Yet another experience on TSN tools interoperability for critical embedded networks

Philippe Cuenot, Thierry Leydier, Damien Fruchard, Massimo Barbero, Quentin Bailleul

Yet another experience on TSN tools interoperability for critical embedded networks

Philippe Cuenot
Continental Automotive
Toulouse, France
philippe.cuenot@continental-corporation.com

Thierry Leydier
Virtualité Réelle
Saint-Orens, France
thierry.leydier@virtualitereelle.com

Damien Fruchard
Airbus Defence and Space
Toulouse, France
damien.fruchard@airbus.com

Massimo Barbero
Thales Alenia Space
Cannes, France
massimo.barbero@thalesaleniaspace.com

Quentin Bailleul
IRT Saint Exupéry
Toulouse, France
quentin.bailleul@irt-saintexupery.com

***Abstract*—The introduction of Ethernet into critical embedded applications opens new needs to master and secure network development and deployment. While Ethernet is a well known Information Technology (IT) brick and deployed in the industry on a case-by-case basis, Time-Sensitive Networking (TSN) complements have only recently emerged in the aerospace and automotive industries as a promising solution to provide real-time, reliability, and availability guaranties for safety-critical systems. The complexity and diversity of TSN mechanisms enforce the use of specialized tools to assist the network engineer for the design, configuration and deployment of the network parameters. On the other hand, IETF has proposed Yet Another Next Generation (YANG) modeling language for interoperability in configuration and monitoring of various network devices. In this paper, we propose to revisit and complement the YANG standardised model in order to enable tool interoperability, with the aim of providing these complements as open source. The benefit of the proposed YANG model will be demonstrated on a TSN industrial use case with a set of tools ranging from network design and configuration to deployment on a Proof of Concept (PoC) platform.**

***Index Terms*—Embedded Network, Tool Interoperability, Ethernet TSN, YANG model**

I. Introduction

Aerospace and automotive industries are moving to Software Defined Systems (SDS) based on virtual resource allocation: a control layer is deployed in the system in order to configure virtual resources for the services offered by the specific domain application. In this sense SDS breaks the tight coupling between the software (SW) and the hardware (HW) of classical information systems, where dedicated device performs the domain application service. Such systems offer an easier management of HW and SW resources for the application but require higher design constraints for a flexible and generic execution platform. At the same time, classical information systems lack of scalability, flexibility and configurability. The new SDS-related architecture paradigms can be applied to critical embedded systems and the associated communication network where the increase of data exchange in terms of bandwidth, the rising number of devices deployed in the network and the need of standardization of the network interfaces suggest to naturally move to switched Ethernet network, with the advantage of reuse and cost reduction on the overall infrastructure. The critical expected properties of embedded applications with mixed criticality traffic imply preconditions on the network, such as tight and predictable communication traversal time, reliable and safe communication, strong availability of data/frame exchanged coupled to network configuration complexity. In this context, the Ethernet Time-Sensitive Networking (TSN) can satisfy the requirements of safety-critical embedded applications. The IEEE802.1Q [1] standard complemented with TSN standard extensions emerging in industrial/automotive domain proposes dedicated means to guarantee real-time and safety on these mixed criticality embedded systems. Nevertheless, the design, the provisioning and the verification of the application requirements of a full TSN Ethernet network in a mixed criticality embedded system requires the usage of specialized tools to let the network engineer be more effective, due to the complexity of the TSN standard. Moreover, these tools are necessary to support certification of the final system.

To simplify exchanges between these different tools, we propose in this article to study the use of the YANG data model for tool interoperability purposes. As YANG is designed for network device configuration and monitoring, we proposed augmentations to the standard models to enable their use in the upstream design, study, test and validation phases of critical embedded networks. These modifications are then tested to ensure automated, flexible and vertical interoperability between a network design and analysis tool and a generic hardware test bench configuration tool. This vertical interoperability has enabled us to explore multiple TSN safety-critical embedded industrial use cases (from design to deployement on a hardware generic test bench) in short iterative loop.

This study has been elaborated in the context of an IRT Saint-Exupery project called EDEN project. EDEN is a multi-domain research project (automotive, aeronautics and space) with the aim to demonstrate confidence in TSN deployment

in critical embedded system domain.

This article digs into the details of the study proposing the following subjects: Section II introduces TSN and YANG. Then, section III analyses the state of art of Ethernet network tools for embedded system. Section IV defines the approach and organization of YANG model applied to Ethernet TSN in the context of EDEN project. Section V documents the use of our YANG models on the industrial use case. Next, section VI discusses the limits of our approach. Finally section VII resumes the conclusion and proposes future work perspective.

II. TSN AND YANG OVERVIEW

TSN provides deterministic and reliable Ethernet communication for real-time traffic, which can coexist with non-real time data traffic. The IEEE802.1Q [1] standard complemented with TSN standard extension is emerging in industrial domain: several working groups for TSN standardisation propose TSN profiles for specific industrial domains. For example, the Aerospace Onboard Ethernet Communications P801.2DP standard is a profile defined for the aerospace industry.

Several TSN's intrinsic mechanisms permit to guarantee QoS for the traffic. The Time-Aware Shaper (TAS), proposed in IEE802.1Qbv [2], combined with network-wide synchronization provided by gPTP protocol, descrided in [3], enable time-triggered communication. Credit-Based Shaper (CBS) enables traffic flow regulation controlled by credit as specified in IEEE802.1Qav [4]. Frame Replication and Elimination for Redundancy (FRER), proposed in IEEE802.1CB [5], allows the frame redundancy. Per Stream Filtering and Policing (PSFP), defined in IEEE802.1Qci [6], enable flow filtering and policing.

Yet Another Next Generation (YANG) is a data modeling language intended for network configuration and monitoring. It was proposed by IETF in RFC6020 [7]. YANG language is used to describe a data structure. Instances of this data structure can be exported in XML or JSON format. These instances travel through the network to configure or monitor a device using YANG-based protocol like NETCONF [8] or RESTCONF [9]. YANG can be seen as a successor of Management Information Base (MIB) and YANG-based protocols as a successor of Simple Network Management Protocol (SNMP).

YANG model files are called `module`. To illustrate this paragraph, a very simple `module` modeling a scientific article is given as an example in Listing 1. The `container` article is a high level object that groups `leaf`, `list` and even `container`. In this `container`, a `list` of section is described. A `list` object is a collection of key/value pair that can contain multiple objects (e.g. `leaves`). In this section `list`, a section-id `leaf` is described. A `leaf` is an object that can contain only one value. An instantiation of this `module` for a two-section article is given in Listing 2. YANG also proposes a mechanism called `augmentation` which allows a `module`, without modifying in it, to be extended by another `module`. In our example, it's possible to propose a module that `augment` the section object of the article `module`, to add a `list` that would describe tables present in the section in the same way as the `list` of images.

```
container article {
    description "Article";
    list section {
        key "section-id";
        leaf "section-id {
            type uint32;
        }
        leaf content {
            type string;
        }
        list image {
            key "title"
            leaf "title" {
                type string;
            }
            leaf "path" {
                type string;
            }
        }
    }
}
```

Listing 1: Simplified example of a scientific article YANG module

```
<article>
    <section>
        <section-id>0</section-id>
        <content>This is the first section
        </content>
    </section>
    <section>
        <section-id>1</section-id>
        <content>This is the second section
        </content>
        <image>
            <title>My chart</title>
            <path>/chart.png</path>
        </image>
    </section>
</article>
```

Listing 2: Simplified xml instantiation of the scientific article YANG module

In practice YANG is mainly used in the IT world to manage large infrastructures. In this context, IETF has proposed a number of models such as model for network and interface, or for protocols such as IP. These models are `augmented` by vendors such as CISCO or HUAWEI to model proprietary mechanisms. In the world of TSN, standardization working group proposes a set of standard YANG `module` to describe TSN mechanisms and enable their configuration and monitoring.

III. RELATED WORK

As mentioned previously, Ethernet IT device providers have developed complete tool suites adopting YANG programmable interfaces using NETCONF or RESTCONF protocol for Ethernet network configuration, supervision, and maintenance [10]. A typical example is the Cisco YANG suite [11] that provides a set of tools and plugins to learn, test, and adopt YANG model for the supervision of an Ethernet network.

For non-IT network, the standardized Centralized Network Configuration (CNC) architecture, part of IEEE Std 802.1Qcc [12], is a key element of TSN standard for configuration of embedded critical network application.

A first implementation of such CNC architecture capable to configure a TSN network of an industrial construction equipment, using the NETCONF protocol and YANG models, was demonstrated in [13]. This demonstration is limited to dynamic TAS configuration for a SMART MPSoC bridge representative of an embedded network application distributed by SoCe company. Despite the use of Linux service for YANG parsing and configuration, the device set-up is operated with specific SoCe drivers, so not applicable to different devices.

In a recent review on TSN network configuration management [14], the authors state that despite the central role of YANG in promoting unified network management, the current standard still needs to be improved to cover configuration of a network composed with different device manufacturers. Similarly in the automotive industry, [15] promotes the use of model-based development, validation and configuration of TSN embedded application. But the authors state that the TSN network configuration and scheduling algorithms are not integrated into the existing software development tools and require further research to enable efficient configuration of TSN network.

Indeed, before configuring the network hardware, it is necessary to design and validate the network configuration using multiple specialized tools (e.g. tools for design, configuration, formal analysis, simulation, ...). In order to use such a variety of tools safely and efficiently, tool interoperability is of major interest. Here also the YANG model can play a central role. To answer the design need, several model-based tools for TSN network architecture design have been proposed on the commercial market. We can mention Pegase from RealTime-at-Work (RTaW) [16] mostly used in automotive industry, Chronos from General Electric [17] targeting aerospace market, TSN designer from RealTime IT [18] or IxNetwork from Keysight [19] addressing the IoT and industrial market. Those products enable to explore and analyze network architecture in order to generate the configuration of TSN mechanisms. But to our knowledge none of them implement a feature enabling non-proprietary tool interoperability all the way down to the hardware network device configuration tool.

RTaW has introduced a first YANG export feature in Pegase tool, completed by a complete tool chain in TSN Studio [20]. TSN Studio enables to configure industrial devices running Linux with standard NETCONF protocol. Despite the use of YANG models, there are still some dependencies to the Pegase data model, for example with regards to the traffic definition or the fixed labeling of physical device. The NETCONF protocol is the only way to configure the devices, which could mismatch the embedded systems requirements. Indeed, NETCONF is quite heavy in term of computing resources and software stack, and it relies on TCP which may not be determinism-friendly. In addition, manufacturers of devices dedicated to embedded systems offer configuration through proprietary solution that do not support YANG interface.

As critical embedded network, industrial IoT domain imposes stringent requirements on the dependability and performance of communication networks. In this context, Chahed et al. [21] explore TSN state of the art after identifying that the large and continuously evolving set of standards poses challenges for adopters seeking to understand it. They exhibit that clear understanding of use-case, available device resources and constraints are key points and raise that the performance of the control plane design and management operation especially for device configuration is an important aspect aimed to be tackled in the future.

To overcome these limitations around configuration integration in development and validation tools as well as TSN network exploration in the context of critical embedded systems, our contribution proposes to complement standard YANG model to enable full interoperability between TSN network design tool chain and device configuration targeting network exploration experiments.

IV. YANG DATA MODEL FOR TSN NETWORK

A. Preamble and initial objectives

To fully understand our approach, it is important to begin by historically describing our issue:

- Initially, interoperability aimed to facilitate the exchange of network models as well as TSN configurations among various design tools (Pegase RTAW and Timaeus-Net in the first step of our project). This initial step was pivotal in shaping our approach to standard Yang models and their limitations, and in subsequent decision-making terms of design. We can refer to this as "**horizontal interoperability**", as interoperability actors have similar or closely related roles in the network development process.
- Subsequently, our objective shifted towards seeking interoperability between network design tools and deployment tools on hardware platforms. Here, we can speak of "**vertical interoperability**", as interoperability actors have different roles. This second step introduced new challenges.

The pursuit of vertical interoperability first faced tools limitations, as tools did not often offer a means of importing from a Yang structure and only supported a single meta-model. Effective interoperability at that time in our project context was thus unidirectional: RTaW-Pegase to Timaeus. The underlying model at this time was already an `augmented` model proposed by RTaW. However, this scenario was interesting as it paved the way for a new potential need: the ability of a tool to support a Yang model as an interoperability "parameter". When we aimed to further enhance this vertical interoperability to ensure the sustainability of our solutions, it became evident that adopting a cleaner approach was essential. This involved creating a customized Yang model that we could subsequently disseminate to the embedded network community.

B. Requirements for a model

To create and setup the YANG model, we started with the definition of requirements, driven by 3 principles: universality, diversity, and reversibility.

a) Universality: Because our final aim is to ensure the global interoperability of our network design and deployment environment, the solution was not only to cover the network configuration, but to have a full description of the network shared by a large set of tools: design and configuration tools, network simulators, deployment solutions. This first principle of universality is crucial regarding both the structure and content of the model. Particularly, to encompass the scope of vertical interoperability, we need to incorporate into the model, information that will be used in a very localized manner. This principle also diverges from the aim of IETF and IEEE models, which solely targeted network devices. For instance, high level network development tools have to deal with pure graphical information related the rendering of networks (e.g. size and position of nodes inside a display). This kind of information has not interest for the network deployment but can be very important for the network designer. So a YANG model shall support any information required along the full network development process.

b) Diversity: At this stage, let's introduce the concept of perspective: for two tools playing the same role in a network design chain, the high-level view of the network may differ. For instance, a critical embedded network would be highly interested in security information, whereas a standard network may not necessarily be concerned with this issue. Another example is openness to applications. For instance, you can limit the traffic definition to the network or extend this definition to the related applications. Indeed, applications can greatly influence traffic behavior and configuration. For example, if the application contains time-triggered temporal constraints and if these constraints are implemented using the time-triggered solution TAS, then the configuration of the TAS shall match these application constraints. Therefore, we need a solution to express these constraints inside the YANG model. This principle of diversity will significantly increase the size of the model; it will also lead to diversity in usage: all elements of the model will not be used in the same way in every network. Thus, there will be diversities in usage or instantiations resulting from the diversity supported by the model as well as the functional diversity of users. So a YANG model shall support any perspective and point of view used along the full network development process.

c) Reversibility: The YANG model shall be sufficient so that when a tool exports a network description, it should be able to recreate the same network description by importing the previously exported model. Due to this principle, new information shall be taken into account: implementation information introduced by each tool. A highly interesting example concerns the modeling of TAS schedulers. When this shaper is used, the network design tool assists by calculating the configuration tables of the TAS schedulers, which contain the opening and closing times of gates on the output ports. This configuration constitutes the implementation of TAS and is specific to each TAS configuration algorithm; for example, in Timaeus-Net, there are options that allow for adjusting the porosity of TAS windows relative to the rest of the traffic. The porosity of a TAS expresses the ability of time triggered traffic to be interlaced with non time triggered traffic. If the porosity is low, then the use of TAS will generate a tunneling effect that will reduce the efficiency of the non time triggered traffic. Changing this option thus leads to a different TAS configuration and different performance for the entire traffic. The modeling proposed by the IEEE standard regarding TAS is very relevant regarding the configuration of gate opening tables. Therefore, this initial information can be reused; however, the IEEE model does not implement any association between the gates configuration and the specific traffic. Thus, if two flows of the same priority and profile (same message size and periodicity) exist, it will not be possible to infer the flows from gate management information. Respecting reversibility can lead to an increase in information: we call this kind of information tool-dedicated information. But other solutions are also possible: in the case of TAS schedulers, for example, we have chosen to address it at the level of the import function itself. Indeed, it would have been too complex and too specific to overload the model in order to trace unequivocally the options of the scheduler creation algorithm. In other words, interoperability stops at the implementation specifics of the exchanging tools. So a YANG model shall support any information which are mandatory to recover a network after an export-import sequence.

d) Maintainability: To conclude this list of requirements, adding one final principle that concerns not the creation of the model but its utilization: the principle of maintainability. Indeed, as soon as we recognize that standard models are incomplete, not yet stabilized, or customizable, we must be prepared in an interoperability scenario to encounter a YANG model that is different from the one we are going to create. Alternatively, for a given model, we must be prepared for different interpretations of the same model. We will delve into the implications of this scalability principle later in the article. This last criteria does not directly concern the YANG model, but rather the way the tools could be adapted to work with such model. Our conclusion of this first activity of requirements, was a set of mandatory information that should be managed inside our model: topology information, traffic information, configuration information, dependability information, tool-dedicated information etc.

C. Elaboration of the YANG model

Implementing the model involves defining the classes and relationships necessary to cover our network modeling needs. At this stage, we are torn between two implementation approaches that need to be reconciled: a primary categorical approach, which is to reuse existing classes and relationships from IEEE and IETF modules whenever possible – this is imperative, just because during the deployment phase, we will need to leverage this information. A secondary contingent approach is to address our principles of universality, diversity, and reversibility. To cover the first approach, our starting point was the set of concepts already implemented inside IETF and IEEE standards; IETF standards contain concepts like net-

work, node, link, interface, etc.; IEEE standards contain TSN concepts like bridge, bridge-port, TT scheduler, gPTP/PTP instances, talkers and listeners, etc. We split these standard concepts into 3 categories:

- concepts that can be reused as they are,
- concepts partially matching our requirements and which have to be improved,
- concepts not mandatory for HW configuration, which are not matching our requirements or which could not be improved.

In the first category (concepts that can be reused as they are), we only have concepts which are required for the hardware configuration itself: for instance, to configure the TAS mechanism, we need to define "gate parameter tables" which are containing the TAS schedulers entries. The YANG concepts defined in the ieee802-dot1q-sched module play this role perfectly. The second category (concepts partially matching our requirements) contains most other standard concepts: improvement can here be done using the YANG `Augment` feature. For this study, we organized the YANG folders in order to separate what is standard, and what is customized. Let's note that even the most obvious concepts must be completed to fit our needs; for instance, the IETF node concept has been augmented to support the "manufacturer-reference", or the "bench-id" (host name of the node in the test bench network, such as "PC_2"). These fields are useful when deploying the network on a generic platform. The last category contains concepts that are not mandatory for configuration, and which are not generic nor detailed enough to implement some of our needs: for instance, the talker-listener paradigm is defined inside the ieee802-dot1q-tsn-types module and is not relevant to implement all kind of traffics. In order to avoid modifications or patches of the standard model, when a part of the model was too far from our need and when it was not possible to augment it, we preferred to create our own class breakdown.

To address the second approach, we need to complement our model with new classes or relationships. Here, we have more freedom, but our approach aims to achieve a complete and high-quality model: at this stage, we aim to ensure that additions do not degrade the overall quality. Once again, compromises will need to be made between the two initial objectives (horizontal and vertical interoperability). To measure the quality of the whole, we use conventional design criteria borrowed from the state of the art in model design: class coherence (classes should implement only one clear and unique concept); coupling (classes should be loosely coupled, avoiding logical and implementation couplings); primitiveness (each class attribute should implement information that cannot be decomposed into elementary information or duplicate already modeled information).

D. Model Overview

We will not expose the entirety of the model (see summary in Table I) but will briefly present some specificities. Let's start with the "additions," which are a few modules we have created to complement existing elements in the standards:

- Irt-tsn: This module contains information for TSN configuration. The simplest example here is CBS. Although very old and common, there is no standard for defining the configuration of CBS parameters.
- Irt-topology, irt-interface, irt-ptp: In these three cases associated with standard modules, we have added informative supplements such as buffer sizes, transmission capacities, references to network nodes (for interfaces as with "bench-id" in the interface module presented earlier), etc..

Now let's talk about classes that we have created from scratch. The simplest example is traffic modeling (Fig 1). The irt-traffic module is intended to cover the generic needs for defining traffic that we identified in our research project. Main container is the "Traffic" class: this singleton is supposed to contain all the items used to describe a full traffic. It mainly contains:

- A collection of flows: each flow itself contains a set of cast ; a flow has properties (like its payload or its period) and constraints (like a maximal latency)
- A collection of classes: a class is a high level concept to describe QoS or standard shapers configuration.
- A collection of protocols: a protocol can be used by a flow
- A collection of additional constraints: these constraints are used to express applicative constraints: for instance, for a cyclic traffic, the exchanged windows or for a chained cyclic, the items of the chain.

E. YANG model instance life cycle in the tool chain

According to the interoperability context, several activities are concerned: network model design, network model validation, devices configuration generation, deployment on the HW platform, validation of the HW platform. These activities can be organized around a workflow, creating a life cycle to enrich the YANG instance of the network.

Parts of the YANG model have to be initialized in the design activity, other will be created later. For instance, the actual IP address is only assigned during the platform deployment. This concept of life cycle can be compared to the "config" standard YANG field used to distinguish parameters that actually can be configured: in our case, model concepts have to map to activities of the life cycle. Sometimes, the same initial requirement projected onto two activities will be implemented using two distinct YANG concepts: it is the case of the Traffic Safety requirement. At network design level this requirement consists in being able to define for critical flows some redundant paths. To implement this first level of requirement, due to the reversibility principle, we decided to create the concept of flow cast: a cast is a set of segments; each segment can be single or multiple; a single segment is a path; a multiple segment is a set of paths, having same extremities but distinct intermediate nodes to ensure safety in

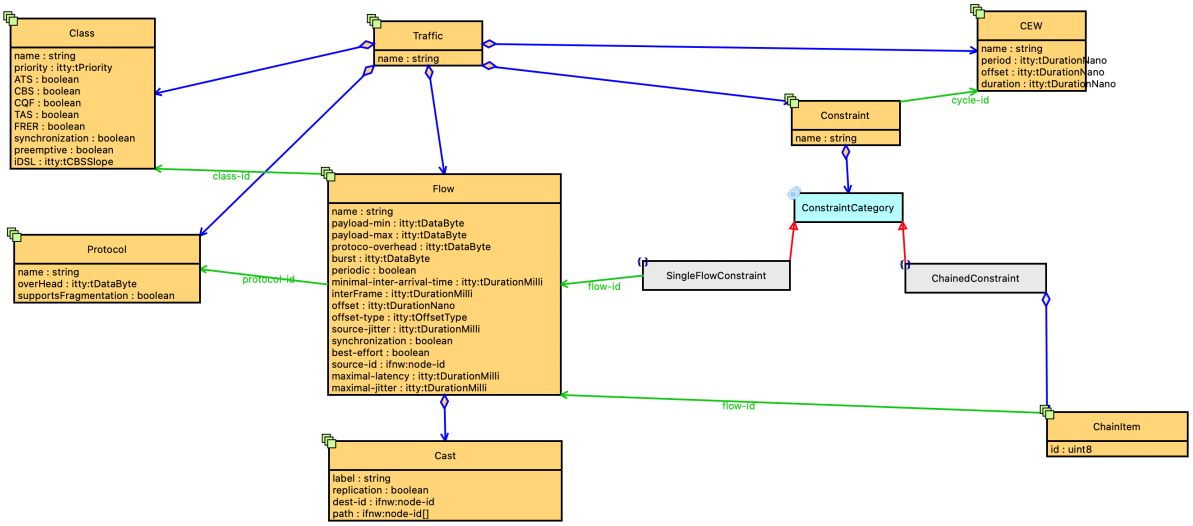

Fig. 1: Class Diagram for Traffic

case of line break. Then at deployment level, the principle is to use the FRER mechanism to implement the redundant paths. A model complement is thus required, focusing on each node ieee802-dot1cb-frer configuration.

F. YANG model implementation in the tools

Both interoperability objectives (horizontal and vertical interoperability) suppose that each tool of the framework is "understanding" the common YANG data model. However, in each tool there is an element of interpretation in this understanding of the model. For example, our YANG model leaves some freedom for port naming (numeric, alphanumeric, etc.); the nature of the tools will also play a role: a tool close to hardware will rely on the hardware identification of the ports while a high level network analysis tool can go so far as to ignore this naming. The consequence is that each tool must be able to adapt its own data model to the generic YANG model. The tools must allow this adaptation to be configured, in order to achieve maximum interoperability.

To ensure the scalability principle of interoperability, the technique of meta-model mapping can be a solution. This technique is based on the following principles:

- Each tool has its own meta-model, which generally corresponds to the tool's specific design, but also to the underlying purpose of the tool.
- Each tool will use an external YANG meta-model, which is different from its own meta-model. This external meta-model will not be hard-coded in the tool but exchangeable: the tool will therefore offer a principle to select the external YANG meta-model.
- An additional mapping interface will be supported by the tool, an interface that should allow aligning the elements of the native meta-model with the elements of the external meta-model

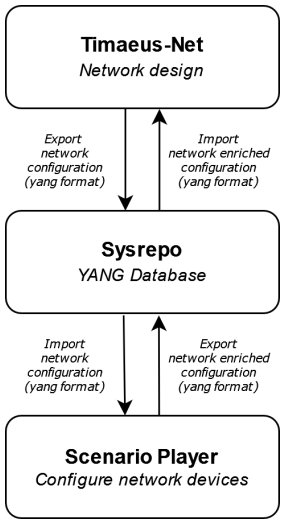

Fig. 2: Block diagram of the tool chain

V. The Experimental Setup

In this section, the experimental setup used to validate the interoperability enabled by the model is presented. Starting with a presentation of the tool chain, followed by a presentation of the industrial case study used for this setup, and finally a feasibility study is described and discussed.

A. Tool chain

The tool chain developed as part of this work aims to support the user from the design and configuration of a TSN network right through to deployment on the hardware. It is summarized in Fig. 2. It can be broken down into three distinct tools, which are detailed below.

a) Design and validation of a TSN network: Timaeus-Net [22] is a tool allowing the user to describe a network, its message flows and their constraints. Using these informations,

Standard	Standard changed	Contribution
iana-type-if.yang	ieee802-dot1q-psfp.yang	irt-eden-usecases.yang
ieee1588-ptp.yang	ieee802-dot1q-sched.yang	irt-frer.yang
ieee802-dot1as-ptp.yang		irt-interface.yang
ieee802-dot1cb-frer-types.yang		irt-ptp.yang
ieee802-dot1cb-frer.yang		irt-topology.yang
ieee802-dot1cb-stream-identification-types.yang		irt-traffic.yang
ieee802-dot1cb-stream-identification.yang		irt-tsn.yang
ieee802-dot1q-ats.yang		irt-types.yang
ieee802-dot1q-bridge.yang		
ieee802-dot1q-preemption.yang		
ieee802-dot1q-stream-filters-gates.yang		
ieee802-dot1q-tsn-types.yang		
ieee802-dot1q-types.yang		
ieee802-types.yang		
ietf-inet-types.yang		
ietf-interfaces.yang		
ietf-ip.yang		
ietf-network-topology.yang		
ietf-network.yang		
ietf-yang-types.yang		

TABLE I: Lists of YANG models used

the tool proposes a configuration of the different TSN mechanisms. Compliance with flows constraints such as latency/jitter is then validated using Network Calculus Approach [23] [24] to compute them in the worst case. When the user is satisfied with their configuration, the tool can then export these different parameters according to the YANG data model described above.

b) Central YANG models storage: These YANG datas are then imported and stored in a centralized database that ensures compliance with the format described in the model. The database used in our case is an open source project called Sysrepo [25]. This database can be queried using NETCONF and it enables interoperability between tools. In the future, other tools (e.g a simulator or analysis tool) could also query this database to perform computations on the network described using Timaeus-Net or other design tools that can export the data in the format described above.

c) Deployment on the targeted hardware network: To deploy the network designed with Timaeus-Net on the hardware, a tool called Scenario Player has been developed. This tool begins by retrieving the configuration described in the central database. It then allows users to describe test scenarios, with fields such as "duration" of the use case `module`, and to completes the Timaeus-Net data by adding information describing the test bench via a graphical user interface. To be more precise, the user maps the objects (end stations, switches, interfaces, ...) described in Timaeus-Net to the test bench hardware using above-mentioned fields such as "bench-id" and "manufacturer-reference". The tool then uses these fields to complete the data with the MAC and IP addresses of each device and generate forwarding and ARP tables. These completed datas are then sent back to the central YANG database for storage and potential use by other tools. Next, Scenario Player configures the various test bed devices (switches, end-stations, traffic generators, measurement instruments) using the completed datas describing the previously designed configuration. Finally, it executes the scenario described, enabling experimental measurements to be carried out on complex use cases.

Note that none of the devices used in the test bed supports a standardized configuration/monitoring protocol such as NETCONF, RESTCONF or other configuration protocol more suited for the critical embedded world. Therefore, Scenario Player translates the configuration of each device according to the proprietary configuration protocol. However, NETCONF is used by our tools to exchange data with the SYSREPO database (the four arrows in Fig. 2).

B. Spatial use case

The following industrial case study was used to investigate the interoperability capabilities of the model described above. This case study is based on the unification and replacement of a satellite's current networks (i.e. MIL-STD-1553 and SpaceWire) using a single 1Gb/s TSN network with 4 switches and 14 end stations. It is described in detail by Chaine et al. in [26].

However, due to hardware limitations (e.g. number of ports or FRER support on Network Interface Cards (NICs)), the case study was reduced to the topology described in Fig. 3. It consists of 5 switches and 10 end stations. The switches SWOBC_A and SWRIU_A have been introduced to overcome the limitation of FRER use on NICs for two end stations, i.e. OBC_A_FhI and RIU_A. From a traffic point of view, there are 101 flows, which have been transposed from the flows transiting on the MIL-STD-1553 and SpaceWire networks currently in use. Theses heterogeneous flows carry a payload between 2 and 1472 bytes with burst from 1 to 4788 packets. They are grouped by similarity into 13 traffic classes. The lowest latency and jitter constraints for each traffic class are summarized in Table II. To meet these constraints, Timaeus-Net has proposed and configured different TSN latency control mechanisms (i.e. CBS and TAS) for the different traffic classes. The use of the TAS imposes a network-wide common clock. This is provided by the gPTP synchronization protocol.

Traffic class Id	1	2	3	4	5	6	7	8	9	10	11	12	13
Lowest latency constraint (ms)	33	100	125	N/A	N/A	31.25	125	1	125	125	N/A	N/A	N/A
Lowest jitter constraint (ms)	N/A	1	10	N/A	0.001	N/A	0.5	0.1	N/A	N/A	N/A	N/A	N/A

TABLE II: Lowest duration constraint for each traffic class

To meet availability requirements, three synchronization domains are used. And finally, the flows between OBC_A_FhI and RIU_A are replicated using the FRER mechanism on the first switch in the path and eliminated on the last.

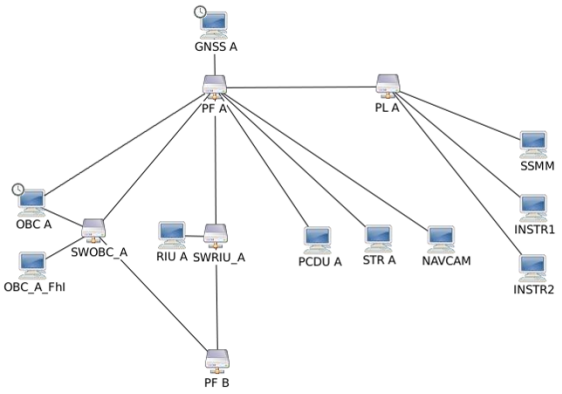

Fig. 3: Unified TSN network satellite topology (view from Timaeus-Net)

C. Feasibility study

To illustrate the interoperability of our augmented YANG model, the case study described above is designed, configured, validated and deployed using the tool chain.

The first step in this tool chain is to design and configure the network. The topology and message flows (periodicity, size, latency constraints, etc.) of the satelitte case study are first described in Timaeus-Net. Then, after exploring several configurations using formal analysis and optimization algorithms, a configuration is selected and then exported as a set of xml files following the data format described in the YANG model. These files are then loaded into the database using NETCONF.

Scenario Player then queries the database using NETCONF to retrieve the network described and configured above. It then enriches this data with the information required for deployment on the hardware. For example, each switch, end station and network interface described in Timaeus-Net is assigned to a switch, end station or interface available on the test bench. These assignments also trigger the addition of data such as the forwarding tables needed to configure flow paths, or the information needed to configure traffic generators such as source and destination mac addresses and VLAN numbers for each flow. These new datas are then sent to the database, once again using NETCONF, for future use by Scenario Player or other tools.

All the data are then parsed, separated by device and used to configure the corresponding devices. In this case study, two types of switch from two different manufacturers, 3 desktop computers running Ubuntu 20.04 totalling 8 NICs, and 2 FreeRTOS targets are automatically configured using the data stored in the database.

Although imperfect, the following metrics illustrate well the complexity of the case study deployment, due to the number of parameters to be configured, and the importance of the effort put into tool interoperability.

The first of these metrics is the number of `leaves` stored in the database. When first imported, this number is 4138 `leaves`. After adding test bed specific datas, it increases to 6462 `leaves`. The distributions are detailed in Table III. We can note that 87.4% of the `leaves` in the first import are additions proposed by our contribution. 93% of these are linked to the description of traffic crossing the network. After adding test bed specific datas, our contribution represents only 58.7% of the `leaves`. This is mainly due to the creation of forwarding tables stored in the ieee-bridge model and the addition of ARP tables in the ietf-interface model. Other industrial case studies designed and deployed with this toolchain showed different distributions (e.g. 10301 `leaves`, 45% of which came from our contribution), explained by differences such as fewer flows with more hops, different TSN mechanisms and larger numbers of interfaces. The differences caused by the choice of TSN mechanisms are illustrated in Table IV, which shows three different configurations on the same topology of an industrial automotive network use case. Table V describes a metric that is more stable to changes of use case. It details the distribution of `leaf` types used to meet our interoperability needs. We can see that few data types are needed (114 after enrichment) to describe and deploy a network, but it's the instantiation of these that greatly increases the number of `leaves`. Take the example of the 41 `leaves` needed to describe a flow, but the instantiation of this model for 101 flows leads to the use of 3483 `leaves` on the satellite use case. We also note that 66.7% of the `leaf` types used in this use case originate from our contribution, which highlights the shortcomings of standard models in terms of tool interoperability.

The second metric is the number of commands made by Scenario Player to configure the 15 devices that reach 2919. This total is broken down into 2199 commands for switches and 720 commands for end stations.

The next metric is the time required to deploy a case study on the hardware. Using the data exported by Timaeus-Net, an experienced user can perform model enrichment in less than ten minutes, and automatic hardware configuration takes less than two minutes. Without this tool interoperability, an experienced user would need at least 8 hours and 45min to reproduce the configuration described in Timaeus-Net. This duration is deducted from the time needed to configure the PL A switch by hand (35min multiplied by 15 devices). This

	Original import		Enriched import	
	Standard	IRT	Standard	IRT
ietf-network	143	106	143	133
ietf-interface	352	112	646	156
irt-usecase	0	0	0	3
ieee-bridge	0	0	1841	0
irt-traffic	0	3382	0	3483
ieee1588-ptp	25	18	39	18
Total	520 (12.6%)	3618 (87.4%)	2669 (41.3%)	3793 (58.7%)

TABLE III: Distribution of the standard and IRT `leaf` instantiations for the two importations of the use case in the central database

Configuration	CBS	FRER	Std	Irt	Total
1	No	No	990 (27%)	2729 (73%)	3719
2	Yes	No	990 (23%)	3241 (77%)	4321
3	No	Yes	990 (26%)	2799 (74%)	3789
4	Yes	Yes	990 (23%)	3311 (77%)	4391

TABLE IV: Distribution of the standard and IRT `leaf` instantiations after first importation of three network configurations of an automotive use case

duration is probably very optimistic, as the PL A is the switch with the simplest configuration (82 commands). Moreover, this metric does not take into account the possibility of human error.

And finally, on the other side of the tool chain, this interoperability allows us to change design tools without impacting the rest of the chain.

VI. Discussion and limits

We demonstrated the feasibility and benefits of tool interoperability using YANG models in the previous section, however there are limitations which are discussed in this section.

First limitations are about the design of the model itself, according to the criteria that we have defined. The universality and diversity principles have conduct us to cover all points of view and all needs for modeling a network and associated traffic. This has led to add or amend missing elements being to the standard models. However, these standard models will evolve and will certainly be partially completed. As a consequence, our add-on shall be updated.

The reversibility principle has highlighted that the cost of implementing an import feature is much higher than that of cost for the export features. Therefore, for a commercial tool the benefits of importing is limited because it does not provide any direct extra features in its own solution.

The last principle, namely maintainability, leads to establish a specific implementation which consists in considering the YANG meta-model as a parameter of the interoperability features. This can be complex especially if the tool does not implement a meta-model to manage its data and/or if it contains limitations. Indeed, some tools of the tool chain are more or less impacted than others by change in the associated models. For example, Timaeus-Net has its own internal meta-model. The data mapping with the YANG model is direct, so a change in model can be handled easily by a few change in the transformation mapping. Unlike Scenario Player that implements a simple parser from YANG to the proprietary configuration tools of the various devices, any change in the YANG model may require more consequent code evolution. This enforces the question of the stability of YANG models and the interconnection between the model and the tools.

Other limitations are inherent to the standard YANG models. One can cite the lack of completeness between the various module creators, for example such as the IETF or IEEE, or the restrictions imposed by certain models. For example, the fact that the standard interface model proposed by the IETF only enables the modeling of single device interfaces is a significant limitation when using the YANG model to describe a network that will inevitably contain multiple devices. This limitation arises from the fact that it's not possible to reference YANG elements which are not explicitly defined as `prototypes` from another element. Therefore, it's not possible to create for each node of the network, the collection of its interfaces. In our case, we used workaround in such cases but standard evolution may be needed. Note that, this limitation is not encountered with the normal use of YANG, but only when trying to describe a complete network, as in our work.

Finally, the last limitation is economic. Indeed, the market of TSN network design and configuration tool for critical embedded system is very fragmented and today established as a niche market whereas tool interoperability is a competitive criteria. The deployment of YANG models as core technology for vertical and horizontal tool interoperability standard for classical IT domain, even supported by vendor specific extension (e.g. with proprietary YANG `augments`), needs to be organized around an alive and open industrial ecosystem. This would enable to increase the YANG models maturity and enlarge the community for additional TSN tool market such as network simulator, network verification and validation, etc. to foster a non-competitive technology facilitating tool supplier collaboration with positive return on invest. Through the work presented in this article with the support of industrial partners and IRT Saint Exupéry hosting TSN research activities, this contribution aims to push forward a YANG ecosystem related to embedded critical system.

VII. Conclusion and future work

In this paper, we propose to `augment` the standard YANG models to ensure vertical interoperability of the tools needed to develop and deploy complex TSN networks. We illustrate the usefulness of this new model by using it to connect two tools

	Original import		Enriched import	
	Standard	IRT	Standard	IRT
ietf-network	8	14	8	15
ietf-interface	16	4	19	5
irt-usecase	0	0	0	3
ieee-bridge	0	0	8	0
irt-traffic	0	41	0	42
ieee1588-ptp	7	3	11	3
Total	31 (33.3%)	62 (66.7%)	46 (40.4%)	68 (59.6%)

TABLE V: Distribution of the standard and IRT `leaf` type for the two importations of the use case in the central database

used to design and configure a satellite's critical embedded TSN network and deploy it on hardware targets. The proposed models and `augmentations` are delivered in open source [27] to initiate an ecosystem dedicated to TSN tool interoperability for embedded critical application. However, this first version needs to be challenged by commercial products that aim to offer interoperable solutions for TSN networks in order to evolve.

Several future works are envisioned around this model in a follow up project. First, we planned to extend the model to support more feature like instrumentation devices. Other tools, such as a network simulator, are also planned to be connected to the central database to take advantage of this unique interface and will most probably require new additions to the model. Finally, this work will also be continued to propose a YANG-based configuration and monitoring protocol adapted to the world of critical embedded systems.

ACKNOWLEDGMENT

The authors thank all people and industrial partners involved in the EDEN project. The French Research Agency (ANR) and the partners of IRT Saint-Exupéry Scientific Cooperation Foundation support this work: Airbus Operations, Airbus Defence and Space, CNES, Continental Automotive, INPT/IRIT, ISAE-SUPAERO, ONERA, Safran Electronics and Defence, Thales Alenia Space and Thales Avionics.

REFERENCES

[1] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018)*, pp. 1–2163, 2022.

[2] "Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[3] "Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)*, pp. 1–421, 2020.

[4] "Ieee standard for local and metropolitan area networks– virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. 1–72, 2010.

[5] "Ieee standard for local and metropolitan area networks–frame replication and elimination for reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017.

[6] "Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 28: Per-stream filtering and policing," *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*, pp. 1–65, 2017.

[7] M. Björklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc6020

[8] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6241

[9] A. Bierman, M. Björklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8040

[10] J. Schönwälder, M. Björklund, and P. Shafer, "Network configuration management using netconf and yang," *IEEE communications magazine*, vol. 48, no. 9, pp. 166–173, 2010.

[11] "Cisco yang suite," https://developer.cisco.com/yangsuite/.

[12] "Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, 2018.

[13] I. Álvarez, A. Servera, J. Proenza, M. Ashjaei, and S. Mubeen, "Implementing a first cnc for scheduling and configuring tsn networks," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022, pp. 1–4.

[14] B. Shi, X. Tu, B. Wu, and Y. Peng, "Recent advances in time-sensitive network configuration management: A literature review," *Journal of Sensor and Actuator Networks*, vol. 12, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2224-2708/12/4/52

[15] M. Ashjaei, L. L. Bello, M. Daneshtalab, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of systems architecture*, vol. 117, p. 102137, 2021.

[16] "Pegase, realtime-at-work," https://www.realtimeatwork.com/rtaw-pegase/.

[17] "Chronos, general electric," https://www.ge.com/research/project/time-sensitive-networking-tsn.

[18] "Tsn designer, realtime it," https://www.realtime-it.de/en/index.php/tsn-designer/.

[19] "Ixnetwork, keysight," https://www.keysight.com/sg/en/products/network-test/protocol-load-test/ixnetwork.html.

[20] "Tsn studio, realtime-at-work," https://tsn.studio/.

[21] H. Chahed and A. Kassler, "Tsn network scheduling—challenges and approaches," *Network Journal*, vol. 3, pp. 585–624, 2023.

[22] M. Barbero, T. Leydier, P. Cuenot, D. Fruchard, and B. Attanasio, "How to design a safe Ethernet TSN network on spacecraft application," in *Data Systems in Aerospace (DASIA 2023)*, 2023.

[23] J.-Y. Le Boudec and P. Thiran, Eds., *Network Calculus*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 3–81. [Online]. Available: https://doi.org/10.1007/3-540-45318-0_1

[24] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," 03 2021.

[25] "Sysrepo storing and managing yang-based configurations for unix/linux applications," https://www.sysrepo.org.

[26] P.-J. Chaine, M. Boyer, C. Pagetti, and F. Wartel, "TSN Support for Quality of Service in Space," in *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, Jan. 2020. [Online]. Available: https://hal.science/hal-02441327

[27] "Eden yang, irt saint exupéry," https://sahara.irt-saintexupery.com/embedded-systems/eden-yang.