A Real-time Task Model Based on Ideal Instant

Fábio Rodrigues de la Rocha Rômulo Silva de Oliveira
frr@das.ufsc.br romulo@das.ufsc.br
LCMI - DAS – Federal University of Santa Catarina
P.O. Box 476, Zip Code 88040-900, Florianópolis – SC, Brazil

Abstract

In many digital control applications, data acquisition and process control are time-critical actions, assumed to be instantaneous and strictly periodic. However, aspects related to real-time scheduling algorithms may postpone the task execution and, consequently, delay the data acquisition and process control. We consider the acquired data from sensors as valid if acquired inside a time-interval that surrounds an ideal instant which is the point of maximum system's benefit. The data acquired before or after this time-interval are considered useless to the controller. We propose a new task model to deal with the variation in the execution time of a specific point inside tasks. In this model, tasks have an ideal instant to present their results or acquire data from sensors (QoS metric).

1. Introduction

In many digital control applications, data acquisition and process control are time-critical actions considered as both instantaneous and strictly periodic. In these systems, the controller is usually implemented as a software task. The scheduler, generally designed as a priority-based preemptive scheduling algorithm, decides which task runs according to a scheduling policy. Many controller tasks can simultaneously try to access I/O devices and processors.

Some aspects related to real-time scheduling algorithms may postpone task execution. As a consequence, the data acquisition and control actuation occur in a variable time. This variability is called jitter. Basically, we can identify three types of jitter: input, output and release. The input-jitter is caused by the variability between the start times in successive activations of the same task. The output-jitter is caused by the variability between the completion time in successive activations of the same task and the release-jitter is caused by the variability between the arrival time and the release time in successive activations of the same task. All types of jitter can degrade the controlled-system performance and even lead to instability.

In order to satisfy control application requirements we assume an ideal instant to acquire data from sensors, which gives the highest contribution to the system. Task value decreases before and after the ideal instant. Data gathered before or after the specific time-interval are considered useless for control purposes. Well-known utility functions based on deadline are not appropriate to describe this situation.

This paper describes utility functions suitable to represent a class of control system applications. Instead of using a deadline as the limit for finishing some actions, we use the concept of an ideal instant d_i (the i index represents the task, so d_i represents the ideal instant d related to the task i) and a time-interval within the data are considered useful. The ideal instant concept does not have to be associated with the end of a task. It can represent the best instant to acquire data or to send computed values to an output device (such as digital-analog converters). The location of the specific point inside a task, which must be executed on the ideal instant, is called action point g_i. Due to inherent scheduling problems, the action point could run at times different from the ideal instant. The time at which the action point runs is called action time (A_i). Our task model can improve the performance and quality of an application class by better representing its requirements. This scheduling problem is similar, but not the same, to the output-jitter problem. So, we evaluate the application of output-jitter reducing techniques to our task model by simulation. In section 2 we present our task model in detail. Many papers use different techniques to reduce the jitter in control systems and will be shown in section 3. Section 4 shows some simulations considering the techniques found in literature focusing the ideal instant problem. Section 5 concludes the paper.

2. The ideal instant model

In the ideal instant model we assume a periodic task set composed by n tasks executing on uniprocessor system. Tasks are executed following a priority policy. Each task can be preempted without penalty, it means a task executing on a shared processor may be interrupted at any instant of time, and its execution resumed later. Also, tasks can be blocked due to mutual-exclusion relationships. Each task T_i, where $1 \leq i \leq n$, has a maximum release jitter J_i; worst-case execution time WCET c_i and a period P_i. Each task T_i is assigned to a relative ideal instant d_i, which represents the highest contribution instant for a task.

According to this model a task is feasible whether its action time A_i occurs inside the time-interval $[s_i, l_i]$. An application is feasible if all their tasks are feasible. The time-interval $[s_i,\ l_i]$ represents the valid interval x_i such as $s_i = d_i - x_i$ and $l_i = d_i + x_i$. The application semantic is considered to define d_i and x_i. i.e. x_i could represent 10% of d_i, so $x_i = (0.1)d_i$.

2.1 Soft real-time system

The utility function given in figure 1 is used to describe soft real-time systems. Suppose the action time of k^{th} activation of task T_i is A_i^k. The contribution of that activation is denoted by V_i^k and so : $V_i^k = e^{-\left|A_i^k - d_i\right|}$. This function has its maximum at the ideal instant (the maximum contribution to the application). The function is asymptotic in the x-axis. In soft real-time systems tasks which miss their deadlines still have a value.

2.2 Firm real-time system

The utility function given in figure 2 is used to represent real-time tasks with firm real-time constraints. Suppose the action time of k^{th} activation of T_i is A_i^k. The contribution of this activation is V_i^k and so: $V_i^k = 0$ when $A_i^k < s_i$ or $A_i^k > l_i$, $V_i^k = 1 - \left|\frac{A_i^k - d_i}{d_i - s_i}\right|$ when $s_i \le A_i^k \le l_i$. The function has its maximum value at the ideal instant and its value linearly decreases until zero. After and before the time-interval this function has zero as a value (a task missed its deadline and there is no benefit in completing the task).

2.3 Hard real-time system

The model used to represent real-time tasks with hard constraints is similar to the firm real-time model. Suppose the action time of k^{th} activation of T_i is A_i^k, the contribution of this activation is V_i^k and so:

$V_i^k = -\infty$ when $A_i^k < s_i$ or $A_i^k > l_i$, $V_i^k = 1 - \left|\frac{A_i^k - d_i}{d_i - s_i}\right|$ when $s_i \le A_i^k \le l_i$

This function has its maximum value at the ideal instant (the maximum contribution to the system) and linearly decreases until zero. After and before the ideal instant the value is $-\infty$, case in which a hard real-time task missed its deadline.

3. Possible approaches from the literature

The approaches from literature can be grouped in classes according to the technique used to solve the problem of response-time variability in control systems:

1) time-driven schedulers; 2) compensation control algorithms 3) output-jitter minimizations thought changes in tasks properties; 4) new task models which consider the jitter and can help to reduce it.

Time-driven schedulers easily solve the problem of the ideal instant. However, they have limitations due to the inflexibility of this technique. The second approach is to include the jitter in the project of a control algorithm. This solution uses control theory to explore ways to deal with jitter instead of eliminating it [5]. In our work the main interest is related to approaches 3 and 4 because they deal with jitter as a real-time problem and consider it together with priority-driven schedulers.

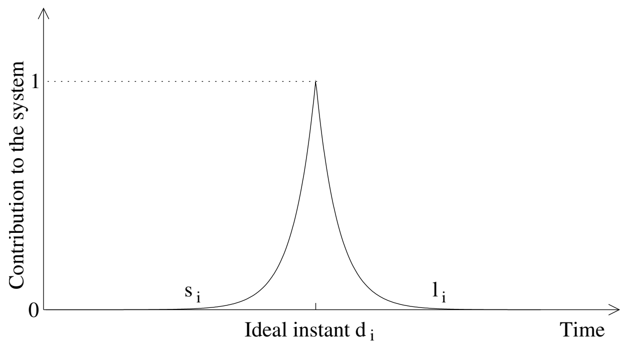

Figure 1: Utility function for soft real-time.

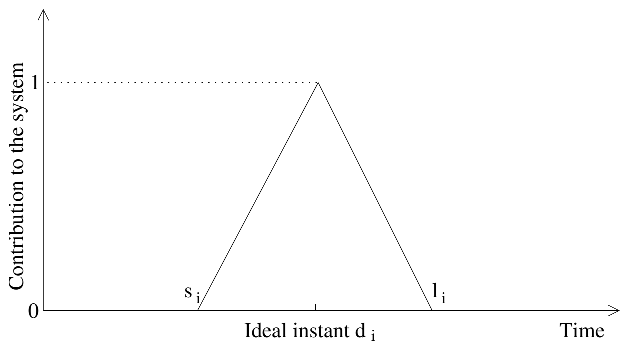

Figure 2: Utility function for firm real-time.

The third approach deals with the jitter problem using minimization algorithms to reduce the output-jitter. The minimization algorithm transforms the original feasible system into another feasible system with a lesser output-jitter. Usually, the algorithm adjusts some task properties such as the relative deadline according to an optimization technique. This class of papers is represented by [3] and [4]. The latter approach (4) is represented by papers such as [2] and [7]. Due to the importance to our research the papers [3] and [7] will be described.

3.1. Solution using output-jitter

The article [3] presents a polynomial algorithm to minimize the output-jitter. The algorithm is applied to a feasible task system under EDF. As a result, it produces a new task system also feasible, although with a lesser output-jitter. Output-jitter refers to a variation between the inter-completion time of successive task activations. A task is characterized by a minimum and a maximum separation P_i^{min} e P_i^{max} between successive completions. In a task system without jitter $P_i^{min} = P_i^{max} = P_i$.

In [3] the author considers periodic tasks denoted by T_i. Let c_i denote a computation time, P_i a period and

ϕ is a jitter tolerance factor of task i. The parameter ϕ has a following meaning: tasks with a large ϕ value have a large tolerance of jitter variation. At first, the deadline is considered equal to the period. The absolute jitter is defined to be $AbsJitter(T_i) \overset{\text{def}}{=} max(P_i^{max} - P_i, P_i - P_i^{min})$ and the absolute jitter of a scale $AbsJitter(\Gamma) \overset{\text{def}}{=} \underset{T_i \in \Gamma}{max}\{AbsJitter(T_i)\}$. In addiction, the weighted jitter is defined to be the jitter of task T_i divided by the tolerance factor ϕ. As a result, $WtdJitter(T_i) \overset{\text{def}}{=} \frac{AbsJitter(T_i)}{\phi}$ and $WtdJitter(\Gamma) \overset{\text{def}}{=} \underset{T_i \in \Gamma}{max}\frac{AbsJitter(T_i)}{\phi}$. The basic idea is to minimize the equation $WtdJitter(\Gamma)$. Before that, we compute the jitter of T_i as $Jitter(T_i) \leq (P_i - c_i)$. Which results in $WdtJitter(\Gamma) \leq \underset{T_i \in \Gamma}{max}\{\frac{P_i - c_i}{\phi_i}\} = \underset{T_i \in \Gamma}{max}\{\frac{c_i}{\phi_i}(\frac{1}{\rho_i} - 1)\}$.

A tighter limit can be obtained on those cases in which the utilization ρ é less than one, thus: $WdtJitter(\Gamma) \leq \underset{T_i \in \Gamma}{max}\{\frac{c_i}{\phi_i}(\frac{\rho}{\rho_i} - 1)\}$. To minimize the absolute jitter $AbsJitter(\Gamma)$, one changes the processor-share reserved for the use of each task ρ_i for θ_i, such that $\theta_i \geq \rho_i$ for all tasks. The goal is to increase the processor-share, such that $\sum_{i=1}^{n} \theta_i \leq 1$. The minimization algorithm chooses new processor-shares for each task. Each new processor-share must be greater than or equal the original ρ. The tightest jitter bound can be obtained using a binary search. Finally, new tasks's deadlines can be obtained from the new processor-shares.

3.2. Solution using task with offsets

The article [7] describes a task set which uses offsets, that is, a constant interval between tasks arrival. Tasks with offsets are mainly used for three reasons. Firstly, to model applications in which the task arrival time must obey a precedence relationship with other tasks. Also, tasks with offsets are suitable to improve the schedulability analysis. Usually, feasibility tests consider the worst-case scheduling scenario. The critical-time instant occurs when all tasks are ready to execute, as a result, it is the point of maximum load. In this scenario, if a task is accepted in the worst-case scenario it will be always feasible. Using offsets the number of ready tasks decreases, so a lower load is achieved. A scheduling test intended to analyze task systems with offsets would give better results.

Finally, offsets can be used to reduce the output-jitter. A possible solution to reduce the output-jitter of a task A would add a new task B with a higher priority. Task A performs the computation, and writes its results in a buffer shared with task A. Task B is released an offset later. Task B has a higher priority and the response time in the worst-case scenario will be lower. Thus, by using task splitting, a lower variation in the completion time of task B is achieved. The approach of replacing a single task for two tasks is feasible if two constraints are met. Task B must start after task A finishes $O_B > r_A$. The original deadline must be kept $O_B + r_B \leq D_A$ where D_A is the original deadline of task A.

4. Evaluation of the two main approaches

In section 2 some metrics were created to express the contribution of a task activation to the system based on the time in which a task presents its results or acquires data from sensors. In this section we present some empirical studies to evaluate the two main approaches from section 3 according to the ideal instant model. Some task sets were created and executed using the scheduling algorithms and the features of those approaches. The quality of task scheduling was evaluated using the metrics from section 2 on soft, firm and hard real-time systems.

4.1. Simulation using output-jitter minimization

In the simulations the metrics of quality from section 2 are collected for each task and the results are presented in table 1. The action point is set to 50% of the effective execution time (the middle of the task) and the ideal instant is set to the middle of the worst-case execution time. Activations in which the execution time is lower than the WCET result in an action time A_i before the ideal instant. Delays in the task execution caused by interference from high-priority tasks result in the shifting of the action time to after the ideal instant.

Tables 1 and 2 show the simulation results. In these simulations tasks were randomly-generated with an integer execution time between 1 and 10. Also the periods were randomly chosen to give a total system load between 0.2 e 0.9. For each load, 300 task sets composed by 5 tasks were generated and executed for 50000 t.u. For each task activation, values are gathered corresponding to the system contribution, considering $x_i = (0.1)d_i$ (3 values: soft, firm and hard). Finally, a mean value is calculated to obtain the benefit (for soft, firm and hard constraints) for each task. For a simulation of 5 tasks we obtain the mean value which gives the system contribution. The results in table are mean values calculated as described above and ranging from 0 to 1 or 100%.

Although the minimization approach can reduce the output-jitter, it does not guarantee good results for the ideal instant problem. The simulations were incapable of showing a visible improvement over the ideal instant when it is not located at the end of task. Thus, the use of minimization can not solve the ideal instant problem.

load	soft	firm	hard	soft	firm	hard
0.2	0.629	0.357	0.357	0.642	0.358	0.358
0.3	0.580	0.309	0.309	0.595	0.316	0.316
0.4	0.534	0.283	0.283	0.538	0.275	0.275
0.5	0.487	0.236	0.236	0.484	0.229	0.229
0.6	0.434	0.205	0.205	0.437	0.196	0.196
0.7	0.393	0.179	0.179	0.387	0.179	0.179
0.8	0.355	0.159	0.159	0.341	0.139	0.139
0.9	0.315	0.127	0.127	0.306	0.123	0.123

Table 1: Before minimize. Table 2: After minimize.

4.2. Tasks with offsets

This section shows how tasks with offsets can be used to reduce the output-jitter of a task system. Using this approach we can model the ideal instant problem though task splitting and offsets. The time instant in which the ideal instant is located is assumed as a single task with a higher priority. This is possible to reduce the variation in the action time. Although, the action point can be located on any task position (not only at the end), it would be more interesting that transactions were composed by three tasks. Because of its small size and also its almost constant instruction set, the task responsible for the ideal instant can be considered with a constant and small execution time C_B.

The division in subtasks presents some problems because the subtask responsible for the ideal instant must have a higher priority. Also, it must be released after an offset time from the transaction release. The offset is computed considering the previous task has finished $O_B > r_A$. Thus, it is necessary to compute the worst-case response time for the initial subtask $r_A = C_A + \sum_{i \epsilon hp(T_A)} C_i \lceil \frac{r_A}{P_i} \rceil$. The offset value means the minimum time to start the subtask B and in case the offset (response time of A) is lower then the ideal instant we consider the ideal instant as a new offset. $O_B = max(r_A, \text{ideal instant})$. Unfortunately, in case task A response time is greater than the ideal instant it is not possible to reach the scheduling goal.

We have developed a set of rules to assign subtasks priorities according to task groups. Subtasks which represent the subtasks B belong to the highest priority group. The subtasks A belong to the medium priority, and the lower priority group covers subtasks C. Inside each group it is used the Deadline Monotonic algorithm to assign priorities. To avoid that task B starts before task A, there is the offset which ensures task A has finished. To avoid that task C starts before task B we choose to use the same offset of task B. Task B has a higher priority than task C, so, it will start before task C.

In the first simulation (results presented by table 3), the action point was positioned at 50% of the effective computation time and the ideal instant positioned at the middle of worst-case execution time, other parameters are consistent with previous simulations.

So far the task division approach and offsets appear to give a high contribution value to the system. Unfortunately, a subtask in which the ideal instant is located is only released after a time measured as $max(r_A, \text{ideal instant})$. The best scenario occurs when the ideal instant is greater than the response time of subtask A. Also, better results can be reached if a less pessimistic algorithm to compute the worst-case response time is applied. The real-time literature has research papers which improve this analysis such as [1] and [6].

In a second simulation, the action point is positioned at 50% of the effective computation time and the ideal instant positioned at $3.WCET$ of the task. As the ideal instant has been moved to a future time, the response time of subtask A is no longer the main factor to compute task B offset. As a result, the action time from task B is near the ideal instant giving a high contribution to the system as shown by table 4. The maximum contribution is never achieved due to interference by higher priority tasks.

load	soft	firm	hard
0.2	0.032	0.000	$-\infty$
0.3	0.033	0.000	$-\infty$
0.4	0.032	0.000	$-\infty$
0.5	0.032	0.000	$-\infty$
0.6	0.033	0.000	$-\infty$
0.7	0.032	0.000	$-\infty$
0.8	0.032	0.000	$-\infty$
0.9	0.032	0.000	$-\infty$

Table 3: At middle.

soft	firm	hard
0.898	0.876	0.876
0.897	0.877	0.877
0.897	0.876	0.876
0.911	0.888	0.888
0.906	0.885	0.885
0.912	0.892	0.892
0.902	0.881	0.881
0.900	0.875	0.875

Table 4: $3.WCET$.

5. Conclusion

In this paper, we proposed a new task model based on ideal instant. The proposed model is typically suitable for control applications improving both the quality and performance of controllers. Evaluations were executed using well known approaches from the real-time literature to deal with the ideal instant problem. They showed that the well known approaches were unable to solve satisfactorily the ideal instant problem.

A scheduling algorithm to address the ideal instant model is necessary. Therefore as a future direction the authors intend to create a new scheduling algorithm to optimize the quality metric of the ideal instant.

References

[1] J. P. Gutierrez and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proc. 19th IEEE Real-Time Systems Symposium (RTSS)*, 1998.

[2] C. Han and K. Lin. Scheduling distance-constrained real-time tasks. *Proceedings of the Real-Time Systems Symposium*, pages 300–308, 1992.

[3] S. k. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari. Scheduling periodic task systems to minimize output jitter. In *Sixth International Conference on Real-Time Computing Systems and Applications*, pages 62–66, 1999.

[4] T. kim, heonshik Shin, and N. Chang. Deadline assignment to reduce output jitter of real-time tasks. *Proc. of The 16th IFAC Workshop on Distributed Computer Control Systems*, pages 67–72, 2000.

[5] B. Lincoln. Jitter compensation in digital control systems. In *Proc. of the 2002 American Control Conf.*, May 2002.

[6] J. Mäki-Turja and M. Nolin. Tighter response-times for tasks with offsets. *Real-time and Embedded Computing Systems and Applications Conference*, 2004.

[7] K. Tindell. Adding time-offsets to schedulability analysis. Technical report, University of York, Computer Science Dept, YCS-94-221., 1994.