Journal of Computer and System Sciences 81 (2015) 1542–1555

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

# Task partitioning and priority assignment for distributed hard real-time systems [☆]

Ricardo Garibay-Martínez [*], Geoffrey Nelissen, Luis Lino Ferreira, Luís Miguel Pinho

*CISTER/INESC-TEC Research Centre, ISEP/IPP, Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto, Portugal*

A R T I C L E   I N F O

*Article history:*
Received 21 July 2014
Received in revised form 29 March 2015
Accepted 5 May 2015
Available online 17 June 2015

*Keywords:*
Real-time
Distributed systems
Task allocation
Priority assignment
Intermediate deadlines
Holistic analysis
Multi-threaded parallel tasks

A B S T R A C T

In this paper, we propose the Distributed using Optimal Priority Assignment (DOPA) heuristic that finds a feasible partitioning and priority assignment for distributed applications based on the linear transactional model. DOPA partitions the tasks and messages in the distributed system, and makes use of the Optimal Priority Assignment (OPA) algorithm known as Audsley's algorithm, to find the priorities for that partition. The experimental results show how the use of the OPA algorithm increases in average the number of schedulable tasks and messages in a distributed system when compared to the use of Deadline Monotonic (DM) usually favoured in other works. Afterwards, we extend these results to the assignment of Parallel/Distributed applications and present a second heuristic named Parallel-DOPA (P-DOPA). In that case, we show how the partitioning process can be simplified by using the Distributed Stretch Transformation (DST), a parallel transaction transformation algorithm introduced in [1].

## 1. Introduction

Modern distributed real-time systems range from safety critical to entertainment and domestic applications, presenting a very diverse set of requirements. Although diverse, in all these areas, distributed applications are becoming larger and more complex. Furthermore, such complex applications require the use of more powerful hardware and software architectures in order to comply with their stringent time constraints. The use of multi-threaded parallel processing has emerged as a promising solution for providing extra computing power to such demanding real-time applications. Therefore, the real-time community has been making efforts to extend traditional real-time tools and scheduling algorithms to consider multi-threaded parallel task models [2–5] for multi-core systems.

However, in some distributed applications, the use of powerful enough multi-core processors is impossible due to Size, Weight, and Power (SWaP) constraints. But it is also possible to comply with the requirements of such computational-intensive applications by aggregating a set of embedded devices connected through an interconnection network and cooperating to achieve a common goal [1].

Modern cars are a very good example for such type of distributed systems [6,7]. They are composed of tens of computing nodes interconnected by various types of communication networks. The complexity of their workload never stops

[☆] Paper related to the special issue derived from REACTION 2013 workshop "Real-time and Distributed Computing in Emerging Applications" organized by M. García-Valls and T. Cucinotta.

[*] Corresponding author.
*E-mail addresses:* rgmaz@isep.ipp.pt (R. Garibay-Martínez), grrpn@isep.ipp.pt (G. Nelissen), llf@isep.ipp.pt (L.L. Ferreira), lmp@isep.ipp.pt (L.M. Pinho).

http://dx.doi.org/10.1016/j.jcss.2015.05.005
0022-0000/© 2015 Elsevier Inc. All rights reserved.

to increase, implying that many of their applications would gain in flexibility if they were parallelised and distributed over the system. Other examples of emerging applications are the self-localisation of autonomous vehicles [8] or collaborative robotic applications where dynamic workloads can be parallelised and distributed over different robots for the generation of a real-time 3D map [9]. In such applications, it may be required to allow networked processors to parallelise and distribute their workloads on peak situations, in order to respect their timing requirements. The fork-join Parallel/Distributed real-time model [1], which is studied in this paper, was designed to consider such execution patterns.

For a given set of applications and a given computing platform, the main challenge addressed in this work is to find a feasible allocation for tasks and messages in a way that all applications end-to-end deadlines are met. Unfortunately, this problem is known to be NP-hard [10]. Furthermore, the problem of task allocation can be viewed as a two-sided problem: (i) finding the partitioning of tasks and messages onto the processing elements of the distributed system, and (ii) finding the priority assignment for tasks and messages in that partition so that the real-time applications complete their execution within their deadline. Those two sub-problems are strongly interrelated as the decision of assigning a task to a given node should depend on the priorities of the other tasks already assigned to that node. Conversely, the priorities of tasks executing on a node might need to be adapted if new tasks are later added to that node. Therefore, a careful trade-off between the solutions of these two sub-problems needs to be taken in order to obtain an efficient global solution.

*Contribution.* In this paper, we first present the Distributed using Optimal Priority Assignment (DOPA) heuristic (of which a preliminary proposal can be found in [11]) that finds a feasible partitioning and priority assignment for distributed applications based on the linear transactional model. We then extend DOPA for the assignment of Parallel/Distributed applications and present a second heuristic called Parallel-DOPA (P-DOPA). Both DOPA and P-DOPA partition the tasks and messages in the distributed system, and make use of the Optimal Priority Assignment (OPA) algorithm, known as Audsley's algorithm [12], to find the priorities of tasks for that partition. However, the OPA algorithm requires tasks to be independent, therefore, in order to use the OPA algorithm for task sets with dependencies; we first need to transform tasks with dependencies to a set of independent tasks by imposing artificial intermediate deadlines. Two different methods for adding intermediate deadlines are presented in the paper: one for linear and one for parallel applications.

*Structure of the paper.* Section 2 presents the related work, whilst Section 3 introduces the system model. Section 4 describes the DOPA heuristic for the linear transactional model which is evaluated through simulations in Section 4.3. The P-DOPA heuristic for the Parallel/Distributed model is described in Section 5 and its evaluation is shown in Section 5.4. Finally, in Section 6 we draw our conclusions.

## 2. Related work

In this section, we review some relevant works related to the problem of allocating sequential tasks and messages in distributed systems. Also, we review some works related to multi-threaded parallel task scheduling for multi-core systems and distributed systems. Nevertheless, in both cases, we focus our attention to the case of pre-emptive fixed-priority scheduling.

The problem of allocating sequential tasks in distributed systems is usually divided in two sub-problems: (i) finding the partitioning of tasks and messages onto the elements of the distributed system (processors and networks, respectively), and (ii) finding the priority assignment for that partitioning. For example, Tindell et al. [13] addressed these issues as an optimisation problem, solving it with the general purpose Simulated Annealing algorithm. The Simulated Annealing algorithm is used for iterating in a random manner over a given allocation, and performs an evaluation based on an "energy function" that measures the quality of the encountered solution (allocation). Tindell et al. [13] used the Deadline Monotonic (DM) scheduling algorithm [14] to assign priorities to tasks.

In [15], Gutierrez et al., proposed an optimisation technique that assumes a set of tasks and messages that are statically allocated to processors and networks (therefore, no partitioning phase is considered); thus, focusing on the problem of assigning priorities to the allocated tasks and messages. Their method is based on imposing artificial intermediate deadlines to the tasks and messages and then using DM to assign the task priorities.

Richard et al. [16] proposed a solution based on branch-and-bound; enumerating the possible paths that can lead to an allocation, and cutting the path whenever a feasible schedule cannot be reached by following such a task assignment. Again, DM is used to assign the priorities assuming that each task is defined by its own deadline and period. The bounding step is performed by checking the schedulability of each branch, based on the schedulability analysis derived by Tindell et al. [17].

In [18] and [19], the authors model the task partitioning problem as an optimisation problem. However, this work still assumes that each task has its own period and deadline, and it uses DM to assign priorities.

Azketa et al. [20] addressed this problem by using general purpose genetic algorithms. They use a genetic algorithm with a permutational solution encoding. They initiate their genetic algorithm by assigning priorities using the HOPA heuristic [15] which is based on DM priority assignment [14] and iterate over different solutions by applying crossover, mutation and clustering operations. To test schedulability they use the holistic analysis presented in [17,21,22].

Research related to multi-threaded parallel fixed-priority real-time tasks has targeted mostly multi-core architectures; of interest to this work, in Lakshmanan et al. [2], the authors introduced the Task Stretch Transformation (TST) model for fork-join parallel synchronous tasks. The TST considers fork-join pre-emptive fixed-priority periodic tasks with implicit deadlines. The fork-join structure is transformed into a sequential structure, and the set of sequential fixed-priority tasks remaining after the transformation are partitioned according to the Fisher–Baruah–Baker First-Fit-Decreasing (FBB-FFD) [23] partitioning algorithm. The authors proved that the TST has a *resource augmentation bound* of 3.42. The resource augmentation bound