A model-based approach for analysing network communication timeliness in IMA systems at concept level

Rodrigo Saar de Moraes
rodrigo.moraes@liu.se
Dept. of Computer and Information Science
Linköping University
Sweden

Simona Bernardi
simonab@unizar.es
Dept. of Computer Science and Systems Engineering
Universidad de Zaragoza
Spain

Simin Nadjm-Tehrani
simin.nadjm-tehrani@liu.se
Dept. of Computer and Information Science
Linköping University
Sweden

ABSTRACT

Analyzing the resource adequacy of complex cyber-physical systems at concept development stage can be a challenging task since there are a lot of uncertainties about the system at this stage. In Integrated Modular Avionics (IMA) systems, with a life-cycle over several decades and potential functionality changes, we need to estimate resource needs at the early stage but leave capacity to absorb future modifications. Given an envisaged set of functions and a mapping to a candidate platform, one needs to assure that the selected network configuration will provide adequate resources to meet communication timeliness. In particular, whether the set of switches, the topology, and the available bandwidth are sufficient to meet the envisaged needs. In this paper, timeliness requirements are expressed as constraints on the freshness of data and a strict bounding of end-to-end latency. We support generation of UML/MARTE-based specifications by creating a domain-specific meta-model for IMA systems and a resource modelling approach for the study of time-critical systems. The instances of this model then specify the application requirements and various network configurations that can be formally analyzed. We present a tool, M2NC, for automatic derivation of a network calculus model through model transformation, and use the state-of-art NC tools for deriving the bounds for end-to-end timeliness. The approach is illustrated on an example avionics case study, consisting of 91 computational processes that exchange 629 different types of messages. The results of the analysis show that our approach can efficiently provide feedback on configurations that are compliant with the requirements imposed by the application and the toolchain provides a systematic mechanism to quickly identify potential future bottlenecks.

CCS CONCEPTS

• **Networks** → *Network performance modeling*; • **Computer systems organization** → **Real-time system architecture**; • **Computing methodologies** → **Model verification and validation**; Modeling methodologies.

RTNS'2021, April 7–9, 2021, NANTES, France

ACM ISBN 978-1-4503-9001-9/21/04…$15.00
https://doi.org/10.1145/3453417.3453427

KEYWORDS

Real-Time Systems, Concept Analysis, Network Resource Adequacy, UML-MARTE, Model Verification

ACM Reference Format:
Rodrigo Saar de Moraes, Simona Bernardi, and Simin Nadjm-Tehrani. 2021. A model-based approach for analysing network communication timeliness in IMA systems at concept level. In *29th International Conference on Real-Time Networks and Systems (RTNS'2021), April 7–9, 2021, NANTES, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3453417.3453427

1 INTRODUCTION

Modelling complex cyber-physical systems (CPS) [11] for analysis of resource adequacy can be a challenging task. In the concept phase, there are a lot of uncertainties concerning the partial specification of the software functions and the hardware platform the software system will run on. Notwithstanding such uncertainties, hardware components must be decided and the platform as a whole needs to be analyzed to ensure that processing and network resources are adequate to fulfill the timeliness requirement of the system. This requires efficient methods that help to create a clear idea about the adequacy of platform resources in the initial concept level architecture. Integrated Modular Avionics (IMA) Systems [21], which are networked systems whose life-cycle spans over several decades, require a flexible and efficient means to consider or plan for future functionality extensions.

In this work, we address the problem of analyzing the resource adequacy in networked systems under timeliness constraints, to provide this support in the early concept phase of the IMA systems life-cycle. In particular, to meet the timeliness constraints, the communication delay between processing nodes of such systems must be strictly bounded. The input to this analysis is a set of computational processes mapped to computing elements. The interaction patterns between processes create a virtual network topology on a networking platform. The combination of the two ideally helps to estimate exact worst-case end-to-end delays. However, since IMA systems are composed of dozens of communication devices and hundreds of message flows, obtaining the exact delays may be combinatorially unfeasible. Therefore several approaches, such as network calculus (NC) [6, 7] and trajectory approach [12, 15], have been proposed that enable computing bounds for the worst-case delay in a networked system.

In this paper, we start from a UML-based representation in the form of UML-MARTE [17] deployment and communication diagrams and propose a transformation scheme from these models to a corresponding network calculus representation. The idea is to support flexible evaluation of candidate platform configurations in terms of the applications and functions that they must support, abstracting away complex network and computational component models. For each model, we aim to verify whether the network resources of a candidate platform configuration are sufficient to support a set of applications that constitute potential avionics functions in an airborne system.

With modularity in mind, the notation we propose suits the evaluation of alternative platform resource levels and reusability in future configurations. Although we illustrate our approach on an IMA concept model, the proposed approach is not restricted to these types of systems, but any system model that exhibits periodic computation and communication patterns. In fact, we believe that our approach is general enough so that complex CPS system architects can find it beneficial for creating their models.

The main contributions of this paper are as follows:

- A Meta-model for IMA domain-specific modelling and a UML-MARTE resource modelling approach for the study of time-critical networked systems at a concept stage
- A workflow and supporting tool (CLASSICS-M2NC) for translation of UML models into network calculus models, bridging the gap between the modelling and the analysis of complex systems with real-world dimensions.
- Definition of a method for extraction of message freshness requirements, and application of the proposed workflow to an avionics use-case to demonstrate the scalability of the approach.

The remainder of this paper is organized as follows: Section 2 presents the related works and Section 3 introduces the problem statement. Section 4 presents the background in UML-MARTE and network calculus. Readers familiar with those can skip the section. In Section 5 the modelling and analysis workflow is presented. Section 6 presents the application of the workflow to an industrial use-case. Finally, conclusions are drawn in Section 7.

2 RELATED WORKS

Wang and Niu [20] discuss the characteristics of Distributed Integrated Modular Avionics Systems (DIMA) and highlight three key techniques that can help in the development process: mixed critical task scheduling, real-time fault-tolerant scheduling, and real-time communication network delay analysis. The first two are concerned with how to schedule tasks to meet timeliness and dependability. The latter aims at ensuring the real-time performance of the distributed system. Similarly, Badache et al. [1] focus on the importance of a careful and thorough study of the temporal model of the system at the early development stage. In particular, they consider end-to-end delays, which are decomposed into 3 parts, the delay at the source node, the delay at the destination node, and the network traversing delay.

All these works identify a dual problem in the verification of complex real-time systems: mutual dependency of timely computation and timely arrival of messages. The second part of the problem is especially challenging when it comes to IMA and other complex CPS. To deal with this challenge, network calculus [6, 7] and trajectory approaches [12, 15] have been proposed to compute theoretical upper bounds for the worst-case communication delay.

Due to its versatility, network calculus (NC) has seen multiple applications in the analysis of IMA and other time-critical systems. Zhao et al. [22] applied NC to Time-Triggered Ethernet (TTEthernet) networks, obtaining bounds that are on average 23% higher than those provided by the state-of-the-art trajectory approach at the time. Such precision is, however, achieved 97% faster, giving NC an edge when rough estimates are acceptable and time-effective analyses are needed.

The application of NC to the context of IMA systems is also studied by Soni et al. [19] who perform an investigation on the pessimism introduced by NC when applied to Avionic Full-Duplex Switched Ethernet (AFDX) networks. They estimate the average upper bound of pessimism to be around 10% for an industrial-sized network implementing a FIFO message queuing policy. Note, however, that their solution does not take advantage of the latest NC techniques, such as TMA analysis [2], and their upper bound on the pessimism is based on a comparison to an optimistic approach. In this sense, we expect the delays obtained by the NC analysis in our work to be comparable to or lower than the 10% obtained by these authors.

Even though these results show that network calculus usually provides pessimistic bounds, the overestimation can be overlooked during the conceptual phase, where a trade-off between repeated fast analysis and a certain degree of overestimation may be feasible. Furthermore, network calculus provides a framework that is suitable for abstracting large-scale systems. It is important to note that the level of pessimism introduced by NC is highly dependent on both the analysis algorithm [2] and the traffic modelling technique [4]. Boyer et al. [4] studied the influence of the modelling technique on the verification time and the quality of the bounds. Their study suggests that a discrete approach to traffic modelling, as opposed to a continuous traffic model, can reduce the pessimism on the bounds by a factor of 18% (on average) at the cost of increasing the time taken for verification.

While exact analysis with model checking is possible, our experience with timed automata [8, 9] has shown that a similar level of abstraction for systems of the sizes we are considering does not scale for analysis with model checking. In fact, Charara et al. [5] establish a comparison between simulations, network calculus, and formal methods to evaluate end-to-end delay on Avionics Full-Duplex Switched Ethernet (AFDX) networks. According to their findings, network calculus is the only scalable method able to provide guaranteed upper bounds on end-to-end delays. The simulation approach provides experimental bounds that might be easily exceeded as simulation can miss rare events, and the model checking approach is prone to state-explosion issues.

Besides the analysis challenge, another challenge is the conceptual modelling of CPS for verification purposes. Indeed, the modelling language used by the system engineers is often semi-formal and the model usually consists of multiple views that represent different aspects of the system. To use model checkers or NC descriptions for answering their questions, the system engineers need to have a deep understanding of the formal models and their properties. To bridge that gap Robati et al. [18], extend the Architecture Analysis and Design Language (AADL) to model TTEthernet-based distributed systems. They define model transformations to enable the verification of the AADL models using Discrete Event System

Specification (DEVS) based simulations. The approach is applied for the verification of small-size IMA systems and the model transformation is partially automated.

In this paper, we consider UML-MARTE as modelling language. Similarly, Louati et al. [13], Ge and Pantel [10], and Mahfoudhi and Karamti [14] propose different UML-MARTE-based modelling and verification frameworks for real-time systems. In particular, they propose different translations from the UML-MARTE models to timed Petri nets, which are used to verify temporal properties defined in temporal logics. Wang et al. [20] extract information from UML-MARTE models to construct a series of theorems that describe properties of IMA systems, such as time constraints, spatial isolation, and health monitoring. While these works tackle important aspects of the verification of timeliness in real-time systems, our work is orthogonal to those since we address resource adequacy in terms of network-related properties.

To sum up, the mentioned works combine modelling and verification frameworks to provide guarantees with tighter bounds, provide exact estimations for small systems, or to analyze other identified properties. Our work, on the other hand, aims to provide the needed support so that our reusable models are populated by engineers, and fast and scalable analysis of multiple instances of future platforms can be provided efficiently with some sacrifice of exactness in the end-to-end delays.

3 COMMUNICATION MODELLING IN NETWORKED SYSTEMS

This work focuses on the development of a generic networked platform representation and translation approach to be used while considering candidate platform configurations. We aim to verify whether the resources of a candidate configuration are sufficient to support an Application Model (AM). The AM captures the resource requirements that the configuration needs to satisfy. The model can be adapted to enable the evaluation of alternative configurations (e.g., network topologies).

3.1 Modelling Networked Systems

As most computer-based systems, IMA networked systems can be modelled in terms of two perspectives: the software applications and the hardware platform on which the applications are deployed. Figure 1 shows the domain model of a generic IMA system with these two perspectives.

On the software side, an application model describes the relevant computation and communication characteristics of a set of communicating computational processes. Each application in the AM is composed of multiple periodic processes, which communicate with each other through periodic messages. From a network-centric point of view, each process is characterized by its period, which also defines the frequency at which it will send its (output) message types. In turn, each message type is characterized by its size in bytes.

A process, with rare exceptions, is both a producer and consumer of messages. In a general model, sending-receiving process pairs can produce and consume messages at the same rate or different rates. The modelling approach should be able to capture all the cases.

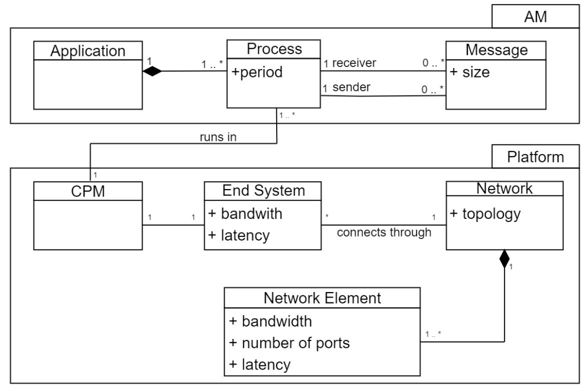

Figure 1: IMA domain model.

Platform configurations represent the different architectures that can be used to support a given AM in terms of physical resources. In this work, we are concerned with networking issues, thus a platform configuration is described in terms of process mappings to nodes and network characteristics (i.e. topology and performance metrics of network elements). In the domain model, processes are run in core processing modules (CPMs), which stand for any hardware device capable of computation, representing, among others, devices such as specific purpose circuit boards, system-on-a-chip boards, microprocessors, microcontrollers, video processors, and traditional processors. CPMs communicate through a network using network interfaces, hereby called end-systems (ESs). The network itself is described in terms of its topology (how different end-systems are connected and through which routes messages can flow). End-systems and network devices are characterized by performance metrics, such as the introduced latency and bandwidth. Also, network devices (e.g., switches, hubs, buses, routers, bridges) are characterized by their number of ports.

Figure 2 shows a high level representation of a generic IMA system with m processes and n CPMs.

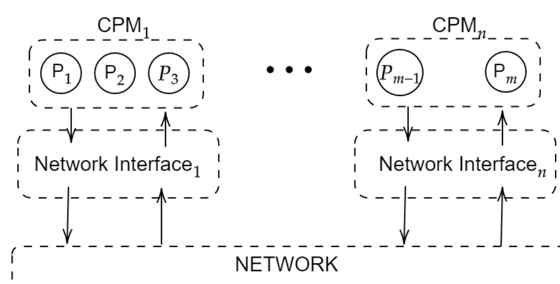

Figure 2: High level representation of an IMA system with m processes and n CPMs.

It is important to note that, since we address modeling at a conceptual stage, some characteristics of the system such as bandwidth allocations and scheduling of individual messages are typically not yet available. Thus, we propose a high-level abstraction model that can be used to verify the resource adequacy and to identify architectural issues before detailed design choices are made.

3.2 AM Communication Timeliness Requirements

Our goal is to verify whether a platform configuration is capable of guaranteeing the timeliness requirements of the application model. In this work, we assume processes have a periodic behavior. Every period, a process will start by reading the messages received from other processes. The process then executes its internal activity, and finally sends out its own set of messages. From a network-centric perspective, the time taken read the received messages and to prepare and place the outgoing messages at the outgoing communication interface are considered negligible in comparison to the time a message spends in the network. However, adding ES overhead to the model can be later accommodated with assuming a fixed bound.

In principle, we consider that processes communicate in sending-receiving pairs subject to one of 3 communication patterns: sending (producer) and receiving (consumer) process share the same period; producer is faster than consumer; or producer is slower than consumer. The third case can lead to a situation in which a receiving process has to either wait for incoming messages, or to work with old data. To avoid problems with processes waiting for data, one can resort to specifying freshness requirements. The freshness of the data contained in a message relates to how old that data can be so that it is still relevant to be consumed by a process. This means that some discrepancy between producing and consuming rate is allowed, but the network should not make the delays longer than an acceptable limit represented by freshness requirements.

In this work, we express freshness requirements in terms of the communication latency accepted by a given process, which is the maximum time it takes for all messages destined to that process to be received. Formally, it can be defined as follows:

Definition [Communication Latency]. *Let P be the set of processes of an application and M_p the set of message types to be received by process p. The communication latency of process p can be defined as:*

$$CommLatency_p = Max_{m \in M_p} delay_m \tag{1}$$

where $delay_m$ is the maximum time process p has to wait to receive any message of type $m \in M_p$ after it was sent.

In other words, such delay corresponds to the maximum time any message of type m spends in the network before reaching its destination. Given our previous assumptions, the communication timeliness of a process can be formulated as follows:

Definition [Timeliness Requirement]. *Let $period_p$ be the period of process p. The timeliness of communication for each process can be defined by:*

$$\forall p \in P : CommLatency_p \leq period_p. \tag{2}$$

In practice, this means that the delay experienced by a message on its way to the destination should not be bigger than the period of the receiving process. This condition ensures that, if new data exists, a process will never work with old data for more than one execution cycle.

4 BASIC CONCEPTS

In the following, we provide the background on UML-MARTE (sub-Section 4.1) and network calculus (sub-Section 4.2).

4.1 UML - MARTE

Unified Modelling Language (UML) [16] is the de-facto industry standard for software modelling. It enables specifying the structure and the behavior of a software system employing different views.

UML also allows defining domain-specific extensions utilizing *profiling*. The basic profiling mechanism is the stereotype that extends one or more UML meta-classes with new properties (i.e., attributes). Thus, when considering a UML model of an application, the stereotype can be applied to the model elements that are instances of the meta-classes extended by the stereotype.

The UML profile MARTE (Modelling and Analysis of Real-Time and Embedded Systems) [17] is an Object Management Group (OMG) standard that defines domain-specific concepts related to the architecture and the performance of real-time systems (RTS). In particular, MARTE enables expressing non-functional properties (e.g., end-to-end delays), resource allocation, scheduling primitives, etc., then providing support to the engineer for the specification and analysis of the schedulability and performance of RTS.

In this work, we consider two types of UML diagrams for the specification of a system: deployment and communication diagrams. Concretely, deployment diagrams are used to represent the platform, (i.e., the physical allocation of processes to CPMs, the end-systems, and the communication network), whereas the communication diagrams are used to model the exchange of messages.

4.1.1 Deployment Diagrams. A UML deployment diagram is a structural diagram used to model the architecture of a software system in terms of the mapping of software components to deployment targets. Software components are stereotyped as ≪*artifact*≫ since they are product of a software development process (like $P_0, \ldots, P_3$ in Figure 5). Deployment targets can be nodes, devices, or execution environments. Nodes represent processing resources. Devices and execution environments, in turn, are specializations of a node. In particular, devices represent hardware with physical computational capabilities for software execution, whereas execution environments represent software containers that implement services required by artifacts at execution time. An association between two different deployment targets represents a communication path through which they exchange signals and messages.

4.1.2 Communication Diagrams. A UML Communication diagram is a behavioral diagram used to represent interactions between objects. The basic elements of a communication diagram are lifelines and messages. Lifelines represent the objects or individual participants in the interaction (like $P_0, \ldots, P_3$ in Figure 4). Messages flows, in turn, are depicted as labeled arrows between lifelines. These arrows indicate the content or type of the messages and the direction of the communication.

4.1.3 MARTE Stereotypes. The MARTE profile [17] addresses two concerns: modelling the features of real-time and embedded systems (RTES), and the specification of non-functional properties for schedulability and performance. Our approach uses a subset of stereotypes of MARTE that are defined in two different packages, that is *HwCommunication* and *Schedulability Analysis Modelling* packages. The *HwCommunication* package addresses the modelling of the communication in RTES. In particular, we will use the ≪*HwMedia*≫ stereotype to specify generic communication resources with the properties described in Table 1.

Table 1: Attributes of the ≪*HwMedia*≫ stereotype

Attribute	Description
packetSize	size in bits of the message packets that can be transmitted in the communication media
bandwidth:	capacity of the communication element
blockT	time the communication media is blocked and cannot transmit
transmMode	transmission mode: simplex, half-duplex or full-duplex
resMult	multiplicity of the modelled element

We use two stereotypes from the *Schedulability Analysis Modelling* package, that is ≪*SaStep*≫ and ≪*SaCommStep*≫. The SaStep stereotype represents a software execution step in the analysis, whereas the SaCommStep stereotype represents the usage of a communication medium to send a message. Table 2 presents the attributes of the two stereotypes that are relevant in the context of this work.

Table 2: Relevant attributes of the ≪*SaStep*≫ and ≪*SaCommStep*≫ stereotypes

Stereotype	Attribute	Description
SaStep	interOccT	inter-occurrence time interval between successive executions
SaCommStep	msgSize	size of the message to be sent

4.2 Network Calculus

Network calculus [6, 7] is a theory of deterministic queuing systems useful for analysing delays and buffer backlogs in computer networks. More specifically, it provides a mathematical framework to compute deterministic bounds on delays, backlogs, and other quality-of-service parameters of a network.

In order to compute worst-case performance bounds for flows in a network, network calculus requires flows and the network to be modelled in terms of functions. A data flow is usually represented by a cumulative function that represents the amount of data observed in the flow and the flow behavior during a given time interval, namely the arrival curve $\alpha(t)$. Similarly, the handling of data by a node or sub-network is modelled by a service curve $\beta(t)$. A service curve quantifies the service provided by a given node of the network, which in network calculus is called a server, regulating the rate at which data are forwarded by this node and the delay introduced by the node in the path of the data flow. Network calculus enables deriving the arrival curve α' of the output flow by convolution of the arrival and service curves, using a (min,+) algebra (see [6, 7]).

Using algebra, the arrival and service functions can be combined and propagated through the path of a given data flow. This technique allows for the computation of local performance bounds for a data flow that interacts with other flows sharing the same nodes on its path.

The propagation of the results through the network, however, is not a straightforward process. Multiple data flows can share subpaths and compete for the services provided by the nodes that compose these subpaths. The existence of common subpaths can lead to overly-pessimistic performance bounds. As an example, if a group of flows shares a common path composed of more than one node, the delay introduced when multiplexing these flows on a server should be considered just once along this path.

Over the years, two different categories of algorithms have emerged to deal with the problems introduced by the propagation and obtain tighter (and less pessimistic) bounds: algebraic algorithms and optimization-based algorithms. Algebraic algorithms rely on heuristics to aggregate sequences of servers into tandems and combine them to obtain a singular representation for each tandem before propagating the results to the next tandem. Optimization-based algorithms transform the entire network into a set of linear programs to be solved to find the optimal combination of servers into tandems before propagating the results.

Unfortunately, most algebraic analysis methods are not able to derive tight bounds. Optimization-based methods, instead, can derive very tight bounds, but the number of linear programs to solve grows exponentially with the network size.

To the best of our knowledge, the Tandem Match Analysis algorithm (TMA) [2] is one of the techniques that provides the least pessimistic bounds. TMA is an algebraic method that uses a set of heuristics derived from the observation of patterns emerging in the optimization-based methods. TMA can compute results comparable (less than 2% of relative error) to those of the optimization-based methods while being several orders of magnitude faster. In this work, we use an implementation of the TMA algorithm provided with the DiscoDNC analysis tool [3].

5 MODELLING AND ANALYSIS APPROACH

Our approach to model and analyze a platform configuration in presence of a given application model (AM) can be represented as a workflow, shown in Figure 3, consisting of 4 actionable steps. The first step is the modelling of the application and the platform configuration with UML and MARTE. The second step is carried out using our tool, which we call CLASSICS-M2NC[1], that translates the UML-MARTE model to a network calculus model. The third step is carried out using a network calculus analysis library, that analyses the network model and calculates the end-to-end delay bound for each message flow (in this paper, DiscoDNC). Finally, in the last step, based on the results produced by the network calculus analysis tool, our tool verifies whether the platform configuration under analysis meets the network timeliness requirements extracted from the AM. The tool presents the results of the verification to the developer, who can then make an informed decision about the suitability of the platform and, possibly, refine or change the platform, e.g. deploy a

[1]The acronym CLASSICS stands for Conceptual Analysis of Safety- and Security-Critical Systems and is the acronym for our current project.

different number or type of switches, or modify the allocation of processes to CPMs, and start a new verification round.

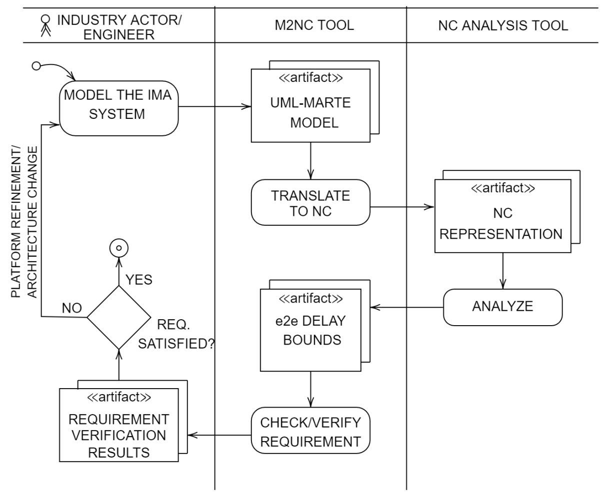

Figure 3: The proposed workflow: the rounded rectangles represent the workflow steps, whereas the rectangles stereotyped ≪*artifact*≫ are the results produced by the steps.

In the following, we describe the first two steps of our approach: the modelling of the architecture and application with UML-MARTE (sub-Section 5.1); the translation of the UML-MARTE model to a network calculus model (sub-Section 5.2).

5.1 Modelling of the IMA system

The developer starts the workflow by carrying out the first step, which is the modelling of the architecture configuration with UML-MARTE. Let us consider, as a running example, a simple AM (with four processes and seven types of messages) and an arbitrary platform configuration (with 2 CPMs, 2 ESs, and a Switch). The model consists of two views:

- A behavioral view, concretely a UML communication diagram, that represents the application communication model, and
- A structural view, a UML Deployment diagram, that represents the platform configuration.

Application Model. The AM consists of a set of processes that communicate by exchanging messages. Figure 4 shows the communication diagram of the running example, where each lifeline represents a process. The messages exchanged between processes, with the direction of the communication, are also shown in the figure. Since the message transmission time, and thus the communication delays between processes, depends on the size of the message, this parameter needs to be specified. The specification can be carried out by stereotyping the message with ≪*SaCommStep*≫, from the MARTE profile (see Table 2), and by setting the *msgSize* attribute to a value (e.g., in Figure 4, the annotation to message *m4*). Given that AMs can have substantial sizes when it comes to the number of messages and processes, the communication diagram can be divided into parts to facilitate the visualization and modelling process. In that case, the same process can be associated with multiple lifelines in different communication diagrams.

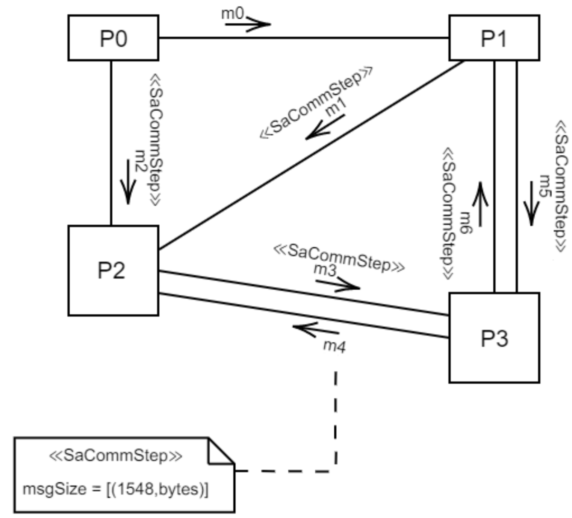

Figure 4: UML-MARTE communication diagram of an AM composed of 4 processes and 7 messages.

Platform Configuration Model. The platform configuration encompasses: 1) the allocation of the processes to the CPMs, and 2) the characterization of the end-systems, the network topology, and the network elements.

Figure 5 shows the deployment diagram that models the platform configuration of the running example. In this structural view, the processes of the AM are modeled by the ≪*artifact*≫ components: each component represents a process in the communication model (see Figure 4). The CPMs are modeled as ≪*device*≫ nodes and, thus, the modelling processes allocation on CPMs is straightforward: an artifact inside a node represents the allocation of a process to a CPM. The processes are stereotyped ≪*SaStep*≫, from the MARTE profile (see Table 2), to specify their period (*interOccT* attribute).

Each end system is represented as a package containing two UML nodes, stereotyped as ≪*HwMedia*≫, that model the sending and receiving parts of the end system (SES and RES, respectively). From the ≪*HwMedia*≫ the *bandwith* attribute is used to represent the bandwidth supported by that interface, and the *blockT* attribute is used to represent the delay introduced on the communication by the internal operations (route processing, cryptography, framing, etc) performed by the interface (see Table 1). The choice of using two different elements to represent the sending and receiving parts of the end-system is based on the assumption that different operations occur when sending or receiving messages, which implies that these operations can introduce very different delays.

The network is also represented as a package that contains different network elements (switches, hubs, buses, etc.) communicating with each other and with the ESs. In general, each of these network components is represented as a node stereotyped as ≪*HwMedia*≫.

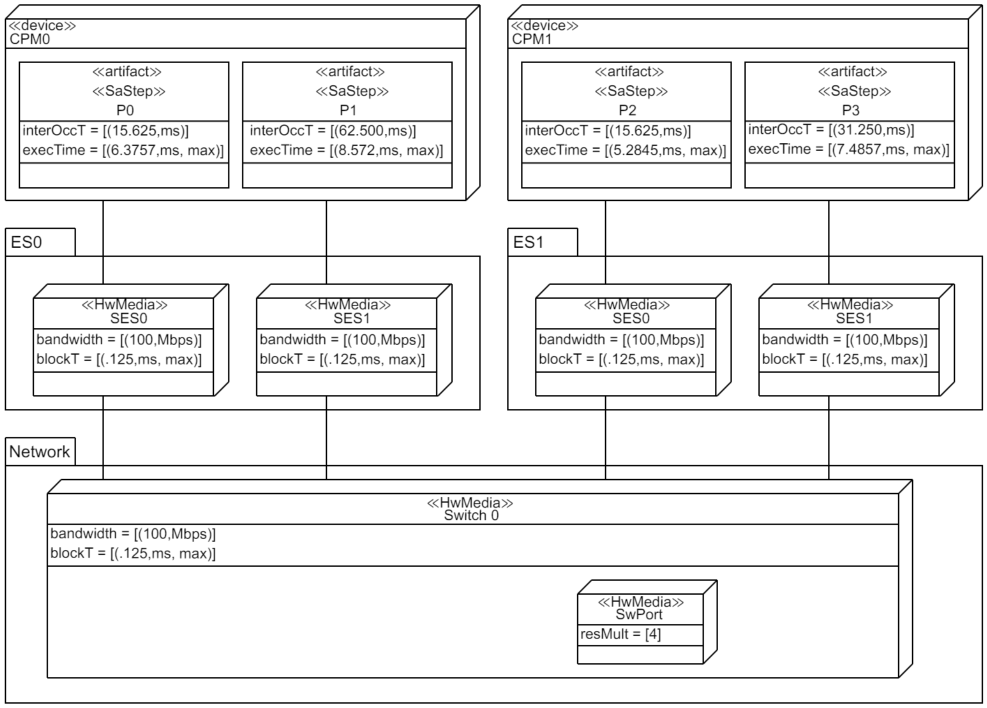

Figure 5: UML-MARTE platform configuration deployment diagram.

As in the end-system case, the *bandwith* and *blockT* attributes of ≪*HwMedia*≫ are used to characterize the bandwidth supported by the network element and the delay it introduces in the communication. In addition, the number of ports of each network element can be represented by deploying another node inside the network element node to represent the ports, and using the *resMult* attribute to specify their multiplicity. The same idea is valid for representing the internal structures of the network elements if needed.

5.2 CLASSICS-M2NC

The translation process from the UML-MARTE representation to the NC domain-specific model consists of building two different sub-models, the network model and the data flow model. The first describes the topology and characteristics of the network in terms of the network calculus server model. The second, in turn, translates the messages sent by the processes into flows of data, modelling their data rates, directions, sinks, and sources.

In particular, the CLASSICS-M2NC tool [2] receives the input model and builds the sub-models in 3 steps. First, the tool parses the input file and extracts information from the elements of the model (processes, messages, CPMs, ESs...). On a second pass, once the elements that compose the model are known, the parser extracts information about the topology of the network configuration, identifying how the hardware elements are connected and to which CPM each process is allocated. During the third and final step, our tool uses the information collected in the two previous steps to build the network model and the data flow model. The process of building these models from the data extracted from the model is described in the remainder of this subsection.

5.2.1 Building the Network Model. The network models are extracted from the UML deployment diagram by translating the ≪*hwMedia*≫ nodes into graphs of one or more servers and their respective connections to each other.

In the case of an end-system, two servers are created, one for the sending and one for the receiving part of the ES. Each of these servers connects to the respective ingoing or outgoing servers on the edge of the network and receives/forwards data flows from/to the CPM it is associated with. The service curve β of each server is constructed as a rate-latency curve using the *blockT* and *bandwidth* parameters of the ≪*hwMedia*≫ stereotyped node, SES or RES. Figure 6 shows the network model of an end-system. In the figure, the circles represent servers, the shaded ones depict the end-system and the dashed ones illustrate the servers on the edge of the network to which the ES connects to. The figure also represents the interaction with the CPM. The service curve of a server and its associated attributes is shown in Figure 7.

The structure of the switches and other network elements are constructed by creating two servers, one representing the incoming channel and the other the outgoing channel, per port. The servers representing the ports are then connected according to the internal structure of the element as defined in the UML deployment diagram. The service curves for the input and output ports of network elements are also constructed based on the *blockT* and *bandwidth* parameters of the ≪*hwMedia*≫ stereotyped nodes based on the expected behavior of the element in question. In the case of fully-connected switches, for example, every ingoing port connects to

[2] The tool and the use-case are available at https://gitlab.liu.se/ida-rtslab/2020-classics-m2nc

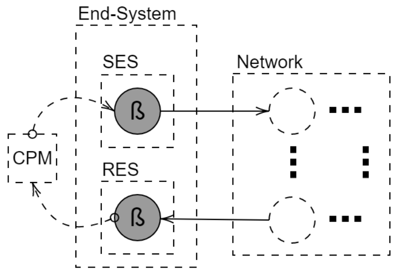

Figure 6: Network model of an end-system.

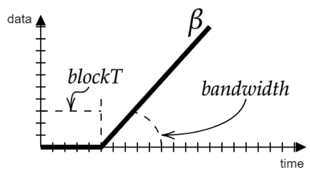

Figure 7: Service curve of a server and its associated attributes from the ≪*hwMedia*≫ stereotyped element.

an outgoing port and all the servers have the same characteristics. Note that in the case of network elements, the value of the *blockT* attribute represents the delay introduced by the whole device and has to be distributed among all the servers a flow has to go through to cross the device. The network model of a 4-port fully-connected switch is shown in Figure 8. The figure also depicts how switches connect to other switches and ESs through their SES and RES parts.

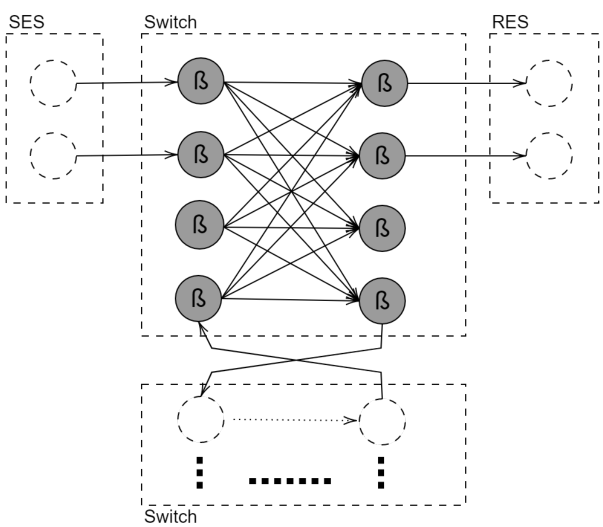

Figure 8: Network model of a fully-connected 4-port switch connecting to 2 ESs and another switch.

5.2.2 Building the Flow Model. Once the network model is built, the flow model is constructed from the UML model by extracting information about the source and destination servers associated with each ES and, consequently, to each message sent or received by the processes allocated to the CPM associated with that ES. Once the servers are identified, the arrival curve of the flow is created as a token-bucket curve based on the *msgSize* and *interOccT* parameters of the ≪*SaCommStep*≫ and ≪*SaStep*≫ stereotypes applied to the messages and processes, respectively. Figures 9 and 10 depict, respectively, a data flow going from a process allocated on CPM_i to a process allocated on CPM_j and the arrival curve of a message data flow and its associated UML-MARTE attributes.

To avoid unnecessary complexity at this stage of the translation, messages that are exchanged between two processes inside the same CPM are not modeled as flows. Instead, we assume the communication inside the CPM is done through shared memory. Thus, these messages do not contribute to the delay bounds observed in the network and can be overlooked during the verification of resource adequacy from a network perspective.

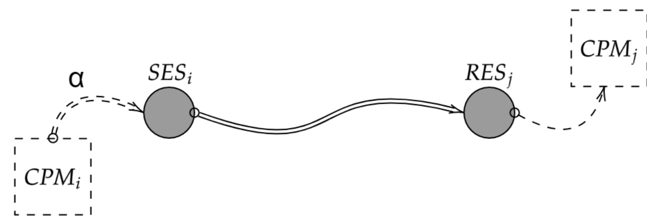

Figure 9: Flow model of a message going from a process allocated on CPM_i to a process allocated on CPM_j.

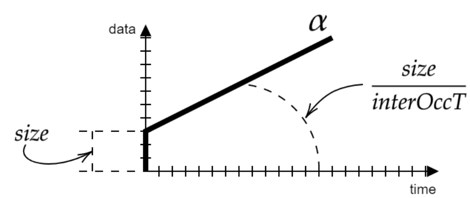

Figure 10: Arrival curve of a message data flow and its associated attributes.

6 INDUSTRIAL USE-CASE APPLICATION

As part of the assessment of the proposed workflow, the representation and translation schemes were applied to a synthetic example use-case provided by the industrial partner Saab. The use-case consists of an avionics AM constructed to mimic the properties of real concepts studied in the company and reflects the type of analysis and information that is of interest to the partner during the early concept stages. It is divided into two independent parts, a mission-oriented part (MOP) and a flight-critical part (FCP). The mission-oriented part contains 8 processes and 31 message types, and the FCP consists of 91 processes that exchange 629 messages types.

Figure 11 depicts a possible platform configuration for the MOP part, with four CPMs, four ESs, and two 4-port switches. Figure 12 shows the deployment diagram that represents this configuration (a larger version of this diagram is left out due to space issues). Tables 3 and 4 show the characteristics of the processes and message types respectively. The bandwidth requirement of each message type (in bits per second) can be derived from Tables 3 and 4 by multiplying the size of the message type (*msgSize*) by the frequency of the source process (i.e., 1 / *interOccT*). Note that, in Table 4, the size of the message type is given in bytes so the additional step of multiplying this number by 8 is necessary. The bandwidth requirements per message type, which range from 5.580Kbps to 5630.145Kbps, are omitted from Table 4 due to space restrictions. In this example, the network bandwidth and the delay introduced by the network interfaces (both in ESs and switches) have been set to 100Mbps and 125μs, respectively. Due to readability and diagram size issues, the communication diagram was left out of the paper (it can be built from the info in Table 4).

Table 3: Characteristics of the processes.

Process Identifier	interOccT	Process Identifier	interOccT
0	16.666ms	4	133.333ms
1	33.333ms	5	1066.666ms
2	16.666ms	6	66.666ms
3	66.666ms	7	33.333ms

Table 4: Characteristics of the message types (size in bytes).

Src	Dst	msgSize	Src	Dst	msgSize
0	1	4956	1	0	3608
2	3	1449	3	2	8519
6	7	1519	7	4	145
5	7	10585	4	7	550
7	6	1196	5	6	7032
4	6	10245	7	5	9631
5	4	11468	4	5	1044
6	5	3783	6	4	8758
0	3	11729	0	7	258
7	3	7384	5	1	775
7	2	11416	6	3	10553
6	2	10624	5	3	9423
5	2	744	4	3	11910
4	2	6739	2	0	6275
2	1	824	3	0	2482
3	1	1558			

The network calculus network model extracted from the deployment diagram of the MOP can be seen in Figure 13. The resulting network model is composed of 24 servers and 42 links between them.

For readability, the models and tables representing the much larger FCP part of the use-case are left out of the paper due to their size and complexity.

6.1 Comparing Platform Configurations

In order to assess the usability of the proposed workflow to compare different platform configurations, the large FCP part of the use-case has been analysed under 6 different platform configurations. The platform configurations were divided into two different topologies with 11 CPMs communicating over 3 or 4 switches, each of them with 3 different bandwidth parameters[3]. A summary of the configurations and the respective verification results can be found in Table 5, and a graphical representation of the network model for a 3 switch configuration can be seen in Figure 14.

The results in Table 5 show that for both topologies only the platform configurations with 1000Mbps interfaces (configurations 3 and 6) comply with the application requirements. Some slower networks, namely configurations 2 and 5, are not able to comply with the application timeliness requirements and are not guaranteed to deliver messages on time. In other cases, namely configurations 1 and 4 the network calculus tool is able to verify already in the very first steps of the analysis that, in those configurations, some message flows will never reach their destination since there is not enough bandwidth on the network to support the traffic, causing infinite delays. In such cases, the analysis tool signals that some server in our NC model lacks resources to service the incoming message flows, and the verification is terminated manually. The verification time for these configurations is presented as approximately zero in the table.

At this stage, it is not possible to assess if the non-compliance of configurations 2 and 5 is due to the pessimism introduced by network calculus, or whether it is inherent to the platform configuration and application characteristics. However, it should be noted that the network calculus analysis algorithm used in the verification (TMA algorithm [2]) significantly outperforms the algorithms used by Soni et al. [19], in terms of tightness of bounds, with an estimated 10% pessimism overhead. Hence, we expect the pessimism overhead introduced by NC in our analysis to be restricted to a single digit, in which case such a configuration should be avoided since there is a big chance it will not support functionality changes during the system's life-cycle.

7 CONCLUSION

In this work, we have presented a model-based workflow and tool chain for the verification of networking resource adequacy in IMA-oriented networked platforms. In order to validate the proposed workflow, we applied the proposed techniques to an example case study from an avionics context. The workflow and its application to the use case has been presented and discussed with the industrial partners, who saw the use of UML-based and analysis workflow as a useful complement to their current practices. Through this evaluation, we show that the proposed approach is useful to support the development team during the early phases of the conceptual development of systems with novel architectures.

The proposed UML-based representation of IMA systems provides a vehicle for engineers to communicate and document the early conceptual decisions in an organized manner. Furthermore, the possibility to separate the representations of the platform and the application-induced communication pattern in several independent elements was found to help the development teams with an

[3]Note that these are examples of platform configurations to illustrate both success and failure cases.

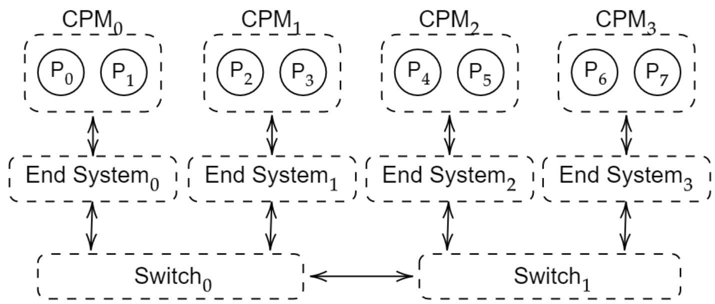

Figure 11: High level diagram representation of the mission-oriented part of the use-case on a platform with 4 CPMs and 2 switches.

Table 5: Results of the comparative verification for the FCP part.

Platform Configuration	CPMs	Switches	Bandwidth	Verification Time	Verification Result
1	11	3	10Mbps	$\approx 0s$	non-compliant
2	11	3	100Mbps	135.18s	non-compliant
3	11	3	1000Mbps	140.65s	compliant
4	11	4	10Mbps	$\approx 0s$	non-compliant
5	11	4	100Mbps	135.14s	non-compliant
6	11	4	1000Mbps	629.35s	compliant

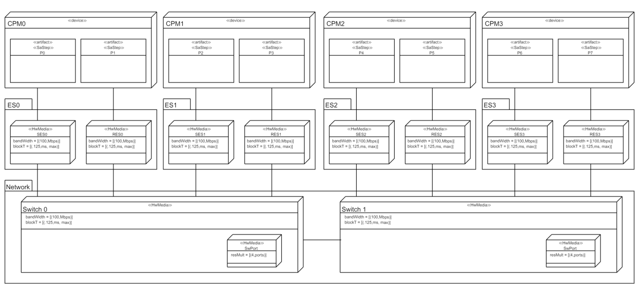

Figure 12: Deployment diagram for the mission-oriented part of the use-case on a platform configuration with 4 CPMs and 2 switches.

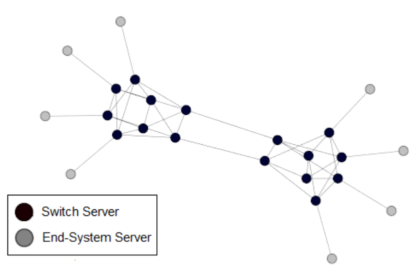

Figure 13: Server graph model extracted from the MOP deployment diagram.

easy way to visualize their models, avoiding unnecessarily big and complex diagrams.

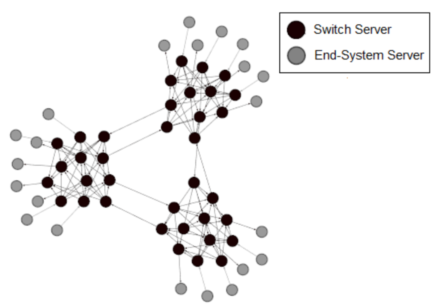

Figure 14: Network model extracted from the FCP deployment diagram, 3 switch configuration.

Even though it is known that techniques such as network calculus usually provide pessimistic bounds, the over-estimation can be overlooked during the conceptual development phase, in which the interest is to obtain a rough estimate of the platform resource adequacy. If needed, the model can, in a second round of analysis be further refined and exact methods, such as timed-automata, can be used to investigate specific subnetworks or specific nodes of interest in the network (those close to being a bottleneck) during the design phase.

Ongoing work on the presented framework includes the extension of the modelling and the translation scheme to verify and analyse other non-functional properties of the IMA systems at the same concept stage. Examples of such properties are safety, security, and energy efficiency.

ACKNOWLEDGMENTS

This work was supported by the Swedish Governmental Agency for Innovation Systems - Vinnova, as part of the national projects on aeronautics, NFFP7, project CLASSICS (NFFP7 2017-04890). Simona Bernardi was partially supported by the project Medrese (RTI2018-098543-B-I00) by the Spanish Ministry of Science, Innovation and Universities. The authors wish to thank Ingemar Söderquist and Mats Ekman from Saab AB for the valuable characterisation of IMA systems and the use case provided.

REFERENCES

[1] Nesrine Badache, Katia Jaffrès-Runser, Jean-Luc Scharbarg, and Christian Fraboul. 2013. End-to-end delay analysis in an Integrated Modular Avionics architecture. In *Int. Conf. Emerging Technologies and Factory Automation (ETFA)*. 1–4.

[2] Steffen Bondorf, Paul Nikolaus, and Jens B. Schmitt. 2017. Quality and Cost of Deterministic Network Calculus: Design and Evaluation of an Accurate and Fast Analysis. In *Proc. ACM Measurement and Analysis of Computing Systems*. 1–34. https://doi.org/10.1145/3084453

[3] S Bondorf and J. B. Schmitt. 2014. The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus. In *Proc. Int. Conf. Performance Evaluation Methodologies and Tools*. 44–49. https://doi.org/10.4108/icst.Valuetools.2014.258167

[4] Marc Boyer, Nicolas Navet, and Marc Fumey. 2012. Experimental assessment of timing verification techniques for AFDX. In *6th European Congress on Embedded Real Time Software and Systems*. Toulouse, France.

[5] H. Charara, J. . Scharbarg, J. Ermont, and C. Fraboul. 2006. Methods for bounding end-to-end delays on an AFDX network. In *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*. 10 pp.–202.

[6] R. L. Cruz. 1991. A calculus for network delay. I. Network elements in isolation. *IEEE Trans. Information Theory* 37, 1 (1991), 114–131.

[7] R. L. Cruz. 1991. A calculus for network delay. II. Network analysis. *IEEE Trans. Information Theory* 37, 1 (1991), 132–141.

[8] Rodrigo Saar de Moraes and Simin Nadjm-Tehrani. 2019. Verifying Resource Adequacy of Networked IMA Systems at Concept Level. In *Formal Techniques for Safety-Critical Systems International Workshop, FTSCS*. 40–56. https://doi.org/10.1007/978-3-030-46902-3_3

[9] Rodrigo Saar de Moraes and Simin Nadjm-Tehrani. 2021. Abstraction Models for Verifying Resource Adequacy of IMA Systems at Concept Level. *Science of Computer Programming* (2021), to appear.

[10] Ning Ge and Marc Pantel. 2012. Time properties dedicated semantics for UML/MARTE safety critical real-time system verification. In *Proc. European Conf. Modelling Foundations and Applications*. 25–39.

[11] Siddhartha Kumar Khaitan and James D McCalley. 2014. Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems J.* 9, 2 (2014), 350–365.

[12] X. Li, O. Cros, and L. George. 2014. The Trajectory approach for AFDX FIFO networks revisited and corrected. In *Int. Conf. on Embedded and Real-Time Computing Systems and Applications*. 1–10.

[13] Aymen Louati, Kamei Barkaoui, and Chadlia Jerad. 2014. Time properties verification of UML/MARTE real-time systems. In *Proc. IEEE Int. Conf. Information Reuse and Integration*. 386–393.

[14] Adel Mahfoudhi and Walid Karamti. 2015. Transformation process of RTS scheduling analysis requirements from UML/MARTE to dynamic priority time Petri Nets. *The Journal of Supercomputing* 71, 10 (2015), 3637–3667.

[15] Steven Martin, Pascale Minet, and Laurent George. 2005. End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach. *Int. J. of Communication Systems* 18, 1 (2005), 37–56.

[16] OMG. 2017. Unified Modeling Language. Version 2.5.1, formal/17-12-05, December 2017.

[17] OMG. 2019. UML Profile for MARTE, Modeling and Analysis of Real-Time Embedded Systems V1.2.

[18] Tiyam Robati, Abdelouahed Gherbi, and John Mullins. 2016. A Modeling and Verification Approach to the Design of Distributed IMA Architectures Using TTEthernet. *Procedia Computer Science* 83 (2016), 229–236.

[19] A. Soni, X. Li, J. Scharbarg, and C. Fraboul. 2017. Work in progress paper: pessimism analysis of network calculus approach on AFDX networks. In *IEEE Int. Symp. on Industrial Embedded Systems (SIES)*. 1–4. https://doi.org/10.1109/SIES.2017.7993380

[20] Hongchun Wang and Wensheng Niu. 2018. A Review on Key Technologies of the Distributed Integrated Modular Avionics System. *Int. J. of Wireless Information Networks* 25, 3 (2018), 358–369.

[21] Christopher B Watkins. 2006. Integrated modular avionics: managing the allocation of shared intersystem resources. In *IEEE/AIAA Digital Avionics Systems Conf.* 1–12.

[22] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. 2017. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems* 53, 2 (2017), 254–287.