Controlling Real-Time Tasks Schedule Using the Value Parameter

Crineu Tres\textsuperscript{1}, Edgar Nett\textsuperscript{2}, Leandro Buss Becker\textsuperscript{1}

\textsuperscript{1}Automation and Systems Department - Federal University of Santa Catarina
Florianopolis, Santa Catarina, Brazil

\textsuperscript{2}Institute for Distributed Systems (IVS) - Otto-von-Guericke Universität Magdeburg
Magdeburg, Germany

\{crineu,lbecker\}@das.ufsc.br, nett@ivs.cs.uni-magdeburg.de

**Abstract.** Modern real-time applications are very dynamic and cannot cope with using worst-case workload. Therefore scheduling algorithms able to deal with overload situations are required. In this context the Value-Based Scheduling theory has become useful to add flexibility to such systems. This paper presents a comparative study among different real-time schedulers during overload, including traditional algorithms and the proposed DMB (Dynamic Misses Based). The main goal is to define the most suitable algorithm to be used with TAFT (Time-Aware Fault-Tolerant) scheduler in order to achieve a predictable system behavior. Obtained results show that DMB reached the most promising results because of its ability to control tasks degradation in a determined way.

1. Introduction

Modern real-time systems, such as applications where robot teams play soccer against each other, are very dynamic, specially due to the unpredictable environment that they interact with. Therefore it is difficult to predict their exact workload, as it can be seen in tasks like sensors reading. Moreover, working with worst case scenarios in these kind of applications would be too pessimistic and implies overestimating the required computational resources.

Such scenario motivates the use of scheduling algorithms that give some flexibility in terms of workload characterization. One way to achieve such flexibility is the capacity to avoid working with worst case execution times (WCET), but this approach implies the need for the ability to cope with overload situations. One algorithm that follows this trend is the Time-Aware Fault-Tolerant (TAFT) scheduler.

Although TAFT behaves properly during overload situations (providing a high overall utilization factor), it has the same problem from most scheduling policies: the incapacity to evaluate which tasks are preferable to be discarded during overloads. In other words, it reflects the incapacity to judge between task’s importance to the application.

The meaning of importance has been widely discussed by the research community over the last two decades. The value parameter was introduced by [Jensen et al. 1985] in order to reflect the benefit obtained from finishing a collection of services. Afterwards [McElhone 1996] argued that its use is a popular way to increase application flexibility.
The main goal of this work is to evaluate if the value parameter represents a useful metric for defining tasks importance and, if so, which is the best scheduler to use in conjunction with it. Moreover, it also focuses on preserving the overall tasks behavior in the long term, i.e. trying to balance misses ratio among all tasks, resulting in a predictable execution. Obtained results suggested the creation of a new scheduling policy that aims to control tasks misses in overload conditions. This new policy is detailed in the paper, and has shown to be very suitable to be applied in conjunction with TAFT scheduler.

The rest of the paper is divided as follows: section 2. presents the related works on value-based scheduling and details the TAFT scheduler, considering its drawbacks that motivated this work; section 3. describes the evaluated algorithms and the new scheduling algorithm defined, together with a description of the simulation conditions in the performed analysis; section 4. presents the obtained results; finally section 5. draws the conclusions from this work and depicts the future directions of this research.

2. Related Works

2.1. Value-Based Scheduling

There are some related works in the literature that use the value parameter to compute the schedule of the application tasks. The introduction of the concept of Time/Utility Functions (TUFs) was made by [Jensen et al. 1985], as an elaborated way of defining the value. TUFs work as generalization of the deadline constraint, specifying the utility to the system resulting from the completion of a task according to the time it occurred.

[Buttazzo et al. 1995] presented a study aiming to find the best scheduling algorithm in overload conditions. The value parameter was used to evaluate results, which shows that the best algorithm changes depending on the workload, as suggested for further improvements. More recently, [Ravindran et al. 2005] ratify the use of TUFs and the utility accrual scheduling paradigm as a more generalized, adaptive, and flexible approach to dynamic real-time systems.

2.2. TAFT Scheduler

The TAFT (Time-Aware Fault-Tolerant) scheduler [Gergeleit and Streich 1996, Nett and Streich 1997, Nett et al. 2001, Becker et al. 2005] was defined to handle timing faults in uncertain environments such that deadlines are still met. In TAFT each task is designed as a TaskPair (TP) with a common deadline (D_{TP}). A TP constitutes a MainPart (MP) and an ExceptPart (EP), thus reflecting the fault-tolerant aspect. From the scheduler’s point of view, both parts are treated as separate scheduling entities having their individual timing parameters. The TAFT scheduler is a hard real-time scheduler in the sense that it always guarantees the completion of either the MP or the EP before D_{TP}. However, an EP has to be executed if and only if its MP cannot be successfully completed before D_{TP}. This means that the real functionality is implemented in the MP. The minimal functionality of the EP is to ensure that the respective TP leaves the controlled application in a fail-safe state and the controlling system in a consistent state.

TAFT adopts a two-level dynamic scheduling strategy to schedule the tasks. The current implementation uses the Latest Release Time (LRT) algorithm for the first level, that provides the highest priority and is in charge of scheduling the EPs, and the Earliest
Deadline First (EDF) algorithm for the second level, that has the lowest priority and is responsible for the MPs.

Another important aspect from TAFT regards to the timing parameter “execution time”, which for the EPs is considered as Worst-Case Execution Time (WCET). On the other hand, for the MPs it can be interpreted as an Expected-Case Execution Time (ECET), which is a measure for the time that a certain percentage of instances of a task needs for a successful completion [Nett et al. 2001].

Nevertheless, experiments conducted in [Becker et al. 2005] show that choosing different ECETs for controlling the completion rate of a task proved not to be a suitable solution. Therefore, in this work we replace the second-level scheduler (formerly EDF) to evaluate other approaches that use the value parameter to compute the schedule.

3. Algorithms Evaluation

3.1. Adopted Performance Criteria

When tasks timing constraints are expressed using a value (or TUF), the scheduling optimality criteria is based on the accrued activity utility, that works as follows: every time a task completes, the total utility of the system is increased by the value of the task (or value of the TUF with completion time as initial parameter). This is called the utility accrual (or UA) criteria. Each analyzed algorithm is simply evaluated under its utility accrued ratio ($UA_R$), calculated as shown in equation 1.

$$UA_R = \frac{\text{utility accrued}}{\text{task set total utility}}$$

Another important definition that is used to characterize the different runs from the simulations is the system utilization. A system has as nominal utilization the sum of all tasks’ nominal load $NL_i$, that can be calculated using equation 2:

$$NL_i = \frac{WCET_i}{\text{period}_i}$$

The nominal utilization can be determined even before the simulation is started because it is based in the WCET, which has a fixed value. On the other hand, the effective utilization can only be calculated at the end of the simulation. This is because it uses as parameter the tasks real execution times, which is the time that the task used in fact to complete. The effective utilization is calculated using the same equation 2, just replacing the WCET in equation for the average real execution time.

3.2. Evaluated Scheduling Algorithms

The schedulers used along this study are mostly the same ones used in the previous works that form the basis of this paper [Buttazzo et al. 1995, Nett et al. 2001]: EDF, HVF and HDF. The classical EDF (Earliest Deadline First) scheduler is used here for performance comparisons, both in cases of underload, where it is the optimum solution, and overload, for evaluation of the domino effect impact. For the HVF (High Value First) scheduler only tasks value matters. HDF (High Density First) is a mix of the other two algorithms,
assigning priorities as $V_i/c_i$, where $V_i$ means the value and $c_i$ the remaining WCET needed to complete the execution of a task $T_i$.

Besides these classical scheduling algorithms a new specific algorithm is proposed in this work. It is created given that the previously ones do not take deadline misses into consideration and thereby they look like less suitable to reach our goals.

### 3.2.1. Dynamic Misses Based Algorithm

The Dynamic Misses Based (DMB) algorithm is defined with the intention of controlling general tasks behavior in overload situations. DMB takes into account the task value and the missed deadlines to calculate the schedule. Priorities are assigned as $V_i \times (1+MD_i)$, where $V_i$ is the value and $MD_i$ the percentage of missed deadlines of task $T_i$.

In a situation with constant values (or step functions as TUF) and without missed deadlines, DMB behaves like HVF scheduler. The difference appears when tasks start missing deadlines and so their priorities raises. Tasks that are in the top of the priority queue at any given moment can become no more the highest priority task and start missing deadlines. This priority inversion with tasks switching places at the priority queue happens indefinitely while tasks have less than 100% misses. At this point tasks reach their maximum possible priority.

It’s important to notice that this behavior is presented only if proper values are chosen among the tasks, since unsuitable values results in the before mentioned HVF behavior. Other important detail relates to deadline misses, which cannot be determined individually, but only in relation with other tasks.

### 3.3. Simulated Task Set

The performed simulation is based in the set of tasks and loads proposed in the Hartstone Benchmark [Weiderman 1989]. More precisely, it uses the PN series, composed of periodic non harmonic tasks. The initial simulation scenario present a nominal utilization of 80%, and the established simulation procedure is to successively increase the workload while monitoring the deadline misses. The WCET of all the tasks are increased by the same percentage amount in order to achieve a higher utilization.

The MPs execution times are based in real observations where it is attested that they can be represented by the beta probability distribution function (PDF), with parameters $a = 2$ and $b = 3$ [Schemmer 2004]. These times oscillate during simulation between 50% and 100% of the WCET. Execution times of EPs are calculated as 5% of the MP WCET. The initial load is increased and the effective utilization variates in average from 85% to 130%. Moreover, values are assigned to tasks at design time and are not further modified, tasks are preemptable and do not compete for resources and all simulations had lasted for something equivalent to a 30 seconds period.

### 4. Obtained Results

#### 4.1. Simulations Using Plain Schedulers

Although a similar analysis was performed in [Buttazzo et al. 1995], except for the DMB algorithm, we decided repeating these experiments for the following reasons: to use a
task set that is closer to a real application, to focus on the effective rather than in the nominal utilization levels and to analyze the performance of the individual tasks along the simulation. The graph from figure 1 shows the utility accrued ratio from all four schedulers during the experiments.

As expected, EDF is the most efficient solution before the 100% utilization barrier but has an outrageous performance loss after it. The well-known domino effect makes EDF an unacceptable solution in overload conditions. An individual analysis of EDF concludes that it performs in an “all or nothing” basis, with tasks presenting almost no misses before 100% utilization and almost no correct completions after that.

HVF and HDF schedulers seems to be immune to overload, and do not present any considerable performance decrease along execution, leading to the conclusion that they present better performance results. The explanation is simple: while EDF and DMB try to execute all tasks in the long run, HVF and HDF begin to discard the lower priority tasks as utilization increases (under overload). Observing figure 2(a), which states the tasks behavior under HDF scheduling, it is clear that task Task_0 is the only one to miss deadlines until it is almost completely discarded. Afterwards Task_1 starts having significant losses, and became the only task with rapid increasing losses until it its also discarded. The HVF graph exhibits the same trend from HDF.

![Figure 1. Plain schedulers performance: UA_R vs. effective utilization](image1.png)

![Figure 2. Tasks misses % vs. effective utilization](image2.png)
As stated before, DMB is another algorithm that suffers the impact from overload, though slightly better than EDF. Its performance after 100% utilization is better because the utility accrued is being evaluated, and DMB takes value into account to compute the schedule, while EDF ignores it. Figure 2(b) shows the individual tasks behavior using DMB. The positive fact is the acknowledgment of the uniform degradation from all tasks, as proposed in the algorithm specification. Another eminent feature is the inversely proportional relation between tasks degradation and value.

Through these individual and overall analysis, it is noticed that in overload conditions stand-alone scheduling algorithms behave as follows:

a) choose one task to discard at a gradual rate, so the other tasks do not suffer with the overload - keeping the system responsive, but without some functionalities;
b) try to “ignore” the overload and continue to execute all tasks - probably losing several deadlines and producing a less responsive or even a non-responsive system.

The addition of the value parameter plays an important role because it is useful in both situations described. In a), as seen in HVF and HDF results, the order in which tasks are discarded (or stop responding) can be determined by the right choose of value. In b), as depicted in the DMB results, values can determine the relation between task losses.

4.2. Simulations Using TAFT Scheduler

In this section the selected algorithms are used in conjunction with TAFT scheduler. Figure 3 shows the performance of all four algorithms. For a closer look in the obtained results graph, the y-axis started at 50% instead of previous value of 0%.

Figure 3. Schedulers performance using TAFT: \( U_{AR} \) vs. effective utilization

Using TAFT a very different scenario is obtained, with higher overall ratios and without any abrupt performance decrease by any algorithm. The main reason for this is found in TAFT definition (section 2.2.), where there are no “missed” deadlines, because before the TP deadline one of its parts (MP or EP) has to be executed. Instead, there are EP executions, which is something totally different and cannot be compared directly. Without missed deadlines transient overload are not carried on.

It is possible to observe again the outstanding results from EDF early on, which is surpassed only after the mark of 110% effective utilization, but without an abrupt performance fall. Moreover, HDF presents the best results in overload conditions, while DMB
and HVF have similar gradual degradation along time. The tasks individual results while scheduled with EDF using TAFT exhibits a behavior that reminds HDF and HVF in the previous analysis, where they started to discard tasks as soon as overloads starts. However this time it is possible to see intervals where multiple tasks are executing both MPs and EPs simultaneously (remember that to execute a MP the EP rate must be bellow 100%).

![Graph](image)

**Figure 4. EP executions % vs. effective utilization**

HDF individual results presented in figure 4(a) exhibits the overall behavior seen in the graphs of the other traditional algorithms, EDF and HVF, and its performance remain similar to the previous analysis. Both tasks with the lower densities execute EPs along simulation, in increasing percentages, while the other three show almost no EP executions. The only observed modification is similar to what happened with EDF, presenting intervals where multiple tasks are executing both MPs and EPs simultaneously.

In fact, DMB in conjunction with TAFT reached the most promising results for the established goal of balancing the EPs occurrences among the TPs. This result is shown in figure 4(b), which presents tasks having linear increase of EPs execution percentage. For a closer look the $y$-axis was limited by the 50% mark. Analyzing two distinct tasks it is possible to establish that the relation between the number of EPs executions is inversely proportional to the relation between the tasks values. This characteristic is found within any two tasks, which ratifies that the absolute number of tasks EPs executions cannot be controlled (because it depends on the effective utilization), but the percentage of EPs executions from one task relatively to any another task can be controlled.

There is also the advantage related to the absence of transient overload propagation. Solutions like DMB can succeed on keeping the system in a less responsive state during some period of time and immediately recover to the normal state afterwards. This behavior meets the goal of having a system in constant high utilization and that does not present any serious damage after a temporary overload situation.

5. Conclusions

The main goal of the performed evaluation was to define the most suitable algorithm to be used in conjunction with the TAFT scheduler, taking into consideration their capacity to follow the value parameter. Therefore two different analyzes were conducted, one focusing on the individual behavior of the original schedulers and another one evaluating
the behavior of the schedulers in conjunction with TAFT. Obtained results show that the scheduling algorithms that use the value parameter present different behavior in respect to the order in which tasks are discarded (or stop responding to the system) or even to determine the relation between tasks losses.

In conjunction with TAFT, the DMB algorithm reached the most promising results for the established goals. The importance of the DMB is not directly related to its performance, since it definitely not showed the best overall results under the defined metric, but on its ability to control tasks degradation in a gracefully and determined way along execution. Nevertheless, solutions like HDF could also be combined with TAFT, with the drawback of losing the lowest valuable tasks, but with the benefit of having the best overall result.

As future work direction it should be considered the use of DMB with dynamic values along execution, looking for an on-line adjustment of tasks and services priorities as they are being executed. Preliminary results show that this is a prosperous research direction to be followed.

References


