Modelling and Analysis Suite for Real Time Applications
(MAST 1.5.1)

Analysis Techniques used in MAST

By: Michael González Harbour mgh@unican.es
    José Carlos Palencia Gutiérrez palencij@unican.es
    José Javier Gutiérrez García gutierjj@unican.es
    Juan María Rivas Concepción rivasjm@unican.es

Copyright © 2000-2015 Universidad de Cantabria, SPAIN

1. Introduction

The MAST toolset contains several schedulability analysis tools capable of analysing single processor and distributed systems scheduled with fixed priority, EDF, and EDF within priorities scheduling policies. The tools are based on different scheduling analysis techniques published in the literature:

- **Classic RM Analysis.** This analysis implements the classic exact response time analysis for single-processor fixed-priority systems first developed by Harter [5] and Joseph and Pandya [10], and later extended by Lehoczky to handle arbitrary deadlines [12] and by Tindell to handle jitter [31]. It corresponds to Technique 5, “Calculating response time with arbitrary deadlines and blocking”, in [11].

- **Varying Priorities Analysis.** This analysis implements the response time analysis for single processor fixed priority systems in which tasks may explicitly change their priorities, developed by González, Klein and Lehoczky [6]. It corresponds to Technique 6, “Calculating response time when priorities vary”, in [11]. In terms of the MAST model, transactions for this tool are built as linear transactions, with a sequence of activities, each representing the execution of an operation with a possibly different priority level specified through a `Permanent_Overriden_Priority` attribute. However, each transaction is limited to having a single segment. A segment is a continuous sequence of activities executed by the same server. Note that all the other fixed priority tools used in MAST require that the priority of an activity is the same at the start and at the end, and thus they do not allow using `Permanent_Overriden_Priority` attributes.

- **EDF Monoprocessor Analysis.** This analysis implements the exact response time analysis for single processor EDF systems first developed by Spuri [28]. In the MAST implementation we use the EDF Within Priorities (see below), because there may be interrupt service routines (modelled as fixed priority tasks) in addition to the EDF tasks.

- **EDF Within Priorities Analysis.** This analysis is a mixture of the response time analysis for fixed priority systems [11][12][31] and for EDF [28]. It is capable of analysing systems with hierarchical schedulers, in which the underlying primary scheduler is based on fixed priorities, and there may be other EDF (secondary) schedulers scheduling tasks at a given priority level. It was developed by González and Palencia [7].

- **Holistic Analysis.** This analysis extends the response time analysis to multiprocessor and distributed systems. It is not an exact analysis, because it makes the assumption that tasks of the same transaction are independent. It was first developed for fixed priority
systems by Tindell and Clark [31][32] and refined by Palencia et al [16]. For EDF systems, Spuri [29] and later Palencia [15] extended this technique for EDF systems scheduled with global-clock deadlines (referred to a global synchronized clock), and Rivas et al [34] for systems scheduled with local-clock deadlines (with local clocks in each processor).

- **Offset Based Approximate Analysis.** This is a response time analysis for multiprocessor and distributed systems that greatly improves the pessimism of the holistic analysis by taking into account that tasks of the same transaction are not independent, through the use of offsets. Offset based analysis for fixed priorities was first introduced by Tindell [33] and then extended to distributed systems by Palencia and González [17]. It was later extended to global-clock EDF systems by Palencia and González [19] and to local-clock EDF systems by Díaz de Cerio et al [3]. Although it provides much better results than the holistic analysis, it is not an exact method because the exact analysis is intractable. The method approximates the interference of a transaction with a maximum interference function that is independent on the phase of the transaction.

- **Offset Based Approximate with Precedence Relations Analysis.** This is an enhancement of the offset based approximate analysis for fixed priority systems in which the priorities of the tasks of a given transaction are used together with the precedence relations among those tasks to provide a tighter estimation of the response times. It was developed by Palencia and González [18], and later enhanced by Redell [22]. The extension to local-clock EDF was done by Díaz de Cerio et al [3].

- **Offset Based Slanted Analysis.** This is another enhancement of the offset based approximate analysis for fixed priority systems in which the maximum interference function is defined with a tighter approximation. This method provides better results than the Offset-Based Approximate Analysis, but it is uncertain if it gets better results than the method with precedence relations. In general both techniques should be applied, and the best results used as an upper bound on the response times, as both methods are pessimistic. It was developed by Mäki-Turja [15].

- **Offset Based Brute Force Analysis.** This is an exact analysis of offset-based transactions, initially developed by Tindell [33]. It analyses all possible combinations of tasks initiating the critical instant for each transaction, which leads to a combinatorial problem that only offers results for very small systems.

- **Analysis for heterogeneous distributed systems.** This technique [23] allows the integration of different response-time analysis techniques so that they can be applied to heterogeneous distributed systems with different scheduling policies in each resource. The Holistic and all the Offset-Based analysis techniques are now integrated in MAST into the distributed analysis for heterogeneous systems.

The MAST model is able to support multi-path transactions in which the event flow can be divided into several paths with fork (multicast) or branch (delivery or query server) elements, and paths may be joined with join (barrier) or merge (concentrator) elements. The analysis of these multipath transactions is described in [9] using the holistic analysis technique, and is implemented in MAST 1.5 with the following restrictions: the transaction has a single input event and only fork, join or merge elements are supported; branch is not supported, as the analysis in [9] only works for transactions with deadlines at or before the end of the period. The holistic analysis technique is the most pessimistic, so in the future we plan to extend offset-based analysis to handle these multipath transactions.
The MAST toolset is able to automatically calculate the blocking times caused by mutual exclusion synchronization. The model includes shared resources with the immediate ceiling [2], priority inheritance [26], and stack resource [2] synchronization protocols. It also allows mixtures of these protocols and their use in multiprocessor systems [20][21], with some restrictions that are described below. For the basic priority inheritance, Rodriguez and Garcia [24] showed that most implementations do not strictly follow the rules in [26], and that the amount of blocking is usually higher than that predicted by the theory. In MAST we take this into account when calculating the blocking times due to the use of the priority inheritance protocol.

The MAST toolset also contains tools to automatically assign priorities and other scheduling parameters. Priority assignment tools are provided for single-processor and distributed systems. In single-processor systems, if deadlines are within the periods the optimum deadline monotonic priority assignment developed by Leung and Layland is used [13]. The Liu and Leyland classic rate monotonic priority assignment for the case of deadlines equal to the periods [14] is known to be a special case of the deadline monotonic assignment. When deadlines are larger than the task periods, the optimum priority assignment developed by Audsley is used [1]. This technique is based on the iterative use of the schedulability analysis tools for different solutions, until a schedulable solution is obtained.

In multiprocessor and distributed systems the problem of assigning scheduling parameters is much harder, as there are strong interrelations between the response times in the different resources. For distributed systems scheduled with fixed priorities, we provide two heuristic solutions based on iteratively applying the schedulability analysis tools. The first one is based on the use of the simulated annealing optimization techniques, first used by Tindell, Burns, and Wellings for assigning priorities [30]. The second heuristic, which usually provides better and faster results is the HOPA algorithm developed by Gutiérrez and González [8]. For distributed EDF systems, the HOSDA [34] algorithm, which is an evolution of HOPA, is provided. This algorithm is capable of assigning and optimizing local and global scheduling deadlines. In addition to HOSDA, other simpler mechanisms algorithms are provided for comparison purposes: the Proportional Deadline assignment (PD) [36], and the Normalized Proportional Deadline assignment (NPD) [36], which takes into account the CPU utilization. These techniques can be applied for the assignment of both local [35] and global [34] scheduling deadlines. All these techniques are integrated into HOSPA [23], the scheduling parameters assignment tool provided for heterogeneous FP/EDF systems.

The MAST toolset is able to determine not only whether the system is schedulable or not, but also how far it is from being schedulable, or how much capacity is available until the system becomes unschedulable. It does so by calculating slacks, which are defined as the percentage by which we can increase the execution times of some operations while keeping the system schedulable (for positive slacks) or the percentage by which we have to decrease the execution times to make the system schedulable (for negative slacks). A slack of zero means that the system is just schedulable, and that even the smallest increment in the execution times would lead to non schedulability. There are different kinds of slacks provided:

- **System slack**: affects all the operations in the system
- **Processing resource slack**: affects only the operations executed in a given processing resource
- **Transaction slack**: affects only the operations used in a given transaction
- **Operation slack**: affects only one single operation
The slacks are calculated by modifying the worst-case execution times and repeating the analysis using a binary search to find the point in which the system becomes unschedulable (or schedulable if it wasn’t). The slack is calculated with a 1% precision to limit the amount of times the analysis is repeated to around 20 times.

References


