Functional Latency Optimization for networked IMA system

Matthias Houssin
ONERA
ISAE-SUPAERO
Toulouse, France
matthias.houssin@onera.com

Oana Hotescu
ISAE-SUPAERO
University of Toulouse
Toulouse, France
oana.hotescu@isae-supaero.fr

Frederic Boniol
ONERA
Toulouse, France
frederic.boniol@onera.com

ABSTRACT

Modern cyber-physical systems are designed in networked distributed architectures that allow resource sharing and time partitioning among software functions of different computing modules. The distributed computing paradigm introduces functional chains, *i.e.*, ordered sets of tasks allocated to different modules that share data. In this paper, we aim at formalizing the allocation of tasks and network messages as an optimization problem that minimizes the time between the entry of a data in a functional chain and the last corresponding output (*i.e.,* the age delay). The formulation is based on the Periodic Scheduling Problem (PSP), a classic NP-hard optimization problem often solved with Satisfiability Modulo Theory (SMT). We consider the context of Integrated Modular Architecture (IMA)with time-triggered network communication.

1 INTRODUCTION

The Integrated Modular Architecture (IMA) paradigm has been largely adopted in the computing modules of many cyber-physical systems. Depending on the application domain, frameworks such as AUTomotive Open System ARchitecture (AUTOSAR) [6] and ARINC653 [1] have been proposed to standardize IMA principles.

The idea under IMA is to define a computing platform composed of general-purpose processing modules, operating systems and interfaces for exchanging data between different processing modules. A processing module can be shared by different applications from different functions. The execution of applications on the distributed architecture implies choosing an allocation for the applications.

Communication between computing modules is provided by one or more networks. A recent trend in the automotive and aerospace industry is to interconnect IMA modules with Time Sensitive Networking (TSN) [7]. TSN is a high data rate and low-latency Ethernet-based standard for real-time communication among mixed-critical applications. It ensures deterministic service thanks to accurate time synchronization, stream reservation and scheduling of time-sensitive data. Time-triggered link access is provided to transmit the scheduled traffic with a Time Aware Shaper (TAS) implemented in the output ports of TSN switches. The TAS requires the configuration of a system-wide transmission schedule defining static time intervals (windows) during which transmission of time-critical traffic can have exclusive access to the port queue and cannot encounter any interference from unscheduled messages.

These enhanced mechanisms provided by TSN introduce new communication opportunities in IMA systems. To take advantage of the full potential of IMA-TSN systems, the system configuration needs to be coupled to the network behaviour. Therefore, appropriate strategies for service policies, routing, tasks allocation and messages schedule need to be designed to achieve end-to-end optimal performances. In this paper, we formulate the problem of finding an optimal allocation of tasks and network messages in IMA-TSN systems. The optimization problem is formulated as a Periodic Scheduling Problem (PSP) [10] with the objective to minimize functional latency such as the reaction delay for data transmitted in functional chains.

This paper is organized as follows. First, we survey related work in Section 2 and present background on IMA in Section 3. Then, we introduce the system-network model in Section 4. In Section 5, we formulate the optimization problem and in Section 6, we provide a preliminary evaluation. Finally, in Section 7, we conclude the paper and propose leads for future work.

2 RELATED WORK

Finding optimal allocations and schedules in IMA and in the real-time network is not a new problem. Considering the large adoption of distributed computing, either on multi-core or interconnected platforms, several works defining optimal allocations and scheduling have been proposed. The work in [11] proposed SMT-based resolution for the scheduling problem in a time-triggered network. The limits of this approach is the high complexity of the resolution as highlighted in [5]. The survey in [10] lists some of these existing works and presents the main optimization criterion, common constraints and resolution methods (mainly SMT and meta-heuristics) for the allocation and scheduling problem. The end-to-end latency constraint is an important constraint and is the main interest in our paper. Our work considers multi-dependency functional chains, which is only done in [8] with a very simplified network behaviour, compared to other works such as [9] that are limited to mono-dependency functional chains.

3 BACKGROUND

The concept of the Integrated Modular Architecture has been proposed in embedded domains involving more and more safety-critical functions. To reduce the weight and number of computing resources, the IMA principle relies on two main ideas: (1) resource and (2) temporal partitioning.

Resource partitioning means that each software function is allocated a set of spatial resources (CPU core, memory area, DMA channel, etc.) in a static manner, meaning that the resource integrator has to assign the maximum allowed resources to each function while respecting space segregation between them. Operating system mechanisms provide protection for function data against any modification from the other functions.

Temporal partitioning means that the scheduling of software functions on each CPU core is defined off-line by a periodic sequence of slots statically organized in a time-frame named the MAjor time Frame (MAF). Each function is allocated a time slot for execution. At the end of this time slot, if not completed, the

function is suspended and execution is given to the next function (according to the time-table).

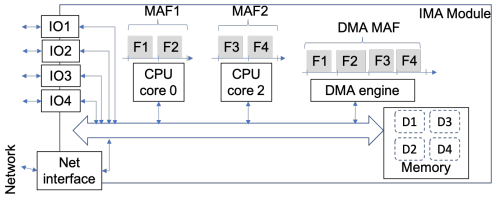


Figure 1: Example of IMA computing module

To illustrate the IMA principle, let us consider the architecture shown in Figure 1. It describes an IMA module consisting of two CPU cores, a single memory, four IO ports and one DMA engine. The module hosts four functions: F_1 and F_2 on core 1, managed by MAF1, F_3 and F_4 on core 2, managed by MAF2. The memory is segregated into four parts. Each function is allocated one IO port. Finally, the single DMA engine is shared by the four functions according the local DMA MAF. Data transfert of F_1 are done during the time slot associated to F_1 and so on.

4 MODEL

Next, we introduce a system-network model unifying tasks and messages. The model is organized in three layers. First, the hardware elements are introduced at the platform layer (subsection 4.1). The software elements (tasks and messages) are abstracted as generic flows at the system layer (subsection 4.2). The last layer models the latency properties to be evaluated (subsection 4.3).

4.1 Platform model

The first layer of the model describes the hardware view of the platform. Follow the approach developed in [5], an IMA system-network platform can be seen as a set of abstract *nodes* modeling in a unified way the computing modules and switches of the platform, connected through a set of abstract *links* modeling the processing and the networking components of the platform.

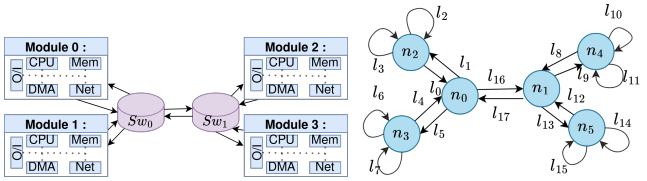
Definition 4.1 (Platform). Let P be an IMA system-network platform. P is defined by $P :=< Nodes, Links >$ where

- $Nodes$ is the finite alphabet of nodes of P (the IMA modules and the network switches),
- and $Links$ is the finite alphabet of links in P (denoting the CPUs, the DMAs, and then network links).

Example 4.2. Let us consider the platform depicted Figure 2a. This platform is composed of four modules and two switches. Figure 2b describes the abstract model of the platform. The CPU and DMA components of Module 0 are modeled by the links l_2 and l_3 as such l_2 and l_3 are looping links from node n_2 to n_2.

Definition 4.3 (Node). Let P be a platform. Let $n \in P.Nodes$ denote either a switch or a module.

Definition 4.4 (Link). Let P be a platform. Let $l \in P.Links$, where l denotes a processing or a networking element inside a node or from a node to another one. Processing elements are CPU or DMA. Links are defined by $l :=< src, dst, d_p, q, type >$ where



(a) Real platform **(b) Modeled platform**

Figure 2: Platform example

- $src \in Nodes$ is the source node of the link;
- $dst \in Nodes$ is the destination node of the link;
- $d_p \in \mathbb{Q}$ is the link propagation delay;
- $q \in \mathbb{Q}$ is the link data-rate;
- $type \in \{CPU, DMA, Net\}$ indicates if the link represents a CPU, a DMA or a network link.

4.2 System Model

The second layer of the model involves the tasks and the messages hosted by the platform.

Definition 4.5 (Task model). Following [3, 5], a task hosted by a IMA module is a software object defined as $\tau :=< T, C, D >$ where $T \in \mathbb{Q}$ is the task period, $C \in \mathbb{Q}$ is the worst-case execution time (WCET), and $D \in \mathbb{Q}$ is the deadline.

Definition 4.6 (Message). A message $m :=< len >$, where $len \in \mathbb{N}$ is the message length, is an object sent through the network by a task τ.

Definition 4.7 (Flow). A flow f is defined as $f :=< \tau, m >$. It denotes the periodic action of a task τ sending a message m.

Definition 4.8 (Flow-tree). The instances of a flow f transit links in the platform. A task often transmits the same data to multiple tasks using a path that can be described as a tree. The allocation of links to a flow f is defined as a flow-tree function $ft : f \mapsto \{(l, l')\} \subset P.Links \times P.Links$. The root of the flow-tree, $root(ft(f))$, is the starting point of any flow instance (necessarily a CPU link). The last links transited by f in the tree are the leaves of the flow-tree defined as $leaves(ft(f)) := \{l | l \in Im(ft(f)) \wedge \forall l' \in Im(ft(f)), (l, l') \notin Im(ft(f))\}$.

Definition 4.9 (Image of a relation). Consider the flow-tree as an example of relation, we define the image of comparable relations:

$$Im : X \mapsto \bigcup_{(x_1,x_2)\in X} \{x_1, x_2\}$$

This definition can be applied to any set of pairs of objects.

For a given platform P, let us denote:

- $Flows$ as the set of flows hosted by P;
- f_k^l is the k-th instance of f considered on link l. It may be either an instance of task or message;
- $I(Flows) = \{f_k^l | f \in Flows, l \in Im(ft(f)), k \in \mathbb{N}\}$;
- $I(f)$ the set of all instances of the flow f.

Definition 4.10 (Workload). Let P be a platform with $l \in P.Links$ and $f \in Flows$. A link l in the flow tree $ft(f)$ is occupied by f

during a certain amount of time. It is the workload wl defined as :

$$wl(f,l) = \begin{cases} f.\tau.C & \text{if } l.type = CPU \\ \frac{f.m.len}{l.q} & \text{if } l.type \in \{DMA, Net\} \end{cases}$$

If the link is a *CPU* link, the task corresponding to flow f is executed, so the workload is given by its WCET. In case of network and *DMA* links, the workload is the time required by f to be sent on l. It depends on the $f.m.len$ and $l.q$[1].

The evolution of flows instances through the system is represented by two types of events: the arrival and the transmission of flows instances on the links.

Definition 4.11 (Arrival time). Let $t_{arr} : I(Flows) \rightarrow \mathbb{R}^+$, $t_{arr}(f_k^l)$ the time when a flow instance f_k arrives on link l.

Definition 4.12 (Transmission time). Let $t_{trans} : I(Flows) \rightarrow \mathbb{R}^+$, $t_{trans}(f_k^l)$ the time when a flow instance f_k is transmitted or executed on link l.

4.3 Functional analysis

Definition 4.13 (Data). A data $\delta \in Data$ is directly related to a flow f. We denote this relation by $srcFlow : \delta \mapsto f$. A new instance of the flow f_k corresponds to a new refresh of the value of δ, denoted δ_k. This matches the principles brought in [4].

Definition 4.14 (Carry). To associate a data instance δ_j to the flow instance f_k which carries it, let us introduce a Boolean function $carry(f_k, \delta_j)$ which can be either 1 if f_k carries δ_j or 0 otherwise.

Definition 4.15 (Functional chain). A functional chain $FC \in Funcs$ is a directed acyclic graph (DAG) of flows. We note $(f, f') \in FC$ where $f, f' \in Flows$ to mean that f precedes f' in FC, indicating that the data transported by f is transmitted to f'.

Example 4.16. An example of functional chain is given in Figure 3 where f_0 is the entry flow of the chain (*i.e.*, it receives input data δ_0) and f_3 is the output flow (*i.e.*, the end of the functional chain).

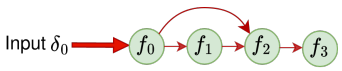


Figure 3: Example of functional chain

Definition 4.17 (System). A system $Sys :=< P, Funcs >$ is a platform P hosting a set of functional chains *Funcs*.

5 OPTIMIZATION PROBLEM

The optimization problem aims to find the optimal allocation of tasks and network messages. This assignment depends on the transmission time of each flow instance in the flow tree which optimizes the functional latency. To perform this global optimization, we assume perfect synchronization. Based on these times, future work will define configurations depending on the characteristics of the distributed systems in terms of synchronization and service policies.

[1]This supposes that the execution time through the DMA is linear with the length of the flow to transmit

5.1 Parameters and variables

The optimization problem considers the following parameters and decision variables.

5.1.1 Constant parameters.

- A system Sys and the hardware platform of the system, $Sys.P$;
- The set of functional chains in the system, $Sys.Funcs$;
- The flows composing the functional chains: $Flows = \{f | f \in \bigcup Im(FC) \forall FC \in Sys.Funcs\}$
- A set of data, $Data$, associated to flows in functional chains.

5.1.2 Decision variables.

- $\forall f \in Flows, ft(f)$ are the flow-trees to be determined for all flows in the system;
- $\forall f_k^l \in I(Flows)$, variables $t_{arr}(f_k^l)$ and $t_{trans}(f_k^l)$ are the arrival and the transmission times to compute for each instance of flow on each link that will lead to an allocation.
- $\forall \delta_j | \delta \in Data \wedge \exists srcFlow(\delta)_j \in I(Flows), \forall f_k \in I(Flows)$, the variable $carry(f_k, \delta_j)$ is to be determined for each flow instance f_k, and each data instance δ_j.

5.2 Objective

Definition 5.1 (Functional latency). Functional latency is the time between the entry of a data instance δ_k, and its use by a frame of an output flow on a leaf link $f'^{l'}_{k'}$:

$$FL_{f'^{l'}_{k'},\delta_k} = carry(f'^{l'}_{k'}, \delta_k) \times [t_{trans}(f'^{l'}_{k'}) + wl(f', l') + l'.d_p - t_{arr}(f_k^l)], \text{ where } f = srcFlow(\delta)$$

The objective of the optimization problem is to minimize the maximum functional latency of all flows carrying data, in short, the age delay [2]:

$$\text{minimize} : \max_{f'^{l'}_{k'} \in I(Flows), \delta \in Data} (FL_{f'^{l'}_{k'},\delta_k})$$

5.3 Constraints

Constraint 5.3.1 (Resource use)**.** This first constraint represents the impossibility for a link or a CPU to process two messages or two tasks at the same time. This translates as "for any pair of flow instances, one must be fully processed on the link l so that the other instance can begin its process on the same link l".

$$\forall l \in Links, \forall f, f' \in Flows | l \in Im(ft(f)) \cap Im(ft(f'))$$
$$\forall k, k' \in \mathbb{N} | f \neq f' \vee k \neq k'$$
$$t_{trans}(f_k^l) + wl(f,l) \leq t_{trans}(f'^{l}_{k'})$$
$$\textbf{XOR } t_{trans}(f'^{l}_{k'}) + wl(f,l) \leq t_{trans}(f_k^l)$$

Constraint 5.3.2 (Ready before running)**.** A flow instance can not be processed by a CPU/DMA link or sent by a network link until it arrives in the queue of the link.

$$\forall f \in Flows, \forall l \in Im(ft(f)) \forall k \in \mathbb{N}, t_{arr}(f_k^l) \leq t_{trans}(f_k^l)$$

Constraint 5.3.3 (CPU allocation and routing)**.** The flow requires a flow tree of links on which flow instances are to be transmitted. The flow tree must start with the CPU link responsible for the

execution of the task associated to the flow. So, Equation 1 enables a single CPU link as the root of the flow-tree, while Equation 2 builds the tree.

$$\forall f \in Flows, \exists! l \in Im(ft(f)) | l.type = CPU \tag{1}$$

$$\begin{aligned} &\forall f \in Flows, \forall l \in Im(ft(f)) | l.type \neq CPU \\ &\exists! l' \in Im(ft(f)) | l'.dst = l.src \end{aligned} \tag{2}$$

Constraint 5.3.4 (Functional chain connectedness)**.** A functional chain is defined by pairs of flows. To ensure the connectedness of the functional chain, for each pair of flows, the instances of the first flow must be routed to the node where the second flow starts.

$$\begin{aligned} &\forall FC, \forall (f, f') \in FC \\ &\forall l' \in Im(ft(f')) | l'.type = CPU \\ &\exists l \in Im(ft(f)) | l.dst = l'.src \end{aligned}$$

Constraint 5.3.5 (Link transition)**.** The transmission of a flow instance from a link to another is the relation between the transmission time on a link l and the arrival time in the following link(s).

$$\begin{aligned} &\forall f_k^l \in I(Flows), \\ &\forall l' \in Im(ft(f)) \setminus \{l\} | l.dst = l'.src \wedge l'.type \neq CPU \\ &t_{arr}(f_k^{l'}) = t_{trans}(f_k^l) + wl(f, l) + l.d_p \end{aligned}$$

Constraint 5.3.6 (Data generation)**.** In real-time systems, data periodically introduced in the system by sensors and then collected by tasks. In our model, we consider the data to be refreshed at each new instance of its source flow:

$$\begin{aligned} &\forall \delta \in Data, f = srcFlow(\delta), \forall l \in Im(ft(f)) \\ &\forall k \in \mathbb{N}, carry(f_k, \delta_k) = 1 \end{aligned}$$

Constraint 5.3.7 (Data transmission)**.** The data generated by the source flows is forwarded to some other flows. The data transfer takes place between two flow instances $f_k^l, {f'}_{k'}^{l'}$ under the following conditions: f precedes f', l is adjacent to l', l' is the root of f' flow tree, and f_k^l is the last instance of f fully transmitted by l before the start of ${f'}_{k'}^{l'}$'s transmission.

$$\forall FC \in Sys.Funcs, \forall f' \in Im(FC) \forall {f'}_{k'}^{l'} \in I(f) | l'.type = CPU$$

$$\forall \delta_j | \delta \in Data \wedge \exists srcFlow(\delta)_j^l \in I(Flows)$$

$$carry({f'}_{k'}, \delta_j) \Leftrightarrow \exists f_k^l | \begin{cases} carry(f_k, \delta_j) \\ (f, f') \in FC \\ l.dst = l'.src \wedge l \in Im(ft(f)) \\ t_{trans}(f_k^l) + wl(f, l) + l.d_p \leq t_{trans}({f'}_{k'}^{l'}) \\ \forall f_{k''}^l : t_{trans}(f_{k''}^l) \leq t_{trans}(f_k^l) \\ \textbf{or } t_{trans}(f_{k''}^l) + wl(f, l) + l.d_p \geq t_{trans}({f'}_{k'}^{l'}) \end{cases}$$

6 PRELIMINARY EVALUATION

6.1 Complexity estimation

The complexity of a Periodic Schedule Problem is non-polynomial in most cases. In Table 1, We compute the complexity associated with each constraint in our problem formulation considering the following notations:

- $n_{Links}, n_{CPU-Links}, max(adj)$ the numbers of links, CPU-links and the maximum number of adjacent links;
- $n_f, n_{Instances}$ the number of flows and the number of all instances of all flows on all links;
- $n_{Funcs}, max(FC)$ the number of functions, and the maximum length of a function (in terms of precedence);
- $n_{Data}, n_{dataInst}$ the number of data in the system, and the number of data instances.

Constraint	Instances
Resource use	$\frac{n_{Instances}^2}{n_{Links}}$
Ready before running	$n_{Instances}$
CPU allocation and routing eq. 1	n_f
CPU allocation and routing eq. 2	$n_f \times n_{Links}$
Functional chains connectedness	$n_{Funcs} \times max(FC) \times n_{CPU-links}$
Link transition	$n_{Instances} \times max(adj)$
Data generation	$n_{dataInst} \times \frac{n_{Instances}}{n_f}$
Data transmission	$n_{Funcs} \times max(FC) \times n_{Data}$

Table 1: Constraints complexity

6.2 Example

The following example represents a small avionics case study based on the platform in Figure 2. Let f_0 is an ADIRS function (which computes the Mach number of the aircraft). f_0 runs on module M_0. It sends it to the automatic pilot (f_1, on M_2) and to the flight control function (f_2, on M_2). f_1 computes and sends the flight objective to f_2. Finally, f_2 computes and sends the angles to apply to the flight surfaces to f_3 (the flight surface control laws, on M_3).

Flow	Task			Message
	T	C	D	len
f_0	8	2	8	200
f_1	8	2	8	200
f_2	4	2	4	200
f_3	2	1	2	-

Table 2: Flows parameters

For the given example, we can count all the instances of all constraints. For one functional chain, more than 5000 constraints are generated. Such a large number of constraints may imply a large computing time. This parameter must be monitored cautiously during incoming optimization trials. If the computing time becomes too long, meta-heuristics need to be considered instead of the exact solving approach.

7 CONCLUSION AND FUTURE WORK

This paper proposes a mathematical formulation to find optimal tasks and network messages allocation for an IMA networked platform model.

As a future work, we intend first to implement the proposed mathematical formulation in an optimization program and solve the problem with an exact solver (such as Z3) or find near-optimal solutions with meta-heuristics (genetic algorithms). Second, this first formulation needs to be extended to include constraints related to specific IMA and network (for instance TSN) service policies which will allow to handle both scheduled and unscheduled traffic.

REFERENCES

[1] Aeronautical Radio Inc. ARINC Specification 653 P1-3. 2013. Avionics Application Software Standard Interface: Required Services.

[2] Mohammad Ashjaei, Nima Khalilzad, Saad Mubeen, Moris Behnam, Ingo Sander, Luís Almeida, and Thomas Nolte. 2017. Designing end-to-end resource reservations in predictable distributed embedded systems. *Real-Time Systems* 53 (11 2017), 1–41. https://doi.org/10.1007/s11241-017-9283-6

[3] Nesrine Badache, Katia Jaffres-Runser, Jean-Luc Scharbarg, and Christian Fraboul. 2014. Managing temporal allocation in integrated modular avionics. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–8.

[4] Matthias Becker, Dakshina Dasari, Saad Mubeen, Moris Behnam, and Thomas Nolte. 2017. End-to-end timing analysis of cause-effect chains in automotive embedded systems. *Journal of Systems Architecture* 80 (2017), 104–113.

[5] Silviu S. Craciunas and Ramon Serna Oliver. 2016. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Systems* 52, 2 (01 Mar 2016), 161–200. https://doi.org/10.1007/s11241-015-9244-x

[6] Simon Fürst, Jürgen Mössinger, Stefan Bunzel, Thomas Weber, Frank Kirschke-Biller, Peter Heitkämper, Gerulf Kinkelin, Kenji Nishikawa, and Klaus Lange. 2009. AUTOSAR–A Worldwide Standard is on the Road. In *14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden*. Citeseer.

[7] IEEE. 2018. 802.1Q - IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks.

[8] Michael Lauer, Jérôme Ermont, Frédéric Boniol, and Claire Pagetti. 2011. Latency and freshness analysis on IMA systems. In *ETFA2011*. IEEE, 1–8.

[9] Shane D McLean, Emil Alexander Juul Hansen, Paul Pop, and Silviu S Craciunas. 2022. Configuring ADAS platforms for automotive applications using metaheuristics. *Frontiers in Robotics and AI* (2022), 353.

[10] Anna Minaeva and Zdeněk Hanzálek. 2021. Survey on periodic scheduling for time-triggered hard real-time systems. *ACM Computing Surveys (CSUR)* 54, 1 (2021), 1–32.

[11] Wilfried Steiner. 2010. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 375–384.