A greedy heuristic for distributing hard real-time applications on an IMA architecture

Émilie Bérard-Deroche, Jean-Luc Scharbarg, Christian Fraboul

A greedy heuristic for distributing hard real-time applications on an IMA architecture

Emilie Deroche*, Jean-Luc Scharbarg* and Christian Fraboul*
*University of Toulouse, IRIT-INPT/ENSEEIHT, 2, rue Charles Camichel, 31000 Toulouse, France
Email: {Emilie.Deroche, Jean-Luc.Scharbarg, Christian.Fraboul}@enseeiht.fr

***Abstract*—Current avionics architectures use complex processors, which are shared by many avionics applications according Integrated Modular Avionics (IMA) concepts. Using less complex processors on small aircraft such as helicopters leads to a distributed IMA architecture. Thus the set of partitions has to be distributed on the set of available processors. This distribution has to deal with both schedulability constraints on each processor and end-to-end latency constraints for chains of communicating partitions. Several mapping approaches exist for various applicative contexts. An approach has been proposed in the context of avionics. It implements an exhaustive analysis of all possible mappings. Time needed to perform this exhaustive analysis is drastically limited by incrementally mapping avionics functions and checking both scheduling and end-to-end constraints at each step. This approach is able to map small avionics application. However, it doesn't scale well, mainly because the scheduling space quickly explodes. In this paper, we integrate a greedy heuristic in the approach, in order to limit the scheduling space. We show that the resulting approach scales much better and gives mapping results which are close to those of the exhaustive approach.**

I. Introduction

Helicopter and aircraft industries attempt to reduce weight and power consumption. The IMA architecture is a first step in this direction: instead of having one function per processor like in federated architectures, several functions share the same processor. Most of the time, communication means are also shared to reduce the number and the weight of cables [1] [2].

Avionics systems are composed of an increasing number of more and more complex functions. It leads to avionics architectures composed of powerful and complex processors. Such processors cannot be used in small aircraft and helicopters (cost, space, ...).

A classical solution to deal with this problem is to have a larger number of (possibly less complex) processors that can be distributed among the whole helicopter. The problem is then to distribute avionics functions on these processors in such a way that timing properties are guaranteed. The first constraint is to guarantee that the set of functions allocated to a given processor is schedulable on this processor. Such scheduling is built statically in IMA: each function is allocated to dedicated slots. It corresponds to a non-preemptive off-line scheduling. The second constraint is to guarantee that end-to-end delay constraints are not exceeded: for example, the delay between the completion of the data acquisition process and the recording of the resulting data in the non-volatile crash-protected recording medium should be guaranteed [3]. When data acquisition is done by function f_1 and recording is done by function f_2, delay between start of f_1 and end of f_2 should not exceed 0.5 seconds. If f_1 and f_2 are allocated to different processors, communication delay has to be considered in the analysis. Thus distribution of avionics functions has to deal with both scheduling of partitions and end-to-end delay analysis.

The problem of mapping real-time applications on a distributed architecture has been addressed in a significant number of papers.

A first group of solutions (e.g. [4]) proceed in two steps: first, partitions are scheduled on execution nodes, second, flows are routed on the avionics network. Such approaches cannot be directly applied to our problem. It would come to, first assign all the functions to available processing elements in such a way that at least one valid scheduling exists for each processing element, second assess that end-to-end delay constraints are satisfied. First problem is that the number of candidate allocations explodes when the number of functions increases. Second problem concerns scheduling problem, which is NP-complete in the strong sense, even in the case of a single processor [5].

A second group of solutions search for an allocation which minimizes the communication costs. In [6], authors propose an algorithm for allocating partitions with an off-line real-time scheduling, taking into account the traversal time of communications with a specific protocol (Token Ring): communicating subsystems are clustered to minimize the bus traffic and to increase schedulability. In [7], the aim is to cluster the dependent tasks onto the same machines such that the network communication is minimized. Such approaches put the focus on the communication cost, which is only a minor issue in our case. A more important feature to be considered for our problem is the value of each end-to-end delay constraint (long or short).

A third group of solutions implement integrated approaches where scheduling of partitions and end-to-end delay constraints are considered together. Such approaches are the most promising ones for our problem, since scheduling of partitions and end-to-end delay constraints are cross-dependent. In [8], the allocation problem is addressed in the context of automotive. Based on a MILP formulation, tasks are allocated, signals are mapped to messages and priorities are assigned

to tasks and messages. In [9], [10], the focus is put on the extensibility or the flexibility of the obtained allocation. Those approaches consider a run-time priority-based scheduling of tasks as opposed to IMA case where a strictly periodic static scheduling is built offline. The work presented in [11] considers non-preemptive distributed scheduling problems with dependencies, strict periodicity constraints and fulfilment of end-to-end delay constraints. However, it considers synchronized processors, which is not the case for civilian avionics.

An integrated approach, dealing with IMA features, has been proposed in [12]. The main idea of this approach is to check both schedulability and delay constraints on-the-fly, i.e. each time a partition is assigned to a processing element. It comes to validate partial allocations, where only a subset of the partitions is allocated. Thus, invalid allocations are early eliminated. It has been shown in [12] that this approach is able to distribute small industrial case studies and that it scales well in a reasonable amount of cases. However, it does not work when the search space for scheduling is too large. It occurs when a large number of functions with tight delay constraints have to be assigned to a number of processors which is hardly sufficient. Such a situation will occur in the context of avionics if we want to distribute a significant part of the redundant avionic functions on a limited number of processors.

The main contribution of this work is to propose and evaluate a heuristic which drastically limit the search space for scheduling. The basic idea is to select candidate scheduling, based on metrics taking into account delay constraints.

The rest of the paper is organized as follows: next Section develops the problem statement and gives main modelling assumptions. Section III explains the heuristic algorithm we propose. Section IV evaluates proposed approach on case studies and compares it with a previous algorithm. Finally, section V concludes the paper and proposes directions for future work.

II. Problem statement

A. System model

We consider an avionics application defined by a set of n communicating partitions $\mathcal{P} = \{P_1, ..., P_n\}$ which will be executed on a set of at most $maxNb_{PE}$ processing elements $\mathcal{E} = \{PE_1, ..., PE_{maxNb_{PE}}\}$. In the context of this paper, we consider that all processing elements are identical. This is reasonable assumption in our context. Nevertheless, the proposed approach can be easily extended to the case with heterogeneous processors.

A partition P_i is characterised by its period T_i and Worst Case Execution Time (WCET) C_i on any of the (identical) processors. Classically, we have:

$$0 \leq C_i \leq T_i \quad (1)$$

Each partition P_i is strictly periodic: the duration between two consecutive executions is exactly T_i as illustrated in Figure 1. Periods are assumed harmonic. This is a classical assumption for avionics systems.

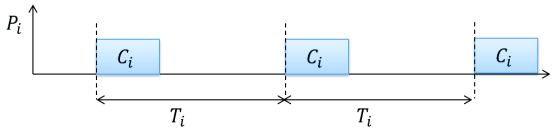

Fig. 1. Strictly periodic partition

A set $\mathcal{CH} = \{ch_1, \ldots, ch_m\}$ of m communication chains is associated to the partitions. Each chain $ch_j = \{P_{j_1}, \ldots, P_{j_k}\}$ indicates that data are transmitted from source partition P_{j_1} to destination partition P_{j_k} through intermediate partitions P_{j_l} $(1 < l < k)$. A delay constraint $D_{ch_j,max}$ is associated to each chain ch_j. It defines the maximum allowed delay between the start of source partition and the end of corresponding destination partition (end-to-end delay constraint). In this paper, we assume that delay constraints follow the Button-To-Action (BTA) semantic defined in [13]. BTA semantic is illustrated in Figure 2. The end-to-end delay is measured from the generation of a data by the first partition in the chain to its first utilization by the last partition in the chain. Other end-

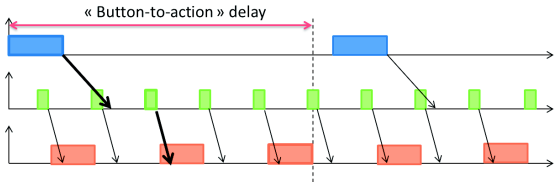

Fig. 2. Button-to-Action semantic

to-end delay semantics have been proposed in [13], e.g. age delay which considers the last utilization of the data by the destination partition. In the context of avionics applications, BTA is the most commonly used semantic. The approach proposed in this paper can deal with any end-to-end delay semantic.

Let's illustrate communication partition features with the example in Table I. It includes six

TABLE I
Real-time specifications of partitions of the illustrative example

Partitions	C_i (ms)	T_i (ms)
P_1	3	10
P_2	2	10
P_3	2	20
P_4	4	40
P_5	1	40
P_6	4	40

partitions with the following communication chains:
$ch_1 = \{P_1, P_2, P_3\}$ with $D_{ch_1,max} = 30$ ms
$ch_2 = \{P_2, P_5\}$ with $D_{ch_2,max} = 40$ ms
$ch_3 = \{P_4, P_5, P_6\}$ with $D_{ch_3,max} = 60$ ms

Each partition implements a read-execution-write semantic: at the beginning of its execution, the partition reads data from its input ports; at the end of its execution, it writes data in its

output ports. Partitions exchange data through virtual channels (APEX channel in ARINC 653 [14]). This is illustrated in Figure 3.

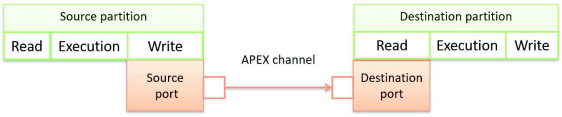

Fig. 3. APEX channel

Sampling ports are assumed: at any time only the last data written in the buffer is considered.

Scheduling of the partitions allocated to a processing element PE_l is defined by IMA. It is based on the off-line construction of a MAjor Frame MAF_l, which statically defines the periodic execution pattern of all the partitions allocated to PE_l. Such a MAF is shown in Figure 4. It considers the example in Table I, where partitions P_1 and P_3 are allocated to processing element PE_1. Since partition periods are harmonic, the duration $DUR(MAF_l)$ of MAF_l is the largest period among partitions allocated to PE_l. In Figure 4, $DUR(MAF_1)$ is the period of P_3, i.e 20 ms. Indeed, we have $T_1 = 10$ ms and $T_3 = 20$ ms.

A MAF is a sequence of intervals with equal duration. The duration $DUR(int_l)$ of an interval of MAF_l is the smallest period among partitions allocated to processing element PE_l. In Figure 4, MAF_1 is composed of two intervals of 10 ms, the period of P_1.

Each partition P_i allocated to a processing element PE_l is allocated one slot every $\frac{T_l}{DUR(int_l)}$ interval. The duration of this slot is C_i, i.e. P_i WCET. Since partition executions have to be strictly periodic, slots allocated to a given partition have to be periodic. In Figure 4, P_1 is allocated one slot per interval, while P_3 is allocated one slot every two intervals.

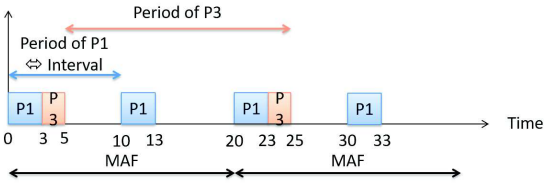

Fig. 4. MAF illustration

r_i defines the start time of the first slot allocated to partition P_i in MAF_l. In the example in Figure 4, we have $r_1 = 0$ and $r_3 = 3$

B. Allocation process

The goal is to find all valid allocations of a set of partitions $\mathcal{P}$ with communication chains $\mathcal{CH}$ on up to $maxNb_{PE}$ processing elements. An allocation is valid iff there exist at least one set of MAF (one per processing element) such that each partition is periodically scheduled on its allocated processing element and end-to-end delay constraints are met.

The allocation process is based on the approach presented in [12]. Schedulability as well as end-to-end constraint verifications are done on-the-fly, i.e. after each allocation of a partition. The idea is to early eliminate groups of invalid allocations and, thus, to limit the search space.

This approach is illustrated on the example in Table I. A tree including all possible allocations is covered in a depth-first manner. A part of this tree is shown in Figure 5. The process starts with an empty allocation (root of the tree). First, P_1 is allocated to a first processor (it can be any processor, since they all are identical). On the next level of the tree, next partition P_2 can be assigned to the same processor as P_1 (left son) or to a different one (right son). In each node, both schedulability of partitions already allocated to processors and end-to-end delay constraints are verified. It comes to find a valid set of MAFs (one MAF per processor), i.e. e set of MAFs such that no end-to-end constraint is violated. This set of MAFs is built by adding the partition allocated in the current level of the tree to the valid set of MAFs built in the father node. If a valid set of MAFs is found in a given node, then, the process moves to the next level in the tree where possible allocations for the following partition are tested (for instance, in figure 5, P_3 is considered after P_2). If no valid set of MAFs is found, the process backtracks to the father node where it searches for another valid set of MAFs. A valid allocation is found in each node where a valid set of MAFs is successfully built and all the partitions have been allocated. It corresponds to lower level nodes in the tree (after the allocation of P_6 in figure 5).

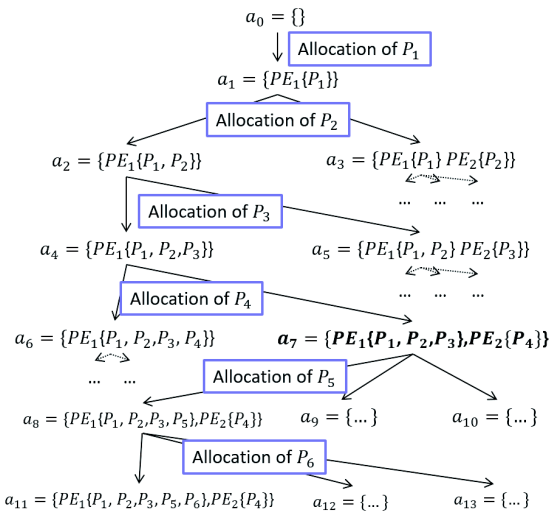

Fig. 5. Tree cover

End-to-end delay computation for a given chain has been detailed in [12]. An end-to-end delay is composed of worst-case execution time of partitions and worst-case durations between these executions. Two cases have to be considered when computing the worst-case duration between two consecutive partitions of a chain:

- when both partitions are allocated on the same processor, this worst-case duration only depends on the MAF of this processor; it is the largest duration between consecutive executions of the two partitions (we assume sampling ports);
- when partitions are allocated on different processors, Worst-Case Traversal Time between the two processors as well as waiting delay on the destination processors are added; since processors are not synchronized, the worst-case waiting delay is the period of the second partition.

When a partition of a chain is not yet allocated, only its WCET is considered (null duration between its execution and the executions of the preceding and following partitions in the chain).

All the details of this end-to-end delay computation can be found in [12].

It should be noticed that, in the context of this paper, we consider a unique worst-case traversal time (WCTT) for all the flows and we assume that it has been predetermined before starting the allocation process. Since this WCTT has to capture all possible allocations, it is pessimistic. The computation is based on existing worst-case end-to-end delay analysis approaches, e.g. the ones presented in [15] for avionics switched Ethernet networks. Such approaches are quite fast since they provide results in a few seconds on a configuration with one thousand flows. However, running such an approach in each node of the allocation tree will not scale.

C. MAFs construction

The problem is to schedule off-line a set of strictly periodic partitions with harmonic periods. Korst and al. show that the problem of non-preemptively scheduling periodic tasks is NP-complete in the strong sense in the case of a single processor, but that it is solvable in polynomial time if the periods are harmonic [5]. It has been shown in [16], [17] that a valid scheduling of tasks with harmonic periods on a single processor exists iff one such scheduling based on the bin structure exists. Thus it comes to schedule tasks by increasing periods in bins. The bin size corresponds to the smallest period among tasks and the number of bins is equal to the ratio between the longest and the smallest period. Indeed, since periods are harmonic, the least common multiple (LCM) of the periods is the largest period among tasks.

Our MAF construction is based on this algorithm, since it fits with our problem. We illustrate this MAF construction on the example in Table I. Part of the tree of considered allocations is shown in figure 5. The unique WCTT is equal to 5 ms in this example.

Let's consider the node where partitions P_1, P_2 and P_3 have been allocated to first processor and P_4 is allocated to second one (bold text). A valid scheduling of these four partitions on first and second processors is shown in Figure 6 (a). At this point, P_5 can be allocated to the first processor, the second one or another one. Let's assign it to first one. Figure 6 (b) depicts a first possible scheduling, where P_5 is placed in the first available interval (immediately after P_3). The obtained MAF set is valid, since end-to-end delay constraints are satisfied. Thus the process advances to P_6 allocation. A first solution is to assign P_6 to the first processor. In this case, P_6 can be scheduled in the second (Figure 6 (c)) or fourth interval in the first processor MAF. However, both solutions lead to ch_3 end-to-end constraint violation. It means that there is no valid MAF set in this node of the tree. Thus the process backtracks and searches for another valid MAF set when P_1, P_2, P_3 and P_5 are allocated on the first processor and P_4 is allocated on the second one. Such a valid MAF set is depicted in figure 6 (d). P_5 is moved to the second interval. Then the process tries again to allocate P_6 to the first processor. A valid MAF set is found. Thus, the current node corresponds to a valid allocation.

III. Heuristic for valid MAF set selection

It has been shown in [12] that such an approach is able to distribute small industrial case studies and that it scales well in a reasonable amount of cases. However, it doesn't work if we want to distribute a large number of redundant avionic functions on a limited number of processors. This is due to the fact that the number of MAF sets to be tested explodes.

In order to deal with this explosion, we propose to eliminate backtracking, using a greedy heuristic. The idea is to choose the most promising valid MAF set in each node of the tree and to never look for another valid MAF sets in this node.

The choice of the most promising valid MAF set is based on the potentialities of each valid MAF set for the allocation of the remaining partitions. In this paper, we consider that the potentialities of a valid MAF set are based on the sum f_c of margins for communication chains in the current allocation:

$$f_c = \sum_{ch_i \in \mathcal{CH} \subset A} ma_{ch_i,A} \tag{2}$$

where

$$ma_{ch_i,A} = D_{ch_i,max} - D_{ch_i,A} \tag{3}$$

The margin of a communication chain is the difference between the chain constraint and the current end-to-end delay $D_{ch_i,A}$ of each chain ch_i. The aim of this metric is to choose the scheduling which maximizes the flexibility for allocating remaining partitions belonging to communication chains.

Coming back to the example in Figure 6, at the point when partition P_5 is allocated to processor one, schedulings in Figure 6 (b) and (d) are both candidates. Potentialities of the first scheduling is:

$$f_c = (30 - 17) + (40 - 35) + (60 - 54) = 24$$

and potentialities of the second one is:

$$f_c = (30 - 17) + (40 - 33) + (60 - 54) = 26$$

Thus, the second scheduling is selected. As previously explained, it leads to a valid total allocation.

In [12], partitions were processed by increasing periods. The goal was to limit the search space for candidate schedulings at upper levels in the tree. Indeed, allocating partitions with

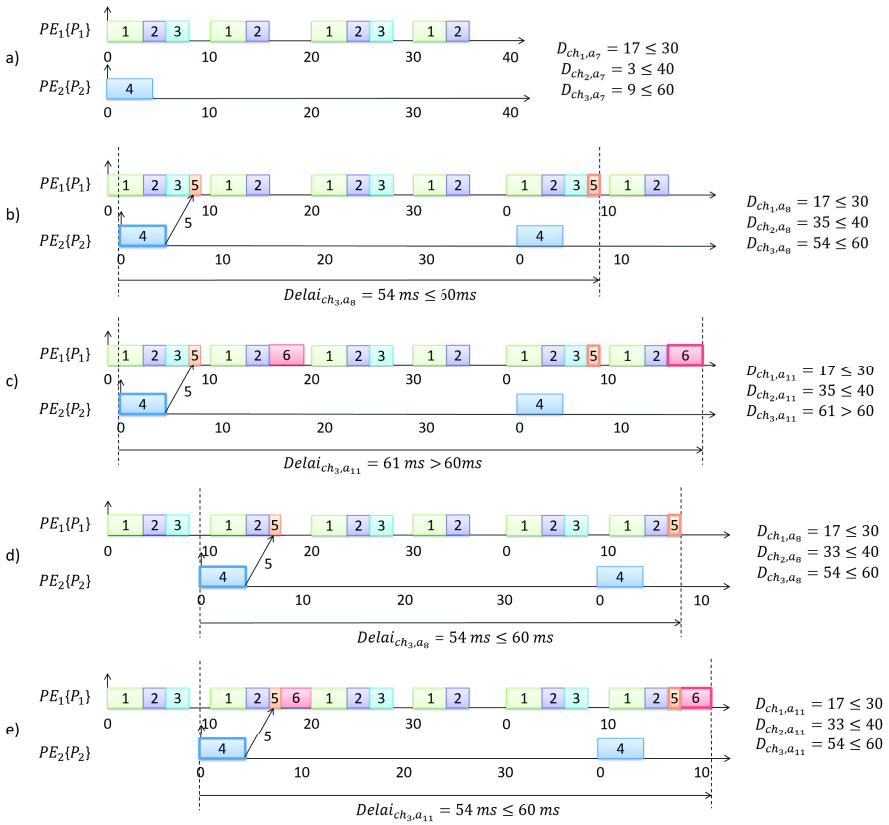

Fig. 6. Scheduling search

large periods increases the size of the MAF, leading to a much larger number of candidate schedulings. Delaying as much as possible the allocation of partitions with large periods also limits the increasing of candidate scheduling.

In the approach presented here, the problem is a bit different, since only one scheduling is selected in each node, without backtracking. It follows that the number of schedulings considered by the selection process in each node remains very small, even for large MAFs. Thus, limiting MAF size in early steps of allocation process will not bring significant improvement in terms of execution time.

Conversely the heuristic for valid MAF set selection is based on margins for communication chains. Thus starting the allocation process with partitions belonging to communication chains with small margins should bring more flexibility in the last steps of the allocation process (when the last partitions have to be allocated). Consequently, partitions are sorted by increasing chain margin. At the beginning of the process, the margin of a communication chain is its end-to-end delay constraint minus the worst-case execution time of all its partitions. Then partitions are sorted in such a way that, if a partition P_i belongs to one chain which has a smaller margin than all the chains including partition P_j, P_i is processed before P_j.

IV. Case studies

The heuristic approach proposed in this paper and the exhaustive integrated one in [12] have been implemented. Two comparisons have been conducted. First one considers a small helicopter application. Second one considers various larger arbitrary configurations.

First goal is to measure to which extend the heuristic approach misses valid allocations which are found by exhaustive one. Second goal is to measure the gain brought by the heuristic approach in terms of execution time.

The worst-case traversal time for a flow a assumed to be 1 ms.

A. Helicopter application

We consider the Vehicle Monitoring System whose applicative architecture is summarized in Figure 7 [18]. It includes seven partitions and six communication chains (two of them are shown in Figure 7). This application is replicated three times for redundancy reasons.

Up to now, each replica is implemented on one powerful processor. The aim is to use less complex processors which should be distributed in different locations in the helicopter. Figure 8 shows 4 candidate allocations for one replica, using between one and four processors (many other allocations can be envisioned). We can see that, due to applicative constraints,

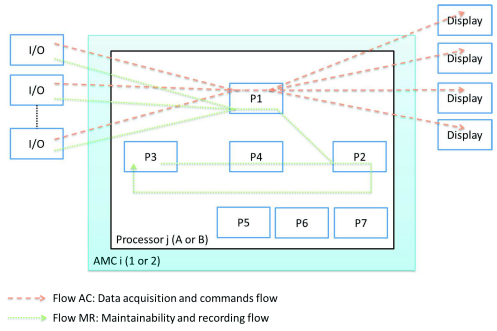

Fig. 7. Main flows of communicating partitions

partitions P_5, P_6 and P_7 have to be replicated on all processors. Indeed, these partitions collect log information from other partitions.

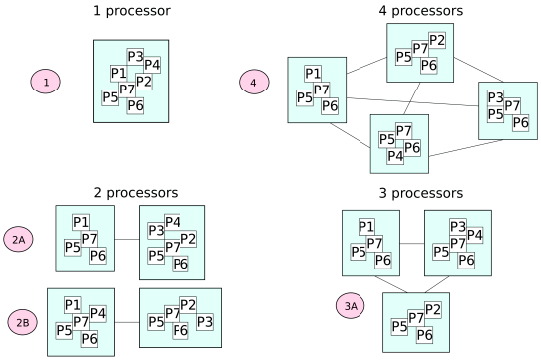

Fig. 8. Possible allocation schemes

Table II gives the period of each partition as well as its WCET on the envisioned less powerful processor.

TABLE II
REAL-TIME SPECIFICATIONS OF PARTITIONS OF THE VEHICLE MONITORING SYSTEM

Partitions	C_i (ms)	T_i (ms)
P_1	10	25
P_2	10	50
P_3	6	100
P_4	6	50
P_5	5	100
P_6	2	100
P_7	4	100

Both exhaustive and heuristic approaches have been tested on this application, taking into account the replicas. For the exhaustive one, two cases have been considered: in the first one, partitions are processed by increasing period. In the second one, a random order is used.

All approaches find 625 valid allocations. It means that, for this specific system, the heuristic approach misses no solution. Thus, the selection of a valid MAF set in each node is pertinent.

Figure 9 shows the execution time and the memory needed by each approach to find the first valid allocation, while Figure 10 gives the same information for the full process (find all valid allocations). All approaches get the first valid allocation in less than 100 ms with similar memory usage. For the full process, the heuristic approach outperforms the exhaustive one, both in terms of execution time (11 seconds versus 162 when partitions are sorted by increasing periods and 381 when they are not) and memory usage. This is due to the drastic limitation of considered valid MAF sets.

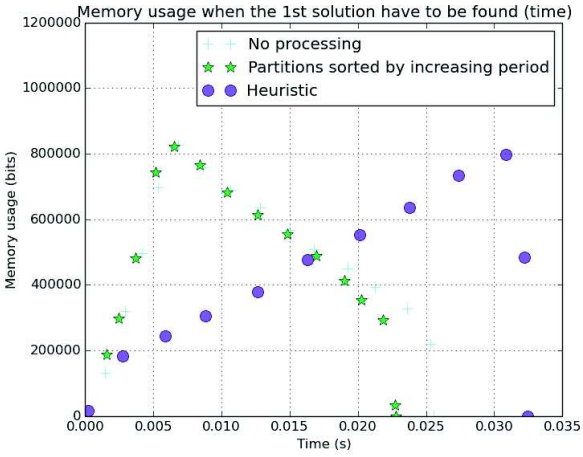

Fig. 9. Time and memory, first solution

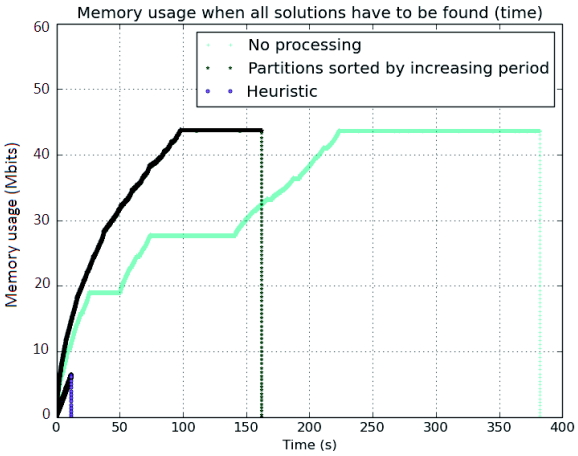

Fig. 10. Time and memory, all solutions

B. Arbitrary configurations

Configurations including various number of partitions are considered. The first one includes 30 partitions. Each partition has a period $T = 25\ ms$ and a WCET $C = 5\ ms$. There are 15 communication chains. Each one includes two partitions. End-to-end delay constraint is 20 ms for each chain.

The partitions are allocated to a set of 2 to up to 10 processing elements.

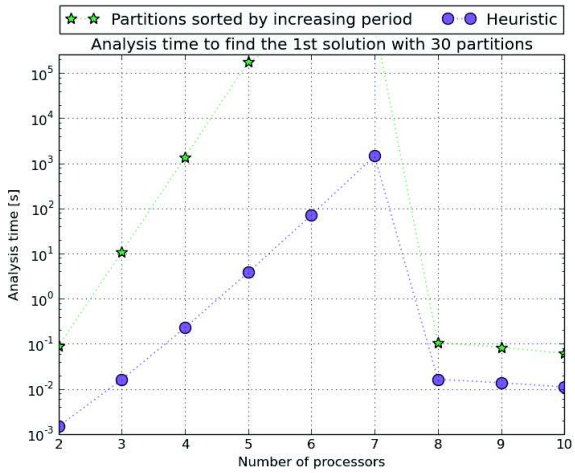

Fig. 11. Both approaches, first solution, $D_{ch_i,max} = 20ms$

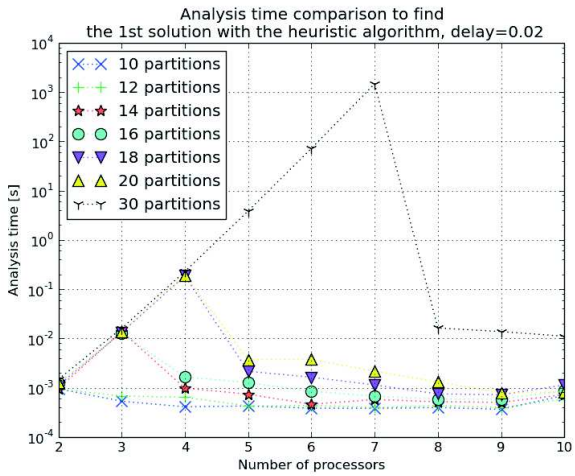

Fig. 12. Heuristic approach, one solution, $D_{ch_i,max} = 20ms$

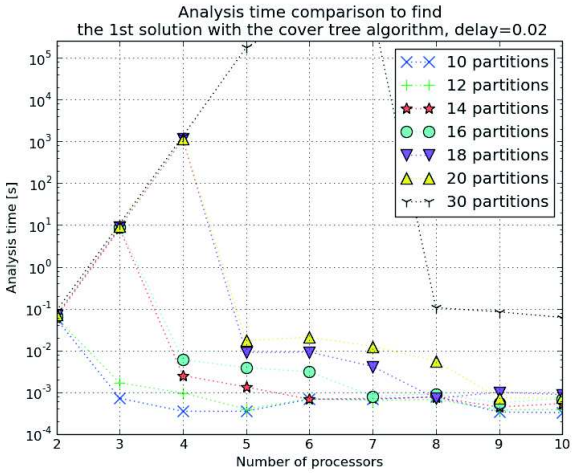

Fig. 13. Exhaustive approach, one solution, $D_{ch_i,max} = 20ms$

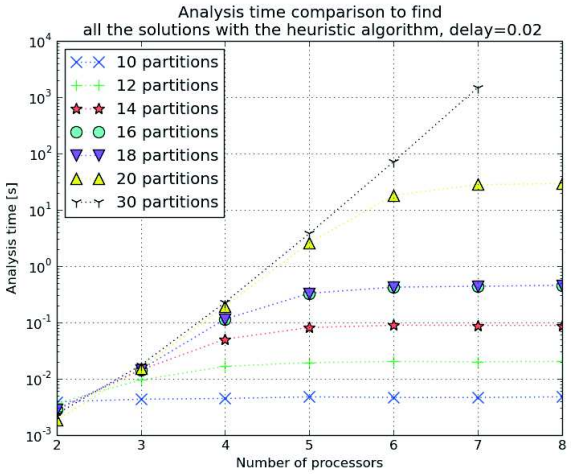

Fig. 14. Heuristic approach, all solutions, $D_{ch_i,max} = 20ms$

First we use both exhaustive and heuristic approaches to find the first valid allocation. For the exhaustive one, we consider that partitions are processed by increasing periods. Both approaches find the same first solution, which means that, once again, valid MAF set selection has no impact on the result. Figure 11 shows that the heuristic approach significantly reduces the execution time: when the limit on the number of processors is 5, the execution time is 177 026 seconds for the exhaustive approach and 3 seconds for the heuristic one. We can observe that the execution time increases with the limit on the number of processors until this limit is 7. Then it decreases. This is due to the fact that nodes in the tree are visited from right to left. The goal is to maximize the number of processors for the distribution of the configuration. On this specific configuration, a valid allocation is quickly found when at least 8 processors are available. This is not the case with less processors.

Similar results are obtained with different numbers of partitions as depicted in Figures 12 (heuristic approach) and 13 (exhaustive approach).

When searching for all valid allocations, the exhaustive approach does not finish in a reasonable time while the heuristic one find all valid allocations in less than 1000 s for all tested configurations, except the one with 30 partitions on up to 8 processors.

Figure 15 shows that the execution time needed to find all valid allocations significantly increases when chain constraints are relaxed to 40 ms. This is due to the fact that the number of valid allocations drastically increases. Thus very few branches are pruned and the algorithm has to visit most of tree nodes. However, the heuristic algorithm always terminates in less than 4000 seconds.

All these results show that the heuristic approach scales much better than the exhaustive one. Thus it is a very promising solution for distribution of avionics applications, since the

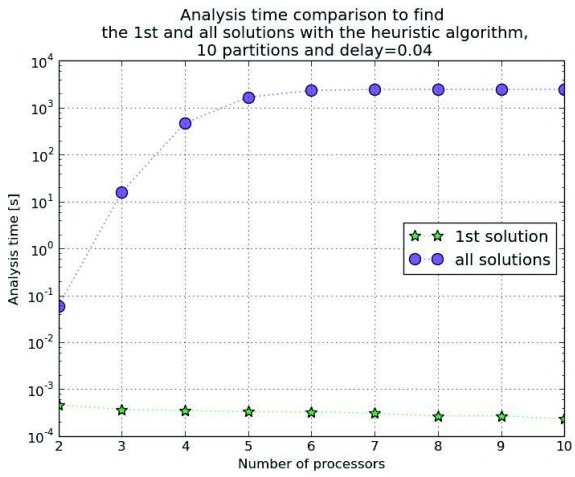

Fig. 15. Analysis time comparison to find the first and all solutions with the heuristic algorithm, 10 partitions $P_i = (25, 5)$ and $D_{ch_i,max} = 40ms$

size of these applications usually does not exceed the size of configurations we have considered in this paragraph. On all tested configurations, the heuristic approach finds the same valid allocations as the exhaustive one. It means that selecting a valid MAF set based on the margin of communication chains is a good solution. Actually, end-to-end communication constraints are the most difficult ones to be satisfied, especially because processing elements are not synchronized.

V. Conclusion

The goal of this paper was to propose a scalable approach for the distribution of avionics application composed of a set of partitions on a set of processors. The proposed approach is based on the exhaustive one proposed in [18]. It checks both schedulability and end-to-end delay constraints on-the-fly, i.e. each time a partition is allocated to a processor. Main contribution of this paper is to propose a greedy heuristic which drastically limits the search space for scheduling. At each step, the most promising scheduling is selected and all other valid schedulings are eliminated. The selection is based on the available margin for communication chains.

We show on a small helicopter application and on various larger configurations that this greedy heuristic drastically reduces the execution time without missing valid solutions.

Thus, the proposed approach is a promising candidate for the distribution of avionics applications.

Next step is to integrate inputs and outputs, such as displays, sensors and actuators, in the distribution process. This can be done by considering specific processing elements where these inputs and outputs are pre-allocated. Then the distribution process starts from a non-empty allocation. We can guess that it will reduce the search space, since additional constraints on communication chains integrating inputs and outputs should lead to early elimination of invalid partial allocations.

Another important problem concerns the choice of one allocation in the obtained set of valid ones. Flexibility and extensibility are very important in the context of avionics, since we don't want to modify the whole allocation when the application slightly changes. Interesting solutions are proposed in [9] and [10]. We plan to investigate their integration in our approach.

References

[1] P. Bieber, F. Boniol, M. Boyer, E. Noulard, and C. Pagetti, "New challenges for future avionic architectures," *Journal Aerospace Lab*, vol. 4, May 2012.

[2] J. Moore, *Digital Avionics Handbook, Second Edition - 2 Volume Set*. CRC Press 2000, 2001, ch. Advanced distributed architectures, pp. 33–1–33–12.

[3] *ED-112A: Minimum operational performance specification for crash protected airborne recorder systems*, EUROCAE, EUROCAE Std., September 2013.

[4] A. Al Sheikh, "Resource allocation in hard real-time avionic systems. scheduling and routing problem," Ph.D. dissertation, EDSYS, September 2011.

[5] J. Korst, E. Aarts, and J. K. Lenstra, "Scheduling periodic tasks," *INFORMS Journal on Computing*, vol. 8, no. 4, pp. 428–435, 1996.

[6] K. Tindell, A. Burns, and A. Wellings, "Allocating hard real-time tasks: An np-hard problem made easy," *Real-Time Systems*, vol. 4, no. 2, pp. 145–165, 1992.

[7] C. Ekelin and J. Jonsson, "A lower-bound algorithm for minimizing network communication in real-time systems," in *Parallel Processing, 2002. Proceedings. International Conference on*, 2002, pp. 343–351.

[8] Q. Zhu, H. Zeng, W. Zheng, M. Di Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Transactions on Embedded Computing Systems*, vol. 4, December 2012.

[9] P. Emberson and I. Bate, "Stressing search with scenarios for flexible solutions to real-time task allocation problems," *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 704–718, October 2010.

[10] Q. Zhu, Y. Yang, M. Di Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the software architecture for extensibility in hard-real-time distributed systems," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 621–636, November 2010.

[11] O. Kermia, "Ordonnancement temps rel multiprocesseur de tches non-premptives avec contraintes de prcdence, de priodicit stricte et de latence," Ph.D. dissertation, 2009, thse de doctorat dirige par Sorel, Yves Informatique Universit de Paris-Sud. Facult des Sciences d'Orsay (Essonne) 2009. [Online]. Available: http://www.theses.fr/2009PA112291

[12] E. Deroche, J.-L. Scharbarg, and C. Fraboul, "Mapping real-time communicating tasks on a distributed ima architecture," in *Emerging Technologies Factory Automation (ETFA), 2016 IEEE 21th Conference on*, Sept 2016, pp. 1–8.

[13] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics," *Proceedings of the IEEE Real-Time System Symposium - Workshop on Compositional Theory and Technology for Real-Time Embedded Systems, Barcelona, Spain, November 30, 2008*, 2008.

[14] *Avionics application software interface, part 1 - Required services, ARINC specification 653P1-3*, Aeronautical radio, Inc., Aeronautical radio, Inc. Std., November 2010.

[15] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *IEEE transactions on industrial informatics*, vol. 6, no. 4, pp. 521–533, Novembre 2010.

[16] F. Eisenbrand, K. Kesavan, R. S. Mattikalli, M. Niemeier, A. W. Nordsieck, M. Skutella, J. Verschae, and A. Wiese, "Solving an avionics real-time scheduling problem by advanced ip-methods," in *European Symposium on Algorithms*. Springer, 2010, pp. 11–22.

[17] F. Eisenbrand, N. Hähnle, M. Niemeier, M. Skutella, J. Verschae, and A. Wiese, *Scheduling Periodic Tasks in a Hard Real-Time Environment*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 299–311.

[18] E. Deroche, J.-L. Scharbarg, and C. Fraboul, "Performance evaluation of a distributed ima architecture," in *Work-in-Progress Proceedings, Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on*, Sept 2015, pp. 17–20.