time triggered-TT or event triggered ET), b) when the optimization is carried out (off-line or on-line), c) what dynamics (kernel workload and/or plant dynamics) are included in the optimization when it is performed on-line, and d) in the case of self-triggered methods (event-based), what type of variables (system state or measured state) are considered to defined the event-boundaries (execution rule).

4 Simulation

The evaluation presented in this work focuses on control performance and resource utilization. A simple but detailed simulation set-up has been designed in order to identify the key features for each evaluated method. Simulations have been carried out with the *Truetime* simulator [17] to implement the multitasking processor together with each FS and ES strategy. The performance analysis targets to measure the impact of each strategy on the controlled plant dynamics.

4.1 Controlled plants

Three identical ball and beam in the form of double integrator were used as controlled plants. Each ball and beam is controlled by a single control task. The three control tasks concurrently execute on a real-time kernel. The ball and beam state space description is

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad (9)$$

where the system output is x_1 , which corresponds to the ball position. The control objective is to keep the ball position at zero coordinate.

For the simulation purposes, it is assumed that the initial condition (ball position) of the three plants has a random non-zero value. When a task is activated the control actions move the plant state to the reference value (zero coordinate). The activation of the control task for each plant is produced after a random time-delay, therefore each control loop is activated at different instants during the simulation period. In this way the values of the initial condition and the starting time of each control loop are randomly generated. However it is important to mention that the same random values are used for every evaluated method, in order to have a fair comparison. The duration of each simulation period is 50 seconds, and a total of 30 different simulation runs were conducted for each evaluated method. The reported results corresponds to the average values of all the simulation runs.

4.2 Evaluation metrics

The two main criteria in order to evaluate the selected methods are: control performance and resource optimization. Therefore, two evaluation metrics have been defined for this purpose.

Control performance is measured during each simulation period (t_{sim}) using a continuous standard quadratic cost function

$$J_{control} = \int_0^{t_{sim}} [x^T(t)Qx(t) + u^T(t)Ru(t)] dt. \quad (10)$$

where the weighting matrices are specified as

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad \text{and} \quad R = 10.$$

For the simulation purposes, it is assumed that plant states are available and therefore there is no need for observers.

Resource utilization is measured as a percentage of use of the processor during each simulation period (t_{sim}). So the resource utilization is defined as:

$$J_{resource} = \frac{1}{t_{sim}} \sum_{i=1}^n E_i \quad (11)$$

where n is the number of tasks sharing the same processor (specifically during these simulations $n = 3$, corresponding to tasks τ_1, τ_2, τ_3), and E corresponds to the total processor time assigned to a specific control task during the simulation period. Therefore, for each specific control task τ_i , the total processor time assigned is specify by:

$$E = \sum_{j=1}^m C_j \quad (12)$$

where m is the number of times that a specific control task is invoked during the simulation period, and C corresponds to the task execution time (a fixed execution time of $100ms$ is assumed for every control task in all the cases.)

4.3 Simulation model

The basic *Truetime* simulation model is shown on figure 3. The model is integrated by three control loops: three plants are being controlled by three control tasks, that are being executed on the real-time kernel processor. Control performance is measured directly from the plant states using the equation 10. Resource utilization is measured directly from the processor using equation 11. The control tasks can be executed either on a time-triggered basis or on a event-triggered basis, depending on the optimization method.

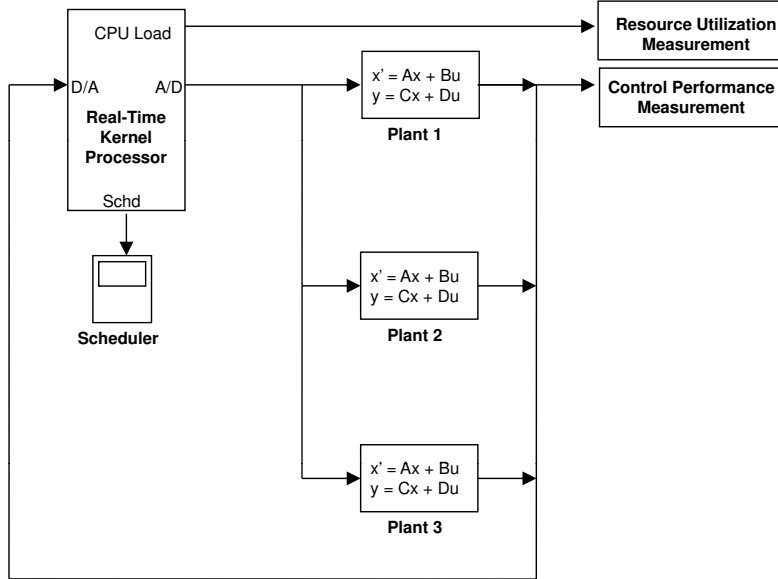


Figure 3: Simulation model

4.4 Tasks and periods for each method

In order to evaluate, not only the control performance, but also the resource utilization, two set of task periods were defined for each method. For the first set, it is considered that there are no restrictions regarding the use of the processor, i.e. U_{ref} for FS methods always equals to 1, and a small η is selected for ES methods in order to have the events executed more frequently. In the second set, it is assumed that the processor cannot be used completely so there is resource restrictions, then U_{ref} equals 0.6 for FS methods and a larger η is used for the ES methods. This first set is used to evaluate the system with a high processor load, and the second one to evaluate it with a low processor load. The periods of the three control tasks are allowed to take values within 150ms and 600ms for the high processor load case, and values of 400 to 600 for the low processor load set. In any case, the execution time of each task is assumed constant and has a value of 100ms.

The specific tasks periods for each of the evaluated methods are summarized in table 2 for high processor load and in and table 3 for low processor load. The details for both cases are described next:

Approach	Type	Task 1	Task 2	Task 3
Static		0.1500	0.6000	0.6000
Off-line FS	FS	0.3000	0.3000	0.3000
On-line RA FS	FS	0.3000	0.3000	0.3000
On-line RCA FS-Inst.	FS	0.1500	0.6000	0.6000
On-line RCA FS-FH	FS	0.1500	0.6000	0.6000
ES SS ER	ES	0.2409	0.2409	0.2409
ES MS ER	ES	0.4212	0.4212	0.4212
ES MS ER Opt.	ES	0.3139	0.3139	0.3139

Table 2: Tasks sampling periods (seconds) for the high processor load case.

Approach	Type	Task 1	Task 2	Task 3
Static		0.4000	0.6000	0.6000
Off-line FS	FS	0.5000	0.5000	0.5000
On-line RA FS	FS	0.5000	0.5000	0.5000
On-line RCA FS-Inst.	FS	0.4000	0.6000	0.6000
On-line RCA FS-FH	FS	0.4000	0.6000	0.6000
ES SS ER	ES	0.4916	0.4916	0.4916
ES MS ER	ES	0.5508	0.5508	0.5508
ES MS ER Opt.	ES	0.5604	0.5604	0.5604

Table 3: Tasks sampling periods (seconds) for the low processor load case.

- Static approach: periods for each task are heuristically selected and the corresponding controllers designed before run-time.
- Off-line FS [4]: since the three plants are equal, the off-line optimization procedure mandates to execute each task with the same period.
- On-line RA FS [5]: the initial periods for the three tasks are the same as in the static approach, and the on-line recursive optimization routing stabilizes the three periods. From the periods initial values to the final values, the controller that is used is re-designed at run-time according to the current period that applies. Tables 2 and 3 show the stabilized periods.
- On-line RCA FS - Inst. [6]: the final outcome of the method mandates to consider at run-time only two periods. Tasks (and controller' gains) switch between these two periods whenever the plant with highest error changes. Tables 2 and 3 show the sampling period values considering

that task 1 has the largest instantaneous error.

- On-line RCA FS - FH [7]: this method mandates to switch periods at run-time continuously within the specified range according to the optimization procedure. Switches of tasks periods (and controllers' gains) occur every 150ms, which is the period of the so-called feedback scheduler. Tables 2 and 3 show the sampling period values considering that task 1 has the largest finite-horizon error.
- ES SS ER [8]: since this is an event-based method, the control task are not executed based on periodic sampling periods. Instead an event condition or execution rule defines when a control task is self-triggered. In this case the event condition represents a function of the system state (see equation 7). In an event-based system η can be used to adjust the processor load. For a high load η was selected to provide a 90% processor load approximately, and for a low load η was selected to obtain close to 55% processor load. Tables 2 and 3 show average interval values for high and load processor load respectively, in both cases the average values are contained within the range established by the static approach.
- ES MS ER [9]:in this event-based method, the execution rule, that triggers a control task, has been defined as a function of the measured state (see equation 8). In this method the shape of the boundaries is an ellipse and is obtained from the system dynamics. Tables 2 and 3 show average interval values for high and load processor load respectively, with the same considerations as the ES SS ER method.
- ES MS ER - Optimized [10]: in this event-based method, the execution rule is also defined as a function of the measured state (see equation 8). The difference of this method with the one proposed by [9], consists that the M parameter that defines the boundaries shape is not based on the system dynamics, but instead is obtained from an iterative algorithm that looks for the optima M that provides the minimum cost given an specific η . Tables 2 and 3 show average interval values for high and load processor load respectively, with the same considerations as the ES SS ER method.

4.5 Controller design

For the time-triggered FS methods, control is achieved by a controller designed to place the continuous closed loop poles at $s_{1,2} = -0.866 \pm 0.5i$, these

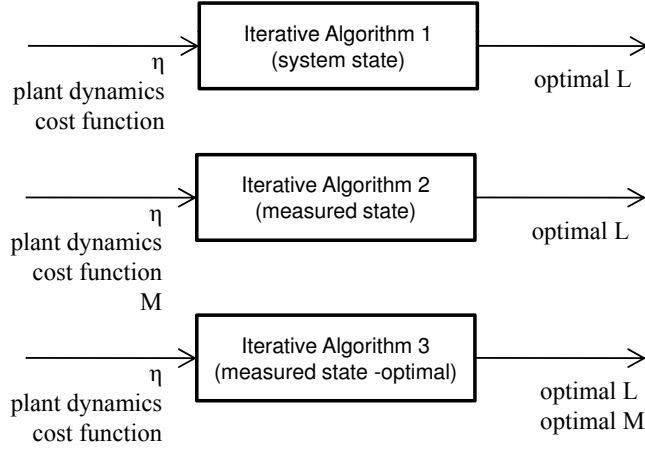


Figure 4: Iterative algorithm parameters for ES methods

poles were obtained using optimal control techniques in order to maximize control performance (see equation (10)). Depending on the sampling periods that applies, the continuous closed loop poles are mapped into the discrete time domain, and then the discrete-time controller is obtained by standard pole placement.

However, the event-based methods cannot use the same optimal techniques used by the FS methods, since there is not a periodical sampling period. Even more, current event-based control theory does not provide a solid analytical technique to obtain an optimal feedback matrix L given a cost function. So in order to obtain an optimal L for the event-based methods, an iterative algorithm was implemented. Given a specific η , and according with the plant dynamics and a cost function (10), the algorithm searches for a feedback matrix L value which provides the minimum control cost (optimal). Figure 4 shows the input parameters and the output values for the algorithm, the algorithm 1 is used for the ES system state ER method, the algorithm 2 is used for the ES measured state ER method and the algorithm 3 for the ES measured state ER - optimized method. Notice that in algorithm 3 the boundary shape matrix M from equation (8) is included in the optimization process not as an input but as an output value. Algorithm 2 uses M as an input since is obtained from the system dynamics.

5 Simulation Results

This section summarizes the results obtained from the simulation of the different methods. First, we present how the feedback matrix L from equation 2 is obtained for the case of the event-based methods, since there is not an analytic way to obtain these values. Then, the control performance and resource utilization results for each ES and FS method are presented and discussed.

5.1 Results for event-based methods

In order to design an optimal controller for event-based methods, an optimization algorithm is required to be executed before the performance simulation, the values obtained from this algorithm are used later in the simulation. The execution of the algorithm, considers just one control loop, and it searches for an optimal controller gain L given a group of specific input variables.

The results obtained from the iterative optimization algorithms can be used to compare the control performance of the ES methods with an optimal time-triggered approach using constant sampling periods. Figure 5 shows the control performance of the three ES methods, the x-axis represents the sampling period, since ES methods do not have a constant sampling period the average value of the varying sampling instants is used instead. The control performance defined by (10) represents a cost function so a lower value means better performance. As expected, for any method, a shorter sampling periods produce a better performance, and when the period is incremented the control performance degrades.

Among the ES methods, the one that produces a better control performance is the method based on a measured state execution rule with an optimized M parameter (ES MS ER - Optimized). Therefore, the shape boundaries (represented by M) in the event-based control has a direct influence in the control performance, so if M is considered as a parameter that can be used in an optimization process for the controller design, then the control performance can be improved considerably. Now comparing the ES MS ER-Optimized method with the optimal time-trigger, it can be observed that for small sampling periods the performance is very similar, but for larger sampling periods this particular ES method has better results. Based on this, it can be expected that the ES method will perform better under specific circumstances, i.e. low processor load, due its ability to optimize the use of resources.

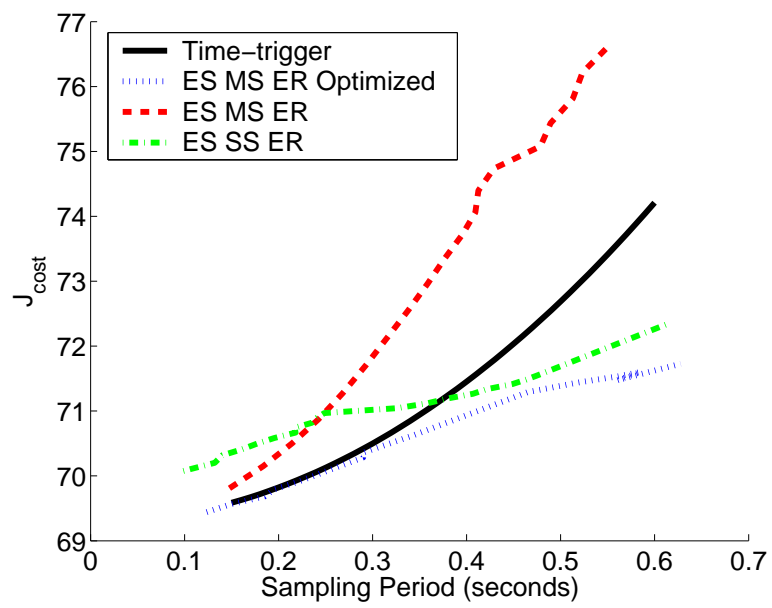


Figure 5: Control performance for optimized ES and time-triggered controllers

5.2 Control performance and resource utilization results

The simulation results are presented in two parts. First the control performance and resource utilization resource considering a high load processor (85%-95%) is presented, then the same type of results are provided for the case of a low load processor (50%-60%). It is important to highlight that this simulation assumes that there are not timing variations caused by the task scheduling, i.e. jitters. As demonstrated by [15], the jitters degrade control performance and may hide the true performance that can be achieved by the different FS and ES methods. So, for the simulation purposes, the degrading effects caused by the jitters were removed completely.

Approach	Type	Control Performance	Resource Utilization
Static		247.5021	0.94
Off-line FS	FS	242.9536	0.91
On-line RA FS	FS	242.9534	0.91
On-line RCA FS-Inst.	FS	241.3384	0.94
On-line RCA FS-FH	FS	241.9337	0.96
ES SS ER	ES	242.4432	0.86
ES MS ER	ES	243.4235	0.88
ES MS ER Opt.	ES	239.6280	0.86

Table 4: Control performance and resource utilization results for a high load processor

Table 4 summarizes the simulation results of the FS and ES methods, considering a high load processor. Analyzing the control performance, we observe that the static method provides the worst performance, as expected. Then considering only the FS methods, the on-line algorithms provide better performance. For the ES methods the optimized one can achieved the best performance. Comparing FS versus ES methods, it can be noticed that FS methods perform better than ES, with the exception of the ES MS ER-Optimized which has the best control performance.

Table 5 summarizes the simulation results of the FS and ES methods, considering a low load processor. This scenario assumes that there are restrictions in the use of the processor, so an efficient use of resources is critical. First, it can be observed that every method suffer a performance degradation due the resource restriction. However, the performance degradation is considerably smaller in the ES methods, since all the three ES methods perform better than any FS method, so it can be stated that event-based control provides a more efficient way to optimize resource utilization compared with

Approach	Type	Control Performance	Resource Utilization
Static		248.2773	0.53
Off-line FS	FS	246.7824	0.54
On-line RA FS	FS	246.7824	0.54
On-line RCA FS-Inst.	FS	244.6351	0.53
On-line RCA FS-FH	FS	245.7684	0.58
ES SS ER	ES	243.7888	0.59
ES MS ER	ES	243.4595	0.58
ES MS ER Opt.	ES	243.0606	0.50

Table 5: Control performance and resource utilization results for a low load processor

time-trigger control.

It is important to highlight that these results are not intended to be definitive in the sense that always one method will perform better than other. But instead, these results can be taken as an indicator of the potential of some approaches and to provide valid information to discuss the benefits and drawback of each tendency under specific circumstances.

6 Conclusions

Feedback scheduling and event-based scheduling of real-time control tasks has received increased attention in the real-time and control research communities in the last years. This paper has presented a control performance and resource utilization analysis that reveals three key aspects. First, it shows that FS and ES have the ability to improve control performance with respect to the standard approach to real-time implementation of control loops. Second, ES provides better resource utilization compared to FS methods. And third, among ES methods the one that optimizes the controller considering the boundaries provides the best performance.

Future work will focus on the performance of experimental evaluation considering the jitters impact. Also, in the control area there are opportunities in the design of an analytical solution for the optimization of event-based methods.

References

- [1] G. Buttazzo, “Research trends in real-time computing for embedded systems.” ACM SIGBED Review, volume 3, number 3, 2006.
- [2] K.-E. Årzén, K.-E. Cervin, J. Eker, and L. Sha, “An introduction to control and scheduling co-design.” 39th IEEE Conference on Decision and Control, 2000.
- [3] C. Lozoya, M. Velasco, and P. Martí, “A 10-year taxonomy on prior work on sampling period selection for resource-constrained real-time control systems.” Work-in-Progress session of the 19th Euromicro Conference on Real-Time Systems, 2007.
- [4] D. Seto, J.P. Lehoczky, L. Sha, and K. Shin, “On task schedulability in real-time control systems.” 17th IEEE Real-Time Systems Symposium, 1996.
- [5] J. Eker, P. Hagander, and K.-E. Årzén, “A feedback scheduler for real-time control tasks.” Control Engineering Practice, volume 8, number 12, 2000.
- [6] P. Martí, C. Lin, S. Brandt, M. Velasco and J.M. Fuertes, “Optimal state feedback based resource allocation for resource-constrained control tasks.” 25th IEEE Real-Time Systems Symposium, 2004.
- [7] R. Castañé, P. Martí, M. Velasco, A. Cervin, and D. Henriksson, “Resource management for control tasks based on the transient dynamics of closed-loop systems.” 18th Euromicro Conf. on Real-Time Systems, 2006.
- [8] P. Tabuada, “Event-triggered real-time scheduling of stabilizing control tasks.” IEEE Transactions on Automatic Control, 2007.
- [9] M. Lemmon, T. Chantem, X.S. Hu, and M. Zyskowski, “On self-triggered full information h-infinity controllers.” Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control, 2007.
- [10] M. Velasco, P. Martí and E. Bini, “Control-driven tasks: modelling and analysis.” 29th IEEE Real-Time Systems Symposium (RTSS08), 2008.

- [11] M. Velasco, P. Martí and C. Lozoya, “On the timing of discrete events in event driven control systems.” 11th International Conference on Hybrid Systems: Computation and Control, 2008.
- [12] W. Hemmels, J. Sandee, and P. van den Bosch, “Analysis of event-driven controllers for linear systems.” International Journal of Control, 2008.
- [13] X. Wang, and M. Lemmon, “Self-triggered feedback control systems with finite gain L2 stability.” 17th IFAC World Congress, 2008.
- [14] A. Anta, and P. Tabuada, “Self-triggered stabilization of homogeneous control systems.” Proceedings of the 2008 American Control Conference, 2008.
- [15] C. Lozoya, M. Velasco, P. Martí and J.M. Fuertes, “Control performance evaluation of selected methods of feedback scheduling of real-time control tasks.” 17th IFAC World Congress, 2008.
- [16] Åström, K.J. and B. Wittenmark, *Computer controlled systems*, Prentice Hall, 1997.
- [17] D. Henriksson, A. Cervin, and K.-E. Årzén, “TrueTime: Simulation of control loops under shared computer resources.” *15th IFAC World Congress*, 2002.