Control Tasks Delay Reduction under Static and Dynamic Scheduling Policies

P. Balbastre, I. Ripoll, A. Crespo
Dept of Computer Engineering, Universidad Politécnica de Valencia
{patricia, iripoll, alfons}@disca.upv.es

Abstract.
Industrial application of digital control requires the synergy between well designed control algorithms and carefully implemented control systems. The control performances can be strongly influenced depending on the data acquisition and control action delays. The paper shows how to evaluate these delays (jitter) under static or dynamic scheduling policies. An evaluation of several set of tasks executed under both scheduling policies is analysed and compared. The results allow us to determine the goodness of both algorithms with respect to the delays due to scheduling. While worst case response time in static scheduling can be easily determined, under EDF scheduling the result is not trivial. A method to determine the worst case response time under EDF scheduling is proposed. The measurement of these delays can be drastically reduced with a task decomposition. This decomposition is studied and evaluated from the delays point of view. The reduction of the data acquisition interval (DAI) and control action interval (CAI) under both scheduling policies can be considered in the control design phase in order to properly adjust the control algorithm.

1. Introduction

In classical digital control system design, all the sampled data signals are assumed to be regularly, synchronously, and equally time spaced sampled, being T the sampling period. But this is not the case in most industrial applications. There are many reasons leading to a different sampling scheme. Another interesting and common situation is the distributed control where multiloop control systems are running in parallel. In this case, the sampling rates could be different and even not synchronized, but the control algorithms are cooperating and exchanging information. This situation also raises the use of multirate control schemes, where the selection of the sampling rate is based on its suitability to control each controlled variable [5]. This imposes the use of discrete time (DT) models with a different sampling period even for the same process.

While the process of control design is focused on obtaining the regulator, subsequently translated into an algorithm, the software design is focused on producing pieces of software that will be executed concurrently under the supervision of a scheduler. A real-time control application is usually structured as a set of periodic and sporadic tasks which interact with the environment by reading the sensor values and sending actions to external devices. The software designer has to ensure that all the tasks meet their deadlines, i.e., the system is schedulable. However, there exists a gap between the controller design and the real-time system implementation which are clearly developed in separate ways. In [12],[13], a method to integrate both phases is proposed. In a non schedulable real-time implementation, they propose to analyze the dynamic of the system adjusting the task periods in order to obtain a schedulable set of tasks.

On the other hand, current research efforts [2], [10], [11] are in the way of making dynamic scheduling a robust and suitable solution to be integrated in real-time operating systems and be used in control systems.

The control designer knows that, although the sensor data acquisition and the process action updating are assumed to be instantaneous and synchronized, they are not. Also, they change as the time goes on. Some of these issues can be known at the design stage and should be incorporated into the control algorithm. But some others are time varying and dependent on the working conditions. One of these aspects is related to the multitasking system and the scheduling policy. When a set of tasks is scheduled using a fixed-priority scheduler, lower priority tasks are delayed (pre-empted) by the execution of higher priority task. There are some works related to the evaluation and reduction of the jitter produced by the delayed of events arriving to the system or high priority task interferences. [7],[8]. To reduce the effects of these disturbances, some kind of prediction and filtering should be incorporated. Among these delays, the control algorithm can take into account those due to the acquisition system and the expected time to compute the
own control output, because they are known or can be accurately estimated.

Considering the output jitter, other work [4] proposes a method to minimize the output jitter. In this approach the scheduling algorithm is the earliest deadline first (EDF) and attempts to reduce the output jitter between two consecutive executions. This approach does not guarantee the minimization all over the periods. In [3] a model for periodic tasks is proposed, that explicitly considers jitter, defined as the uncertainty in the arrival times of individual activation of tasks.

In [6] a decomposition method of each task into three tasks is used to reduced the jitter variation, defining a Data Acquisition Interval (DAI) and Control Action Interval (CAI) under priority based scheduling. The scheduling analysis of this new set of tasks is proposed. In [1] a methodology to find the appropriate scheduling to minimize the degrading of the control performances is proposed. The main concepts considered in that paper come from the two complementary design aspects. The control effort and the effect of delays in performance degrading arise in the control design. On the other hand, the CAI, its relation with the task scheme and scheduling parameters assignment (priority and offsets), and the influence of the scheduling are considered in the real time system implementation.

In this paper, we compare the delays introduced by the scheduler in two cases: fixed-priority (DM) and dynamic-priority (EDF) scheduling. We extend the decomposition method applied to static scheduling in [6] to tasks executed under EDF scheduling, and we compare the results obtained when the decomposed tasks are executed under both scheduling policies. While fixed-priority scheduling analysis provides the worst response time of a task during the critical instant and it corresponds to the maximum delay that a task will suffer in the hyperperiod, the corresponding delay when a set of tasks is executed under EDF is not trivial. A method to calculate the worst case response time of each task in the system is proposed in this paper.

The paper is organized as follows. In section 2 the problem of the control tasks is discussed, and the main aspects of the problem are pointed out. In Section 3 and 4, a task decomposition for control tasks to minimise the DAI and CAI in static and dynamic scheduling is proposed. Section 5 shows the results obtained by simulation considering several sets of tasks. One of the problems pointed out in the EDF scheduling algorithm is the worst case execution time determination, in this aspect, section 6, details a modification of the method proposed in [15]. Finally, some conclusions and future works are outlined in the last section. The main contributions of the paper are presented in sections 4, 5 and 6, where a partition task method as well as an algorithm to minimize the CAI associated to one, some or all the tasks, are proposed.

2. Control Tasks

Control applications require defining several parallel activities to model the environment. Nowadays, more and more applications require complex computation and the use of complex algorithms that can compromise the response time of the system. The activities involving a control loop task can be structured in the following parts: i) Data acquisition: data from external sensor is acquired; ii) Computation of the control action: it corresponds to the regulator computation and should be executed as soon as possible. At the end of this part, the basic control action is ready to be sent to the actuators; iii) Output the control action: must be done either as soon as possible or within a fixed time interval.

The control computation can be divided into a main part (mandatory) and one or more optional parts. The mandatory part gives a first solution to the control problem. Optional parts try to improve the solution. Nevertheless, in this paper, the optional parts will not be considered.

When several periodic control tasks are executed, the task activation can suffer delays due to the scheduling policy and the task priorities. Depending on the computation times and periods, different tasks can have higher or lower delays in the starting or finishing instants of each activation. This situation can add additional delays to the input jitter due to the data acquisition. A simple example will help to show the delays effects in the control performance. Consider two identical simple processes (P1 and P2) with the same transfer functions $G(s) = \frac{100}{s+6}$. The same controllers are required for both processes. Tasks T1 and T2 control each process and T1 has higher priority than T2. The control response considering the delay effects due to scheduling are plotted in Figure 1.

Considering only the effects due to the scheduling, the Figure 1 shows the worst control response of the task with lower priority. In order to measure the effects of the scheduling, the following definitions can be made.

![Figure 1 Closed-loop response of P1 and P2](image)

The release time of the task 1 in its k activation is
denoted by \( trel_i \). If \( n \) is the number of task activation within the hyperperiod, \( WCRT_i \) is the maximum value of \( WCAT_i = \max \{ trel_i \} \) and \( BCRT_i \) is the minimum value expressed as \( BCAT_i = \min \{ trel_i \} \). The time interval \( [BCRT_i, WCRT_i] \) includes all the release time of a task \( T_i \). In control systems, the BCRT corresponds to the minimum delay the control action is sent by the task \( T_i \) while the WCRT is the maximum delay. The CAI of a task \( T_i \) is denoted by

\[
CAI_i = 100 \times \frac{WCRT_i - BCRT_i}{P_i}
\]

The CAI expresses the percentage of variation of control action of a task relative to its period. This measure gives information about the delay variation will suffer the control action of a task and allows to compare the performances of a control system.

On the other hand, let \( tact_i^k \) be the time that the \( k \)th invocation of task \( i \) commences its execution. The \( WCAT_i \) denotes the maximum value of \( WCAT_i = \max \{ tact_i^k \} \) and \( BCAT_i \) is the minimum value expressed as \( BCAT_i = \min \{ tact_i^k \} \).

The time interval \( [BCAT_i, WCAT_i] \) includes all the activation time of a task \( T_i \). In a similar way, the DAI of a task \( T_i \) is denoted by

\[
DAI_i = 100 \times \frac{WCAT_i - BCAT_i}{P_i}
\]

The DAI expresses the percentage of variation of sensor acquisition relative to its period. Intuitively, if the task \( T_i \) has the higher priority and it is executed under a priority based scheduling, its DAI will be \( 0\% \) since in all the activation will start at the beginning of its period. The CAI of the same task will be also \( 0\% \), and the WCRT = BCRT = WCAT.

3. CAI minimization in DM scheduling.

In order to reduce the jitter terms, a task decomposition criteria and its schedulability analysis is proposed in [6] for set of tasks executed under priority based scheduling policy. Figure 2 shows the task decomposition of a task \( T_1 \) in three tasks (initial \( T_{ij} \), main \( T_{im} \) and final \( T_{if} \)). The priorities assigned to these tasks are also shown in the figure. This task formulation ensures a fixed delay of the control action and a small variable delay range for each task is accomplished. This approach dramatically reduces the CAI and allows a better algorithm behaviour.

In [6] a schedulability test for the task partitioning is proposed. In the test the offset of the final task is adjusted to minimize the absolute delay as well as the CAI.

4. CAI minimization in EDF scheduling

Given a set of task to be scheduled in EDF scheduling, each task candidates to a DAI and CAI reduction is split according to the following criteria (figure 3): a new deadline is calculated for initial and final parts, and an offset is applied to final part.

\[
D_{if} = WCRT_i^{if} - \text{offset}
\]

The algorithm iterates until the WCRT of final parts are equal to their respective deadlines. When the final parts deadlines have been computed, the same iterative method is applied to obtain the initial parts deadlines.

The deadline assigned to initial parts is shorter than the corresponding mandatory part deadline. Therefore, initial parts will be executed before mandatory ones when executed by a deadline driven scheduler. On the other hand, final parts have to be delayed to guarantee a desirable behaviour. This can be achieved by adding an offset to the final part. This offset is the latest time that the final part can start its execution:

\[
\text{offset} = WCRT_i^{nf} - C_{if}
\]

The earlier time a final part can start its execution is the offset of the final part, and the latest time that a final part can finish without miss its deadline is given by its
deadline relative to the task activation (CAI = 100*D_j).

The earlier time an initial part can start its execution is given by:

$$BCAT_i = \sum_{j=1}^{N} [C_j \cdot e(D_i < D_j)]$$

where N is the number of tasks and the e function returns a value of 1 if the condition in brackets is true, otherwise it returns 0. The DAI is calculated as the difference between the obtained D_j and the BCAT_i:

$$DAI_i = 100* (D_j - BCAT_i)$$

To guarantee the system schedulability the associated offset to the final part has to be after the mandatory termination time. That is, we have to assure that when a final part will be ready to execute, the mandatory part was already finished. Therefore, the task set with offsets is schedulable since the task set without offsets is also schedulable. Figure 4 gives the pseudo-code for CAI and DAI minimization.

```
begin
  Compute_WCRT(T) //Computation of final parts deadlines
  while (3j / Tj).Deadline != Tj.WCRT do
    for i = 1 to N Tasks do
      Tj.Deadline = Tj.WCRT; od;
    od;
  Compute_WCRT(T); // Computation of initial parts deadlines
  while (3i / Ti).Deadline != Ti.WCRT do
    for i = 1 to N Tasks do
      Ti.Deadline = Ti.WCRT; od;
    Compute_WCRT(T); //Computation of the offsets for the final parts
    for i = 1 to N Tasks do
      Ti.Offset = Tm.WCRT - Ti; od;
  od;
end
```

Figure 4. CAI minimization algorithm

The proposed algorithm intends firstly to minimize the variance of the final parts, and last to minimize the execution interval of the initial parts. We could have assigned the deadlines of the initial and final parts at the same time, but this would not minimize the variance of the final parts at the maximum rate. However, this is a possibility if a more homogeneous reduction is required. Another possibility is to minimize by tasks, that is, firstly to reduce the variance of the higher priority tasks. This could be applied when a more reduction is required for some tasks of the set. In [1] it is analyzed the delay effects in the control performance degrading. Two main parameters are determinant: the CAI and the control effort of the regulator. Control systems are more sensible to delay variations when the control effort is higher. Applied to this case, tasks with higher control effort would reduce CAI in a higher rate than other tasks in the set.

The computation of the WCRT in EDF (function Compute_WCRT(T) in Figure 4) is explained in Section 6.

5. Simulation results

To evaluate the DAI and CAI under static and dynamic scheduling policies, several task sets of N tasks have been randomly generated. Task sets are generated as follows:

- The utilization factor is $U_i = \frac{U_i}{U} \cdot \sum_{i=1}^{N} U_i$ with $U_i$ randomly generated between 0 to 1.
- The period $P_i$ is obtained as $P_i = C_i / U_i$ where $C_i$ is randomly generated between 1 to $C_{max}$.
- The deadline $D_i$ is obtained as $D_i = D_i \cdot P_i$, where $D_i$ is randomly generated between 0.5 to 1.

If the task set is unfeasible or the computation time of any task is greater than its deadline, the whole task set is generated again. The experiments consist of simulations with $N=5$ and total utilization factor $U$ of 0.65, ..., 0.99. 500 sets of tasks have been generated for each utilization factor, resulting in 7500 total simulations.

Figure 5 shows a comparison between CAI and DAI with decomposed tasks and non-split tasks.

![Figure 5. CAI and DAI comparison between split and non-split tasks set.](image)

Figure 6. CAI comparison between static and dynamic scheduling policies

As shown in Figure 5, a substantial reduction of CAI and DAI is accomplished. Unlike the static case, the task with the shortest deadline does not increases its CAI and DAI, since it is not always the highest priority task. Figure 6 shows a comparison between CAI and DAI minimization in static and dynamic priority algorithms.
6. Worst case response time in EDF

There are not too much work in the calculation of the WCRT of a task under EDF scheduling. In [14] a possible scenario to find this time is identified:

The WCRT of a task $T_i$ is found in a busy period in which all other tasks are released synchronously at the beginning of the period and then at their maximum rate.

The higher priority workload arrived to time $t$ is: (1)

$$ W_{i}(a,t) = \sum_{s_{j} \in s_{i}, t \neq s_{j}} \left[ \min \left\{ \frac{t}{P_j}, a \right\} + \left\lfloor \frac{a + D_j - D_i}{P_j} \right\rfloor C_j + \delta_j(a) C_j \right] . $$

where

$$ \delta_j(a,t) = \begin{cases} \min \left\{ \frac{t-a \left\lfloor \frac{a}{P_j} \right\rfloor}{P_j}, 1 + \frac{a}{P_j} \right\} & \text{if } t > a + \left\lfloor \frac{a}{P_j} \right\rfloor P_j \\ 0 & \text{otherwise} \end{cases} $$

The WCRT relative to $a$ of task $i$ is $W_{i}(a) = W_{i} - a - D_i$

In [17] a set of significant values for the parameter $a$ is given. Namely:

$$ A = \left\{ \bigcup_{j \in J} \left\{ k P_j + D_j - D_i : k \geq 0 \right\} \right\} \cap (0, L_i) $$

This is the set of instances for which $a + D_i$ coincides with at least the absolute deadline of another task’s instance. In these instances a change in the priority assignment occurs, so $T_i$ has a lower priority than before. In this situation, Spuri considers that $T_i$ will always have the lowest priority. To find the WCRT of all tasks in the set $T$ this method must be applied for every task $T_i$. This is not realistic, since in the run-time only one task will have the lowest priority and not all at the same time. When several absolute deadlines of tasks coincides, a rule of priorities must be established. A criterion could be to execute according to relative deadlines. In this case, the expression (1) must be rewritten:

$$ W_{i}(a,t) = \sum_{s_{j} \in s_{i}, t \neq s_{j}} \left[ \min \left\{ \frac{t}{P_j}, a \right\} \xi(i, j, D_i) + \left\lfloor \frac{a + D_j - D_i}{P_j} \right\rfloor C_j + \delta_j(a) C_j \right] . $$

And the maximum (1) is found in the coincidence of absolute deadlines only for tasks $D_j = D_i$. If $D_j > D_i$, the maximum values are found in the next instant. The new set of $a$ values proposed is:

$$ A = \left\{ \bigcup_{j \in J} \left\{ k P_j + D_j - D_i + \varepsilon_j : k \geq 0 \right\} \right\} \cap (0, L_i) $$

7. Conclusions and future work

In digital control systems, the control signal is always delivered with some delay. In a multitasking environment, the delay is variable, depending on the task priorities. If these delays are known in advance, they can be taken into account by the control algorithm. But the effect of the delay is not the same in all the control loops.

The priority assigned to the tasks determines the respective delay each task will suffer. In this paper, a comparison of these delays in static and dynamic scheduling policies has been presented. In order to reduce the delays, a task decomposition is proposed and analysed. The reduced delays improve significantly the control performance. The paper also presents a method to determine the WCRT under EDF scheduling.

8. References


