



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :

Pierre-Julien CHAINE

le mardi 21 juin 2022

Titre :

Adaptabilité de Time Sensitive Networking aux exigences de
l'industrie aérospatiale

École doctorale et discipline ou spécialité :

ED MITT : Informatique et Télécommunications

Unité de recherche :

Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

Mme Claire PAGETTI (directrice de thèse)

Jury :

M. Jean-Luc SCHARBARG Professeur INPT-ENSEEIH - Président

Mme Liliana CUCU-GROSJEAN Directrice de recherche INRIA Paris - Examinatrice

Mme Ahlem MIFDAOUI Professeure ISAE-SUPAERO - Examinatrice

Mme Claire PAGETTI Ingénieure de Recherche ONERA - Directrice de thèse

M. Sébastien PILLEMENT Professeur Polytech'Nantes - Rapporteur

M. Ye-Qiong SONG Professeur Université de Lorraine - Rapporteur

"The pessimist sees difficulty in every opportunity. The optimist sees opportunity in every difficulty."
Winston Churchill

Remerciements

Je souhaite tout d'abord remercier très sincèrement ma directrice de thèse, la Maîtresse de Conférences Claire Pagetti, sans qui l'aboutissement de ces travaux n'aurait pu être possible. Ton accompagnement sans faille, tant sur le plan technique que sur le plan moral, m'a permis de traverser cette épreuve. La thèse est en effet autant un incroyable travail technique qu'un défi moral. Je me rappellerai longtemps de nos échanges où je finissais toujours par te dire que "encore une fois tu avais raison" et tu me répondais simplement "c'est l'expérience". Merci Claire.

Je souhaite ensuite remercier tout aussi sincèrement mon co-directeur de thèse, le Maître de Conférences Marc Boyer. Ton expérience des réseaux embarqués industriels et ta connaissance de la communauté locale et internationale de ce domaine sont pour beaucoup dans la pertinence et le succès de mes travaux. C'est grâce à toi, en proposant le sujet initial à Airbus, que cette thèse a pu voir le jour. Je retiendrais surtout nos longues discussions du lundi après-midi, parfois un peu philosophiques, parfois plus techniques, sur la compréhension des standards de TSN ou des réseaux de manière plus générale. J'ai eu un immense plaisir à travailler avec toi et je me réjouis que nous continuions à collaborer après la thèse. Merci Marc.

Enfin, je souhaite remercier très chaleureusement mon co-directeur du monde industriel, Franck Wartel. Nous nous rencontrons lors de mon stage de fin d'études pendant le printemps/été 2018. A la fin de ce stage, nous décidons de continuer à travailler ensemble et tu acceptes d'encadrer mes travaux. Tu as été pour moi un mentor, me faisant profiter tant de tes connaissances techniques que celles du monde de l'entreprise. Notre complicité au travail et dans les déplacements professionnels que nous avons faits ensemble m'a beaucoup touché. Merci Franck.

Mes remerciements vont ensuite aux membres du jury pour l'intérêt qu'ils ont porté à mes travaux. En particulier, je souhaite remercier le Professeur Sébastien Pillement et le professeur Ye-Qiong Song pour leur lecture approfondie du manuscrit et les remarques pertinentes dans leurs rapports ainsi que durant la soutenance. De plus, je souhaite remercier la Professeure Alhem Mifdaoui, le Professeur Jean-Luc Scharbarg ainsi que la Directrice de Recherche Liliana Cucu-Grosjean pour interventions enrichissantes lors de la soutenance.

Je souhaite aussi remercier l'entreprise Airbus Defence and Space SAS ainsi que l'Agence Nationale pour la Recherche Technologique (ANRT) pour avoir approuvé mon projet de thèse CIFRE et accepté de le financer. J'ai une pensée toute particulière pour Dominique Pibrac et Antoine Certain qui ont grandement contribué à la réussite du montage de ce projet.

Je ne peux pas oublier de remercier mes collègues d'AIRBUS ainsi que mes collègues docteurs et docteurs-en-devenir à l'ONERA. C'est grâce à votre soutien au quotidien que j'ai pu traverser, non sans un certain plaisir, ces trois années de thèse. Une pensée pour mes partenaires de "pause" à l'ONERA Arthur, Damien, Felix, Hedwinn, Iryna, Iban, Lucien; nos brefs moments ensemble me manquent déjà. Une pensée aussi pour Damien, Matthieu, Jérôme, Thomas, Philippe, Sylvain, Marie, Valentin et Anthony pour leur sympathie au quotidien.

Pour terminer, je souhaite remercier mes proches notamment Hugo et Romain pour m'avoir supporté, dans tous les sens du terme, dans cette épreuve difficile. Bien évidemment, un immense merci à parents et mon cher frère pour avoir toujours cru en moi et pour m'avoir soutenu dans mes différents projets, quels qu'en soit la nature.

Merci.

Abstract

The spacecraft industry is facing a new challenge: offering new capabilities and new missions around Earth, in the solar system and beyond. This will not be achievable without providing more performance on board satellites. In particular, the current satellite network technologies will not be able to handle this increasing demand for long. This leads the spacecraft industry to consider an upgrade of their on-board networks. Such an upgrade is also an opportunity to envision a disruptive evolution of the current communication architecture and move from a network system relying on MIL-STD-1553 for real-time traffic and Spacewire for high throughput traffic to a unified network relying on a single technology supporting both types of traffic. One technology is raising the interest of satellite manufacturers: IEEE Time Sensitive Networking (TSN), the state-of-the-art Ethernet technology, deemed to be capable of supporting the needs of next-generation satellites. The goal of the PhD was thus to assess the suitability of TSN with respect to space systems requirements.

In order to address this problem, we first identified a set of technologies – Ethernet, ARINC 664, TTEthernet, TSN and SpaceFibre – potentially able to answer to the future application demand. We made a qualitative comparison of those technologies with respect to what is foreseen as the expected requirements. The comparison was based on three properties dealing respectively with network performance, time management and fault tolerance. This led us to select three suitable candidates including TSN. While the two other candidates had already been studied and even started to be included in satellite network designs at the time of writing this manuscript, TSN was completely unknown to the spacecraft industry.

After this preliminary step, we studied in depth the set of more than twenty standards that TSN encompasses, each composed of several mechanisms and each mechanism composed of several parameters. We identified IEEE 802.1Qbv Time Aware Shaper standard as the core TSN standard capable to satisfy the network performance requirements. In addition to Qbv, we discussed the interest of other TSN standards (i.e. IEEE 802.1Qci, 802.1CB, 802.1AS, 802.1Qbu), which are now being fitted into a TSN aerospace profile in an IEEE/SAE joint effort.

The next step was then to define a strategy to automatically compute suitable configurations for the standards in this profile. By configuration, we mean a set of assigned values for all the parameters of all the used mechanisms of the list of considered standards. Due to the huge number of parameters' values to set, automatic strategies are a real game-changer to pave the way for large scale industrial use of TSN.

In our approach, we focused on the configuration of TSN Qbv standard alone so that the network copes with the satellite performance requirements. At that stage, we considered that fault tolerance capabilities could either be provided by the network or the applications running over the network. While automatic configuration strategies relying on a network wide schedule of frame transmissions are the dominant approach in the literature, we proposed a brand new configuration strategy called Egress TT. In practice, Egress TT configurations consist of scheduled frame transmissions on the last hop port in the path of any flow. What happens between the source and the last hop port may be variable, as it depends on the time at which the message is emitted by the source and on the delays encountered in the previous hops. Nevertheless, the variability of this delay is absorbed by a correctly chosen release instant in the last hop port. This novel configuration strategy improves the scalability of configuration strategies from the state-of-the-art and reduces the necessary development effort for the upgrade of legacy application software towards a next-generation on-board satellite network system.

Résumé Français

L'industrie aérospatiale fait face à un nouveau défi : proposer de nouvelles fonctionnalités et de nouvelles missions autour de la Terre, dans le système Solaire et au-delà. Ces nouveautés ne se feront pas sans une amélioration de la performance à bord des satellites, notamment au niveau de l'architecture de communication. C'est la raison pour laquelle l'industrie aérospatiale envisage un changement radical de ses réseaux embarqués, passant du bus MIL-STD-1553 pour le trafic temps réel et Spacewire pour le trafic haut débit, à un réseau «unifié» reposant sur une technologie unique capable de transporter ces deux types de trafic. Au début de la thèse, IEEE Time Sensitive Networking (TSN), la technologie état de l'art d'Ethernet, a commencé à attirer l'attention de différents acteurs du spatial. De fait, le but de cette thèse a été de mettre en évidence l'adéquation de TSN avec les exigences de l'industrie aérospatiale.

Afin de résoudre ce problème, nous avons commencé par identifier un ensemble de technologies – Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking et Spacefibre – a priori capables de répondre aux besoins des futures missions. Nous avons ensuite proposé une comparaison qualitative de ces technologies en se basant sur leur compatibilité avec les futures exigences des satellites. Cette comparaison s'est organisée autour de deux thèmes : qualité de service (i.e. performance réseau et tolérance aux fautes) et gestion du temps. Elle nous amènera à sélectionner trois candidats : TTEthernet, Spacefibre et TSN. Tandis que TTEthernet et Spacefibre étaient déjà connus et commençaient même à être intégrés dans des architectures réseaux embarqués satellite au moment d'écrire ce document, Time Sensitive Networking était lui totalement nouveau pour l'industrie aérospatiale.

Ainsi, après cette étape préliminaire, nous avons étudié en profondeur les très nombreux standards de TSN. Nous avons identifié IEEE 802.1Qbv dit Time Aware Shaper comme le standard TSN indispensable pour répondre aux exigences en performance réseau des futurs satellites. Nous avons par ailleurs discuté de l'intérêt d'autres standards TSN (i.e. IEEE 802.1Qci, 802.1CB, 802.1AS, 802.1Qbu) qui sont, avec Qbv, en voie d'être inclus dans un profil TSN dédié à l'industrie aérospatiale.

Afin de valider la compatibilité de TSN, nous nous sommes intéressés à la génération de configurations TSN. Cette tâche n'est pas aisée car chaque configuration nécessite d'instancier un très grand nombre de paramètres. De fait, ces configurations sont presque toujours générées de manière automatique. Cette automatisation est un véritable levier dans l'industrialisation du TSN, à la fois dans les satellites, et d'autres domaines d'application. Ainsi, nous nous sommes concentrés sur la configuration automatique du standard Qbv afin d'adresser les besoins en performance, considérant que les fonctions de tolérances aux fautes pouvaient être reléguées au niveau applicatif. Alors que les stratégies automatiques reposant sur des émissions planifiées à date fixe dans tous les équipements du réseau étaient très répandues dans l'état de l'art, nous avons proposé une nouvelle stratégie de configuration intitulée Egress TT. En pratique, les configurations Egress TT reposent sur des émissions planifiées à date fixe seulement dans le dernier équipement du trajet de n'importe quel flot. Le délai d'un message entre sa source et le dernier équipement dans son trajet peut être variable. En effet, il dépend de l'instant auquel le message a été émis à sa source et aux potentiels ralentissements qu'il rencontrerait dans le réseau. Néanmoins, ce délai variable est absorbé par une planification des émissions bien choisie au dernier saut. Cette nouvelle stratégie propose un meilleur passage à l'échelle que les stratégies existantes. Elle permet aussi de réduire l'effort de développement nécessaire pour la mise à jour des logiciels applicatifs vers l'architecture réseau nouvelle génération.

Contents

1	Introduction	17
I	Context	21
2	Introduction of Key Satellite Concepts	23
2.1	Generalities	23
2.1.1	Satellites and their Missions	23
2.1.2	Platform & Payload	23
2.2	SAVOIR Reference Architecture, a Standardization at European Level	24
2.3	Legacy Network System Generic Architecture Overview	25
2.3.1	Devices Overview	26
2.3.2	Network Overview	27
3	Introduction to Networking Technologies	29
3.1	Reminder: the OSI Model	29
3.2	Network Design Paradigms	31
3.2.1	Definitions	31
3.2.2	Shared v.s. Point-to-Point Medium	32
3.2.3	Event-Triggered v.s. Time-Triggered Networks	33
3.3	Quality of Service	33
3.3.1	Definition	33
3.3.2	Performance Metrics and Properties	34
3.3.3	Fault Tolerance Metrics and Properties	35
3.3.4	Definition of Traffic Categories	36
3.4	MIL-STD-1553	36
3.5	SpaceWire & SpaceFibre	37
3.5.1	SpaceWire	37
3.5.2	SpaceFibre	38
3.6	The Ethernet Family	40
3.6.1	Ethernet	40
3.6.2	ARINC 664 (AFDX)	42
3.6.3	TTEthernet	43
3.6.4	Time Sensitive Networking	43

4	Focus on Time Sensitive Networking	45
4.1	Overview of Time-Sensitive Networking	45
4.1.1	General Overview	45
4.1.2	Reminder on 802.1 Switches and End-Station	47
4.1.3	Vocabulary	49
4.1.4	Configuration Challenge	51
4.2	Network Performance with 802.1Qbv- <i>Enhancements for Scheduled Traffic</i>	52
4.2.1	Introduction	52
4.2.2	Transmission Selection Algorithms	54
4.2.3	Transmission Gates	58
4.2.4	Configuration Problem for 802.1Qbv	61
4.2.5	802.1Qbv Architectures of Interest	61
4.2.6	Summary	69
 II Contribution: Selection of Candidate Technologies Compliant with the Quality of Service Requirements of a Next-Generation Satellite Network		 71
5	Problem Statement 1	73
5.1	Preselected Technologies	73
5.2	Satellite On-Board Applications Modelling	74
5.2.1	Devices	74
5.2.2	Motivating Example	75
5.2.3	Applications	75
5.3	Application Level Properties	77
5.3.1	Mixed Traffic Types	77
5.3.2	Time Management	78
5.3.3	Fault Tolerant Operations	78
5.4	Contribution Overview	78
6	Methodology: Qualitative Comparison Relying On Criteria Regrouped Per Properties	81
6.1	Related Works: Selection of a Next Generation On-Board Network	81
6.1.1	In the Space Domain	82
6.1.2	Outside the Space Domain	84
6.2	Definition of Our Criteria	86
6.2.1	Criteria for Application Level Property 1	86
6.2.2	Criteria for Application Level Property 2	87
6.2.3	Criteria for Application Level Property 3	87
7	Qualitative Comparison of the Pre-Selected Technologies with respect to the Identified Criteria	89
7.1	Technologies Capabilities w.r.t. space requirements	89
7.1.1	Ethernet	89
7.1.2	ARINC 664	93
7.1.3	TTEthernet	96
7.1.4	Time Sensitive Networking	98

7.1.5	SpaceFibre	101
7.2	Analysis	103
7.2.1	Summary of the Comparison	103
7.2.2	Third-party Arguments for the Selection of an Upgrade Candidate	103
III Contribution: Computation of TSN Networks Configurations for Next-Generation Satellite Networks		107
8	Problem Statement 2	109
8.1	Flow Modelling	109
8.1.1	Definitions	109
8.1.2	Motivating Example: Adding Flows	111
8.1.3	Restriction to Flow Level Requirements	111
8.2	Quality of Service Requirements	112
8.2.1	Reference Instants	112
8.2.2	Mixed Traffic Types Requirements	113
8.2.3	Fault Tolerance Requirement	114
8.2.4	Motivating Example: Adding Flow Constraints	114
8.3	Industrial Considerations	115
8.3.1	Production Contract and Release Instant	115
8.3.2	Application Emission Scheme	115
8.3.3	Cost of a Network Upgrade	116
8.4	Contribution Overview	116
9	Insights on the Configuration of IEEE 801.Qbv	119
9.1	Configuration Model	119
9.1.1	802.1Qbv Port Configuration	119
9.1.2	System Configuration	122
9.2	Related Works: Existing System Configurations for Low Jitter Requirements Support	122
9.2.1	Methodology: Configuration Generation with Constraint Programming	123
9.2.2	End-to-End TT configuration and its derivatives	123
9.3	Limitations of State of the Art Strategies w.r.t. Problem 2	128
9.3.1	Application Impact Issues	128
9.3.2	Scalability Limitations	129
9.3.3	Considerations on Upgrade Costs	129
9.4	Spacecraft Industry Use Cases	130
9.4.1	Airbus Generic Next-Generation Satellite Use Case	130
9.4.2	ORION Crew Exploration Vehicle Use Case	131
10	Reduction of the Application Impact and Computation Effort of State of the Art Configuration Generation with Egress TT	133
10.1	Egress TT, a Novel Approach for the Generation of Configurations Suited for Low Jitter Traffic	133
10.1.1	What is Egress TT?	133
10.1.2	How does it work ?	135
10.1.3	What makes Egress TT interesting?	135
10.1.4	Suitability of Egress TT	136

10.2	A First Implementation of Egress TT in IEEE 802.1Qbv Networks: Exclusive Queue Allocation	136
10.2.1	Adaptation of Egress TT Configurations to TSN Networks	136
10.2.2	Exclusive Queue Allocation Concept	138
10.2.3	Constraints Formalization for Exclusive Queue Allocation	138
10.2.4	Focus on the Computation of Release Instants	142
10.3	A Second Implementation of Egress TT with Improvement of the Schedulability of Exclusive Queue Allocation:Size Based Isolation	143
10.3.1	Size Based Isolation Concept	144
10.3.2	Constraints Formalization for Size Based Isolation	144
10.4	Limitations of Exclusive Queue Allocation, Size Based Isolation and Egress TT . . .	146
11	Experimentations: Performance Comparison of Egress TT and End-to-End TT Configurations	147
11.1	Software Architecture	147
11.1.1	Data Conversion	147
11.1.2	Configuration	148
11.1.3	Simulation	148
11.1.4	Validation	148
11.2	Latency and Scalability Comparison	149
11.2.1	Scalability	149
11.2.2	Network Latency	153
11.3	Experimentation on Larger Use Cases	155
11.3.1	Airbus Generic Next-Generation Satellite Use Case	155
11.3.2	ORION CEV Use Case	157
IV	Conclusions & Perspectives	161
12	Conclusions & Perspectives	163
V	Appendices	169
A	Introduction to Time Sensitive Networking	171
A.1	802.1Qbu & 802.3br -Frame Preemption	171
A.1.1	Introduction	171
A.1.2	Frame format for Frame Preemption Support	171
A.1.3	Frame Preemption How To	172
A.1.4	Preemption Limitation	174
A.1.5	Configuring Frame Preemption	174
A.1.6	Behaviour of Frame Preemption when Combined with Transmission Gates in Exclusive Gating	175
A.1.7	Summary	176
A.2	802.1Qci-Per Stream Policing and Filtering	176
A.2.1	Introduction	176
A.2.2	Concept of Streams	176
A.2.3	Stream Identification Function	177
A.2.4	QCi ingress port organization	178

A.2.5	Summary	182
A.3	Fault Tolerance with 802.1CB- <i>Frame Replication and Elimination for Reliability</i> . .	182
A.3.1	Introduction	182
A.3.2	Frame Format for Redundancy Support	183
A.3.3	Difference with ARINC 664 Redundancy	183
A.3.4	Summary	183
B	Use Cases: Flows & Flows Constraints Definitions	185
B.1	Airbus Generic Satellite Use Case	185
B.1.1	Flows	185
B.1.2	Flow Constraints	188
B.2	Orion CEV Use Case	189
B.2.1	Flows	189
B.2.2	Flow Constraints	198
C	A Step Further: Relaxing Hypotheses	205
C.1	Relaxing Hypothesis 2	205
D	Résumé Long Français	207

Chapter 1

Introduction

This PhD project was funded by Airbus Defence and Space and ARNT (*Association Nationale pour la Recherche Technologique*) and is included in Airbus TANIA-DP roadmap (*Technological Assessment for New Instruments and Avionics - Data Processing*), which, among others, aims at providing new communication standards for satellites on-board networks which would benefit future missions, be it on performance, on cost, on compatibility with ground segment standards or on system operation sides.

Context

In accordance with the ever-expanding volume of data generated and handled by ground-system equipments (telephone, cars, scientific instruments, etc.), satellites must be capable of *producing* and *transmitting* massive amounts of data in order to meet their users' requirements. While improving the performance of data production will not raise any major difficulties since instruments with such capabilities are already available on the market (e.g. COTS multi-gigabit camera, etc.); improving the performance of the on-board networks carrying that data remains complex. That is the reason why this document focuses on next-generation satellite embedded networks.

Satellite on-board networks serve two different purposes. First, they are in charge of conveying the necessary information for the nominal behaviour of a satellite (e.g. control of the thrusters, control of the communication sub-system, control of the solar panels, etc.). Then, they convey the data acquired from the instruments (such as telescopes, weather sensors, etc.) towards the antennas of the satellite which forward it to ground-stations. These networks are, most of the time, supported by two technologies: MIL-STD-1553 (1553 for short) and Spacewire.

Reason for a Change

While the current satellite architecture has demonstrated its strength and maturity for the past 15 years, it is starting to show its limits, be it on data-rate, on development and update costs, on availability of COTS components, or even on potential synergies with other industrial domains or academia. Therefore, the spacecraft industry is considering an upgrade of their on-board networks.

The next-generation network is foreseen as a "*unified*" network, meaning that a single technology would be used to support the needs of current and future missions on both satellite nominal behaviour management and instruments data-transfer. This technology should not only have better

performances than 1553 and SpaceWire but also be easy to analyse and configure as well as ease the development and integration process, and help reduce the overall cost of the satellite. Several technologies have been preselected by the industrial partner for this next-gen "unified" network: Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking and Spacefibre).

Among all these technologies, one opportunity has appeared at the beginning of the PhD: Time Sensitive Networking (TSN for short), the state of the art IEEE Ethernet technology, said to be capable of supporting both real-time and high-bandwidth traffic of a satellite. Therefore, the industrial proposed the subject of this PhD: "*Suitability of Time Sensitive Networking for the Spacecraft Industry*". This work is novel in the space domain since TSN had never been neither identified or considered as a potential candidate for a next-generation on-board network. This context was published in [18].

PhD Approach

In order to evaluate the suitability of Time Sensitive Networking with respect to the foreseen needs of next-generation satellites, we propose a two-step approach.

In a first step, we will aim at making sure that TSN is indeed a good candidate for the replacement of the current on-board networks by applying qualitative assessment. To do so, we will define a set of high level properties representative of the needs of future satellite missions. Inspired from existing state of the art comparisons, we will refine these properties into criteria. Finally, we will discuss the suitability of the set of technologies preselected by our industrial partner (i.e. Ethernet, ARINC 664, TTEthernet, Spacefibre and Time Sensitive Networking) with respect to these criteria and with these elements, compare these technologies with one another.

Then in a second step, we will go further into the assessment of the suitability of TSN by refining and formalizing the quality of service requirements of next-generation satellites. We will show that Time Sensitive Networking is suitable by generating one or several network configurations. If it is not possible to compute a configuration that fulfils these requirements, this would mean that TSN is not suitable for future missions. To do so, we will inspire from existing configuration approaches in the state of the art, based on frame schedules and constraint programming, to propose our own configuration approach. TSN is composed of numerous standards controlled by many parameters, which makes its configuration difficult. Therefore, in order to reduce this effort, we chose to reduce the set of TSN standards to IEEE 802.1Qbv only.

This two-step approach is illustrated in Fig. 1.1.

Organization of the Manuscript

The manuscript is organised as follows: in a first part, we give elements of context. The part has been written to provide the reader with the necessary background to understand the concepts and the vocabulary of the manuscript. In Chapter 2, we introduce key satellite concepts, i.e. what a satellite is, what its missions are, what the satellites networking technologies are, etc. Then, in Chapter 3, we propose an introduction to networking technologies. It starts from generic network concepts, and then focuses on the different technologies that will be mentioned or used in this manuscript. Finally, in Chapter 4, we give a focus on Time Sensitive Networking, the IEEE state of the art Ethernet technology, around which this PhD thesis was focused.

The manuscript is organized around two contributions presented in respectively Part II and Part III. Instead of a single related works chapter, we will propose a problem statement and an associated related work for each contribution.

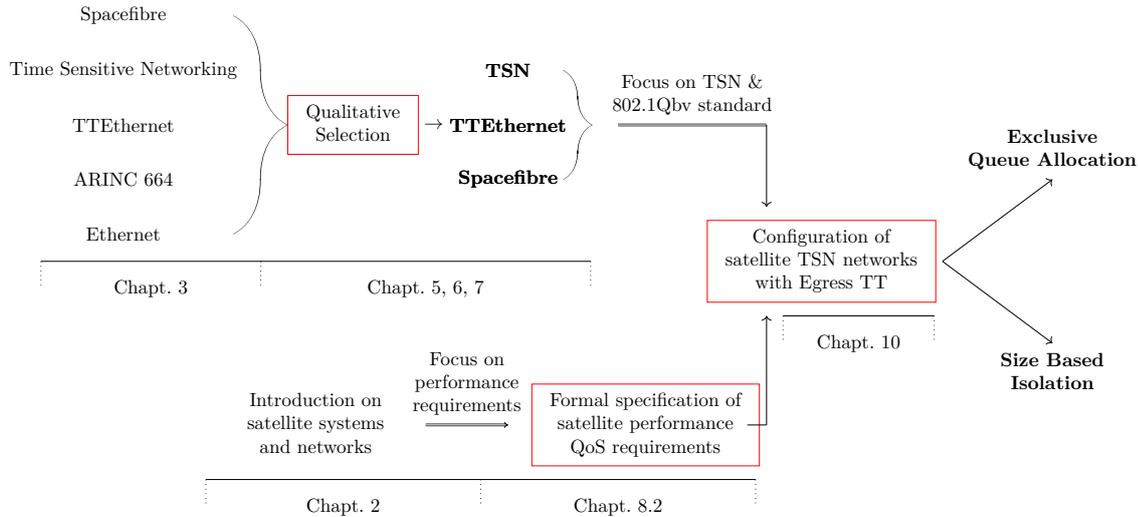


Figure 1.1: Technology selection process for a next-generation satellite on-board network

In Part II, we present the first contribution: *the qualitative selection of high-throughput technologies for next-generation satellite networks*. Chapter 5 presents a first problem statement, related to the identification of candidate technologies for a satellite network upgrade. We will introduce the list of pre-selected technologies as well as high level satellites requirements. In Chapter 6, we present our proposed methodology for the technology selection, supported by a related work. Finally, Chapter 7 presents the application of the methodology to the list of preselected technologies as well as an analysis of the results of the comparison.

In Part III, we present the second contribution: *the computation of TSN networks configurations for next-generation satellite networks*. In Chapter 8, we present the second problem statement, along with a more precise description of the satellite network quality of service requirements and industrial considerations. Chapter 9 proposes insights on the configuration of TSN standard IEEE 802.1Qbv. We describe the parameters of the configurations and the existing configuration strategies available in the state of the art. In Chapter 10, we propose a novel configuration approach for networks with very-low jitter requirements. We first describe the concept, interest and suitability of this novel approach and then apply it to the configuration of TSN networks. We will propose two strategies or *implementations* for the generation of these configurations. Finally, in Chapter 11, we evaluate the performance of the two implementations introduced in the previous chapter against an implementation of the state of the art configuration approach.

List of Publications

International Conferences

- [18] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. *TSN Support for Quality of Service in Space*. In 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020), Toulouse, France, January 2020. This paper introduces the interest of the spacecraft industry toward TSN as well as a very high level qualitative analysis of its suitability with satellite requirements.
- [19] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Comparative study of Ethernet technologies for next-generation satellite on-board networks. In Proc. of the 40th Int. Conference on Digital Avionics System Conference (DASC), 2021. This paper introduces the methodology and results of the qualitative comparison of Ethernet, ARINC 664, TTethernet and TSN for a next-generation on-board network.
- [Submitted] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. *Egress TT configurations for TSN networks*. In 30th International Conference on Real-Time Network and Systems Conference (RTNS), 2022. This paper presents the concept of Egress TT configurations and the two implementations for TSN networks that we detail in the second contribution in this manuscript.

Communications in an International Congresses

- [15] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. *Suitability of Time Sensitive Networking for Space*. In TSN/A Conference (TSN/A 2019), Bad-Homburg, Germany, October 2019. This presentation introduced the interest of the spacecraft industry towards TSN to TSN components and TSN-related software providers. It also proposed a glimpse on the first TSN profile for aerospace.
- [16] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. *TSN Support for Quality of Service in Space ?*. In Junior Researcher Workshop on Real-Time Computing (JWRTC 2019), Toulouse, France, November 2019. This poster illustrated the need for a next generation unified network and summarized the high level qualitative analysis of [18].
- [20] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. *Comparative study of high throughput technologies for next-generation satellite on-board networks*. In Data Systems In Aerospace (DASIA), 2021. This paper enhanced the paper [19] by adding Spacefibre to the set of compared technologies.

Reports

- [17] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Formal specification of satellite on-board networks requirements. Working paper or preprint, September 2020. This report proposed an open-access generic model of the satellite network and a formal expression of its requirements.

Part I
Context

Chapter 2

Introduction of Key Satellite Concepts

This first chapter starts with an introduction or a reminder on satellite generalities. It details the different types of satellites and the multiple missions they can be used for. It then introduces the *Platform & Payload* concept, a fundamental concept when designing satellites. The chapter continues with the presentation of the SAVOIR Reference Architecture, an ESA initiative for a generic satellite on-board functional architecture. Finally, this chapter zooms on the devices and networks on board a satellite that will be mentioned or used in the next chapters. It explains what the devices are and how they are linked together using a *platform* and a *payload* network.

2.1 Generalities

2.1.1 Satellites and their Missions

A satellite is a man-made "*electronic device that is sent into space and moves around the earth or another planet*" (Collins 2021). It is mainly used around Earth for telecommunications applications (e.g. telephone with *Inmarsat*, television with *Eutelsat*, internet with *OneWeb*), Earth observation applications (military, picture as a service with *Pleiades Neo*, ocean study with *Sentinel-6B*) and positioning applications. In the solar system, satellites are used for scientific applications (Sun study with *Solar Orbiter*, Mercury study with *BepiColombo*, Jupiter and its moons study with *Juice*). The format (i.e. size, weight), the orbit and the duration of the mission of the satellite vary: while satellites in geostationary orbit are large vehicles with at least fifteen years of operations (see Fig. 2.1a), low earth orbit satellites are smaller objects with shorter missions up to about five years (see Fig. 2.1b).

2.1.2 Platform & Payload

A satellite is generally composed of two parts : ***Payload*** and ***Platform*** (see Fig. 2.2). On the one hand, the *payload* (PL) is the purpose of the satellite, it is the part of the satellite that generates added value for its owner, it is specific to each and every mission. Generally, the payload is composed of instruments such as antennas or transponders for a telecommunication satellite, telescopes or cameras for an Earth observation satellite, or any form of scientific instruments for a scientific mission (radiation sensors, magneto-sensors, etc.).

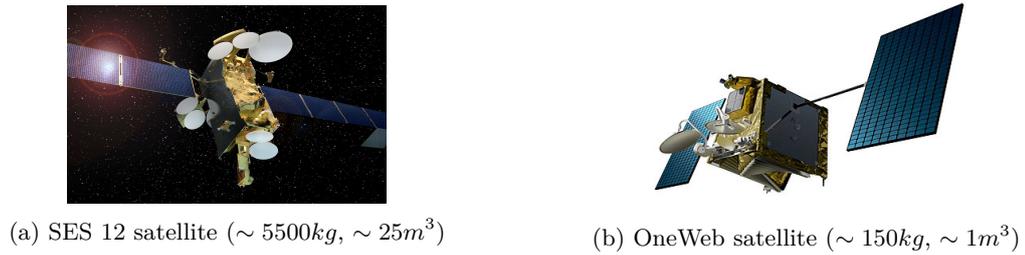
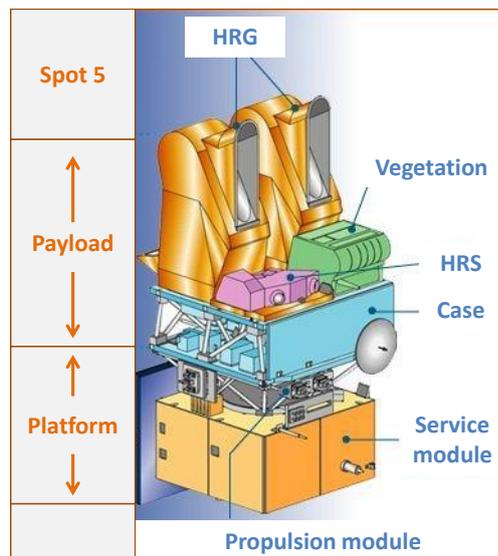


Figure 2.1: Satellite sizes

On the other hand, the *platform* (PF) is the infrastructure which allows the satellite to achieve its mission. If this part of the satellite does not work properly, the satellite might become useless or even be lost. The platform is composed of all the systems and subsystems that ensure a nominal behaviour of the satellite. Some notable systems are the Power Supply Subsystem, the Attitude and Orbit Control Subsystem (*AOCS*), the control and monitoring of the payload status, the management of telecommunications with ground stations system, etc.

Figure 2.2: *Platform* and *Payload* on Spot-5 Satellite

2.2 SAVOIR Reference Architecture, a Standardization at European Level

For a long time, agencies and space companies, at prime and supplier levels, have raised the need of increasing the level of reuse and standardization in spacecraft avionics systems in order to improve efficiency and reduce development time and costs. This has led to studies and initiatives which are

now federated in the Space Avionics Open Interface Architecture (SAVOIR) framework [37] through different working groups.

SAVOIR is, in a way, a reference to which any satellite manufacturer can rely on when designing a satellite. Plus, it does not contain any industrials' proprietary information and therefore is a good support for discussions and publications dealing with satellite architectures. SAVOIR vocabulary and reference architecture will be the terminology used in this document.

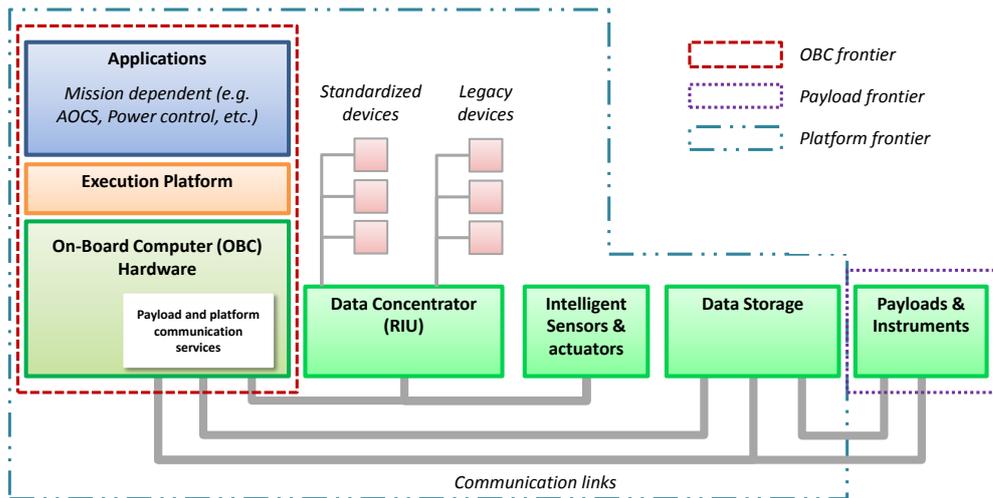


Figure 2.3: SAVOIR Reference Architecture

The SAVOIR framework defines the *SAVOIR Functional Reference Architecture* based on the needs of all kinds of *missions* (scientific, telecommunication, earth observation, etc.). The SAVOIR Functional Reference Architecture suggests a physical implementation for the electronic units of the platform; the On-Board Computer (OBC) architecture and the payload units. It aims at defining standard building blocks and their associated functions. It focuses on data management and communications means, but also considers ECSS (*European Cooperation for Space Standardization*) compliant interfaces interconnecting the blocks. In order to be relevant to a sizeable range of mission, generic specifications, considered as a common-core of avionic specifications have been gathered in several SAVOIR documents or *handbooks* such as, for instance, the FDIR¹ handbook. The Functional Reference Architecture is illustrated in Fig. 2.3.

This architecture is an important reference when designing a satellite on-board architecture. The network architectures as well as the performance and safety requirements (see Section 8.2) of this document will be compliant with SAVOIR.

2.3 Legacy Network System Generic Architecture Overview

This section presents a reference architecture for this research. It is compliant with the SAVOIR Functional Reference Architecture and is shown in Fig. 2.4. It contains several devices corresponding

¹Fault Detection Isolation and Recovery

to different functions interconnected with communication links. It is considered as generic enough and representative of future satellites.

In the next section, we describe the devices and, in the following one, the associated network.

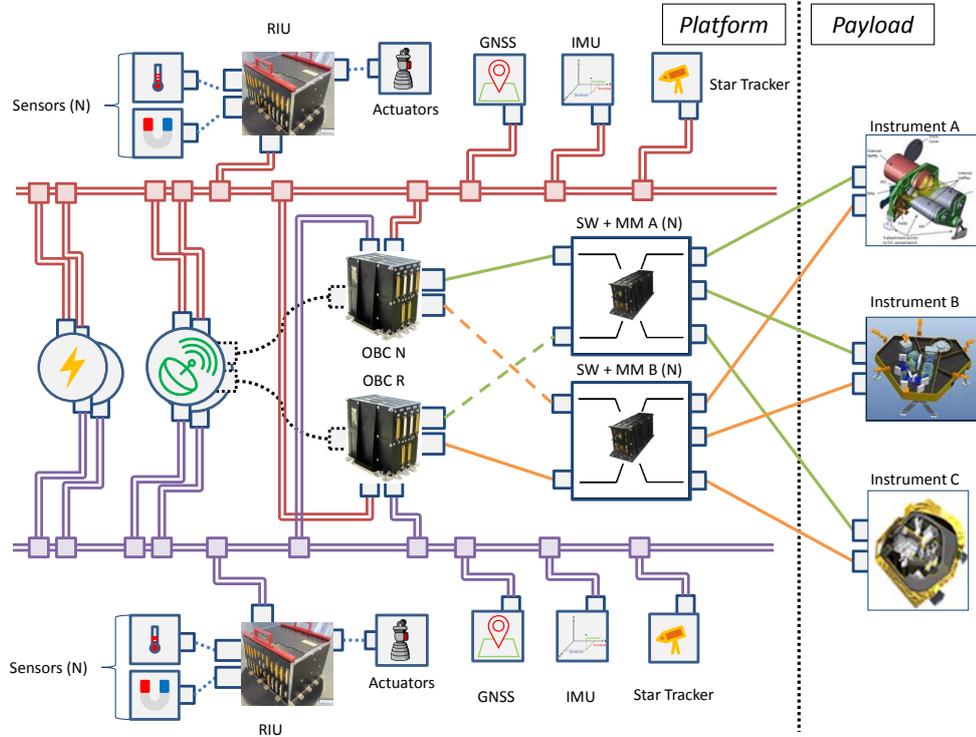


Figure 2.4: Generic Satellite Network System

2.3.1 Devices Overview

The most important device in the satellite architecture is the *On-Board Computer* or *OBC*. This OBC is the *master* of the satellite i.e. it manages all the other devices of the system.

On the platform part, it is connected to sensors (e.g. star trackers, thermal sensors, magnetometers, etc.) and actuators (e.g. thrusters, reaction wheels etc.), being eventually gathered in a *Remote Interface Unit - RIU*. The power control and distribution unit and the solar panel control unit are also connected to the OBC. The OBC is also hosting, among others, the *AOCS - Attitude and Orbit Control Subsystem* (also known as *GNC - Guidance and Navigation Control-*) functions. To do so, it gathers information from several sensors, processes and exploits them in order to control the propulsion system of the satellite.

On the payload part, the OBC is connected to a data storage system (Mass Memory - MM), usually a solid state mass memory, mainly used for storing payload data generated by the instruments.

Finally, the OBC is generally in charge of routing Telecommand (TC) and Telemetry (TM) received from the ground in the communication subsystem towards the instruments.

For a vast majority of missions, all the devices in the platform part of the satellite are duplicated. They work in *cold redundancy* meaning that only one device is active at a time (cf. Def. 16). If ever it ends up in a faulty state the other device is turned on while the fault is investigated by ground operators.

2.3.2 Network Overview

As for the satellite system, the on-board network is composed of two interconnected networks: *platform* and *payload* networks. Each of these networks fulfils diverging and sometimes contrasting needs.

On the one hand, the *platform network*, featured in red and purple in Fig. 2.4, is in charge of conveying all the necessary information used to guarantee the nominal behaviour of the satellite. It transmits data from sensors (position, magnetic field, temperature, etc.) as well as, among others, flight control commands. This kind of traffic, often described as *time critical traffic* requires bounded latency and low jitter communications. However, due to the small size and small volume of messages, a low data rate is enough to achieve the platform needs. In general, the platform network is implemented using a MIL-STD-1553 bus [33] or CAN [71] bus.

On the other hand, the *payload network*, featured in green and orange in Fig. 2.4, requires a very high data rate in order to convey the large amount of raw data generated by the payload instruments such as pictures from telescopes, telemeters from weather sensor or IoT (*Internet of Things*) data. The constraints are less stringent for a payload network: a delay in the packet communication path will most likely not impact the nominal behaviour of the satellite. The payload network is based in general on SpaceWire [42].

The communications links can be duplicated two or four times depending on the mission and are used in a cold redundancy scheme.

Conclusion

This chapter has proposed a snapshot view of what a satellite is, the *Platform & Payload* concept and the SAVOIR Reference Architecture. Then, it focused on the on-board architecture, centralized around the On-Board Computer as well as the on-board communication networks relying mostly on MIL-STD 1553 and SpaceWire.

Although these networks have served their purpose for the past fifteen years, they are starting to reach their limits, in particular in terms of available bandwidth. Therefore, they need to be upgraded. In the next chapter, the existing candidate technologies for this upgraded network are introduced.

Chapter 3

Introduction to Networking Technologies

This chapter introduces all the necessary building blocks to define and design a network system followed by a brief presentation of the technologies considered during the PhD. After a reminder on the OSI model, the concept of *network* is defined and the existing design paradigms for L2 networks are identified. Then, the different *Quality of Service* properties offered to a communication over a specific network are introduced. Finally, two legacy technologies i.e. *MIL-STD-1553* and *Space Wire*; and five candidates for the next generation satellite network, identified by the industrial partner at the beginning of the PhD, i.e. *SpaceFibre*, *Full Duplex Switched Ethernet*, *ARINC 664*, *TTEthernet* and *Time Sensitive Networking* are overviewed.

3.1 Reminder: the OSI Model

The OSI model or *Open System Interconnection* model [70] is a conceptual framework used to describe the functions of a networking system without any assumption on the technologies used to implement it. As suggested by *Guy Pujolles*: "*(translated) Even though this model is not used anymore, it still serves to define the [network] vocabulary and to be a reference for defining network functions*", we will use the OSI model to introduce some network-related vocabulary. Fig. 3.1 proposes a representation of this model.

The OSI model describes seven layers (denoted $L[\textit{number of layer}]$) from the physical layer (L1) up to the application layer (L7). Three objects are associated with each layer i.e. a *service*, a *protocol* and a *service access point*.

Definition 1 (Service of level N) *The service of level N describes the set of actions (including primitives, events, etc.) that is provided by layer N for the upper layer N+1.*

Definition 2 (Protocol of level N) *The protocol of level N describes the set of rules required to provide the service of level N and communication between two entities of level N through a N-PDU or Protocol Data Unit of level N.*

Definition 3 (Service Access Point of level N) *A service access point of level N, or N-SAP, is located at the edge between layer N+1 and layer N. Service of level N is provided to layer N+1 through a N-SAP.*

Let us describe briefly the physical medium layer and the first four layers of the OSI model.

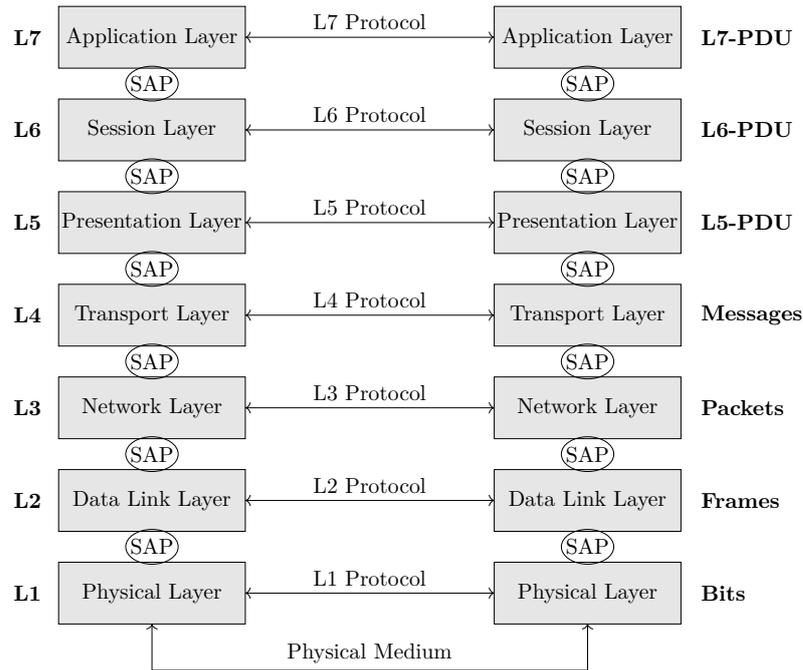


Figure 3.1: OSI Layer Model

OSI L0 - Physical Medium.

The *Physical Medium*, sometimes called Layer 0 (L0), is in charge of conveying a physical signal encoding some binary elements of informations from a point to another, towards their destination. Several physical media exist such as copper wires, fibre optics, radio signals, etc.

OSI L1 - Physical Layer.

The *Physical Layer*, or Layer 1 (L1), contains the set of rules and procedures required so that the binary information can be conveyed on the physical medium. For instance, a physical layer can define an encoding strategy e.g. Manchester [98] for emitting/receiving binary information on the physical medium.

OSI L2 - Data Link Layer.

The *Data Link Layer*, or Layer 2 (L2) handles the data links on the physical medium by grouping binary information into frames (see Def. 4). The frame is the entity in which a certain number of bytes are conveyed simultaneously over the physical medium. The Data Link Layer is composed of two sub-layers: Media Access Control (MAC) and Logical Link Control (LLC). The MAC sub-layer contains the set of rules necessary to share the same physical medium between several stations. An example of such rule is the CSMA/CD for *Carrier Sense Multiple Access, with Collision Detection* [58]. The Logical Link Control is in charge of increasing the reliability of the MAC sub-layer by adding flow control (optional) and error control (optional) services. Ethernet (see 3.6.1) is the most common Data Link Layer protocol. In Ethernet, the LLC sub-layer exists but neither its flow control nor error control services are used.

OSI L3 - Network Layer.

The *Network Layer*, or Layer 3 (L3) is in charge of conveying a *packet* (see Def. 5) from its source to its destination. The packet contains the address of the data recipient or information to guide the packet across the network towards its recipient. The L3 layer also provides flow control and error control services. The most notable L3 protocol is IP [66].

OSI L4 - Transport Layer.

The *Transport Layer*, or Layer 4 (L4) is in charge of the end-to-end transfer of user data gathered in *messages*. These messages are transported from the emitter to the receiver. The transport layer supports a communication between two users located in different systems, independently of the characteristics of the underlying network on which the communication occurs, in a seamless manner while guaranteeing the quality of service required by users. In *Internet*, TCP - *Transport Control Protocol* [67] and UDP - *User Datagram Protocol* [68] are the most common.

In the remaining of this document we will focus on technologies providing L2 protocols.

3.2 Network Design Paradigms

Two elements have to be decided when designing a L2 networking system: the choice of *physical topology* and the *communication paradigm*. Before detailing these two elements, let us define basic network vocabulary on which the elements' definitions can rely on. The definitions below have several external sources: "Les Réseaux" [98], OSRA-NET (ESA's Network Specification [38]), IEEE 802.1Q [56] and IEEE TSN standards [62]. Some of them also come from our own understanding of networking concepts.

3.2.1 Definitions

The terms *frame*, *packet*, *network nodes*, *end-point/end-system/end-station*, *switch*, *physical-topology* are defined hereafter.

Definition 4 (Frame) *In the ISO model, a frame is the support of communication of Layer 2 - Data Link Layer. In this document, unless if explicitly stated, the word frame will refer to Ethernet frames, compliant with IEEE 802.3 standard [58].*

Definition 5 (Packet) *In the ISO model, a packet is the support of communication of Layer 3 - Network Layer. In this document, unless if explicitly stated, the word packet will refer to IP packets, compliant with IETF RFC 791 standard [66].*

Definition 6 (Network nodes) *Component of a network physical topology. In our study, a network node can be understood as an end-station (see Def.7) or as a switch (see Def.8).*

Definition 7 (End-point, End-system, End-Station) *Component of network that needs to transmit or receive data. It is called end-point because this component will not forward any frames it receives to another network node (End-point or Switch) contrarily to a switch. It is also called End-System (ES) in the Airbus terminology and End-Station in the IEEE terminology. We will use any of these words indifferently in this document.*

Definition 8 (Switch) *Component of network that connects other network nodes together. It forwards frames that it receives from one of its input port to one out its output port that will lead the frames to their destination.*

Definition 9 (Physical Topology) *Physical network organization of all the network nodes of a system as well as the links connecting them together.*

3.2.2 Shared v.s. Point-to-Point Medium

The choice of a physical topology lies upon a dilemma: shall the medium be shared between all network nodes or shall it only connect two nodes together point-to-point? Depending on the answer to that question, the available topologies differ. We briefly describe them in the next paragraphs.

Shared Medium Topologies

In a shared medium topology, there is only one physical medium shared by all network nodes. When a message is emitted by an end-station, any network node is able to see that messages and to decide whether it should retrieve it (being recipient) or not. Examples of such topologies are *bus*, *ring*, etc. (see Fig. 3.2).

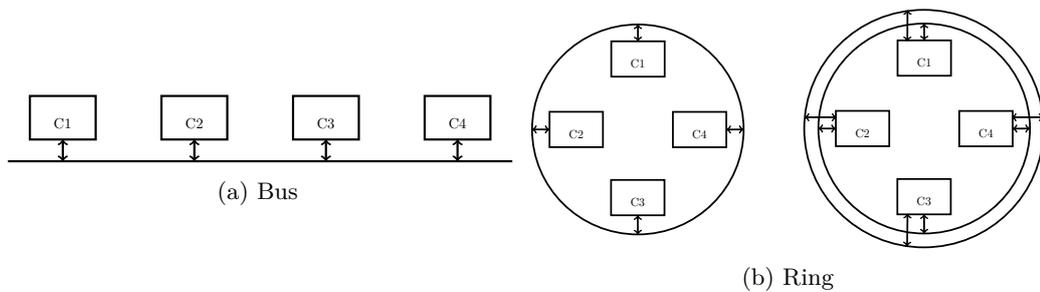


Figure 3.2: Shared Medium Topologies Examples

Point-to-point Medium Topologies

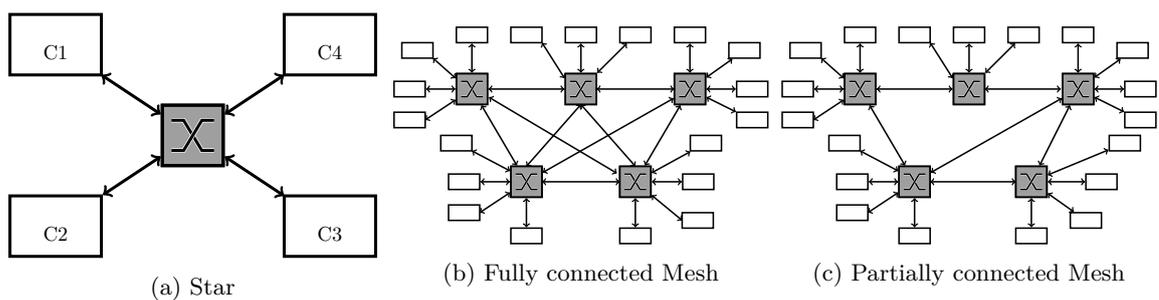


Figure 3.3: Point-to-Point Medium Topologies Examples

In point-to-point medium topologies, there are several physical mediums, each medium connecting only two network nodes to one another. In order for two end-stations in the network to communicate

with each other, if not already connected together, their frames must go through one or several intermediary network nodes e.g. switches. In these topologies, only the recipient end-station sees and receives frames destined to it. Example of such topologies are *star*, *mesh*, etc. (see. Fig. 3.3).

In this document, bus topologies (shared medium) and star or mesh topologies (point-to-point medium) will be encountered.

3.2.3 Event-Triggered v.s. Time-Triggered Networks

Once the physical topology of the network has been defined, the way communication happens on top of that physical topology has to be decided. It can either happen in an *Event-Triggered* or in a *Time-Triggered* fashion. We briefly describe these two approaches in the following paragraphs. An extensive comparison of Event Triggered v.s. Time Triggered systems is available in [73].

Event-Triggered

In an *Event-Triggered* network, the system reacts asynchronously to the occurrence of an event. Examples of such events are interrupts (e.g. when a frame is received, an interrupt can be used to signal it to the application level) or release of a shared resource (e.g. release of a mutex, end of DMA transfer, etc.). In event triggered networks, emissions of frames are asynchronous.

Time-Triggered

In a *Time-Triggered* network, frames are emitted in a synchronous manner i.e. they are sent at a known date for the whole network. The emission of every frame has been planned a priori off line. To do so, any emission is associated to a slot i.e. a time interval in which the frame is emitted. The slots scheduling repeats in a periodic manner. This type of network relies on a TDMA - *Time Division Multiple Access* [21] strategy. Time Triggered networks require the existence of clocks in a certain number of nodes in the network as well as a synchronization strategy for these clocks to properly apply the slots scheduling across the network.

Remark 1 *Some technologies have the ability to implement either the Event-Triggered, the Time-Triggered or both communication approaches.*

After the design choices have been made, a network system can be configured to provide *Quality of Service*. We define it in the next section.

3.3 Quality of Service

3.3.1 Definition

Let us now define the concept of quality of service and provide some examples of performance and fault tolerance metrics. The last paragraph defines the expression *Traffic Category*.

Definition 10 (Quality of Service (QoS)) *The Quality of Service for a data exchange (i.e. a group of frames transmitted from its source to its destination(s)) is a set of properties that can be guaranteed during the exchange. For instance, for a given data exchange, the Quality of Service can be to have a maximum jitter of 1 microsecond for all frames of the exchange. Quality of Service may equally refer to network performance metrics (see 3.3.2) or network fault tolerance metrics (see 3.3.3).*

3.3.2 Performance Metrics and Properties

The network performance metrics relate to the data exchange **temporal** behaviour. In this manuscript, let us consider 2 metrics: *Latency* and *Jitter*

Definition 11 (Latency, End-to-End Latency) *Duration necessary for a frame to go through a network, from emitter to receiver. It can be qualified of network or application end-to-end latency (E2E for short). The two definitions are illustrated in Fig. 3.4.*

1. **Network E2E Latency** *The considered duration is the time between which the frame is emitted on the medium and the time when the frame is received at physical layer in the receiving ES.*
2. **Application E2E Latency** *The considered duration is the time between which the frame is passed to the network stack of the emitting ES to the time when the network stack of the receiving ES makes it available for the receiving application.*

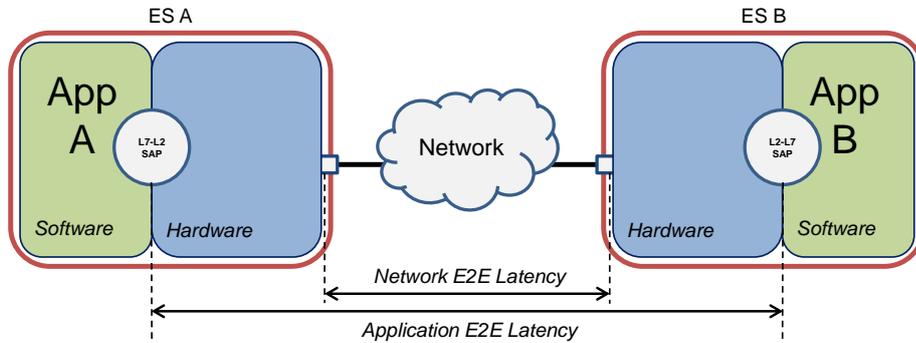


Figure 3.4: Application and Network E2E Latencies

Definition 12 (Jitter) *We distinguish two definitions:*

1. **Scheduling Jitter** *A Scheduling Jitter is the length, in time, of a time window around an expected time of action within which the action is actually performed. It represents the variation of the instant at which the action is performed.*
2. **Transmission/Transit Jitter** *In the networking community, jitter is understood as the variability of the network E2E latency.*

Using these two metrics, properties such as *bounded latency* (see Def. 13) or *low jitter* (see Def. 14) can be defined.

Remark 2 (Determinism, Bounded Latency, Low Jitter) *Among performance quality of service properties, determinism is often used. We find that term quite ambiguous even if one definition is proposed in [29]. Therefore, we propose to use more precise properties like bounded latency or low jitter instead of determinism.*

- In a warm redundancy scheme, the duplicated devices are idle but will not participate to network nominal activities until the nominal device is deemed faulty
- In a hot redundancy scheme, all duplicated devices are on and take part in the network nominal activities. An arbitration strategy will be needed to decide how to handle faulty devices.

Fig. 3.5 illustrates a cold, warm and hot link redundancy between two end-stations.

3.3.4 Definition of Traffic Categories

Let us introduce the concept of *Traffic Categories*.

Definition 18 (Traffic Category) *A traffic category is a classification of flows according to their QoS requirements such as bounded latency, low jitter, etc. For instance, Time Critical Traffic (see Def. 20) and Best Effort Traffic (see Def. 19) are traffic categories.*

Remark 3 *The list of traffic categories below is not exhaustive.*

Definition 19 (Best Effort Traffic) *A best effort traffic correspond to traffic which requires no quality of service whatsoever.*

Definition 20 (Critical Traffic) *A critical traffic is understood as a traffic category which requires network performance quality of service i.e. bounded low latency and low jitter as well as network fault tolerance quality of service i.e. tolerance to the loss of a frame (that can be achieved for instance with redundancy).*

Now that basic network vocabulary has been properly defined, let us use it to introduce the different networking technologies considered in this document. The presentation is organised as follows: a general introduction followed by a discussion on medium access. Then, the performance and the fault tolerance capabilities¹ of the technology are briefly discussed. The presentation finishes with a description of the links between these technologies and the spacecraft industry.

3.4 MIL-STD-1553

General Information

MIL-STD-1553B [33] is the specification of a databus originally developed for the US military in the 1970's. It is used in space as Platform bus for a majority of satellites since 1983. It is a single medium time-triggered technology.

Medium Access

The 1553 bus connects one bus controller (BC) to up to thirty-one remote terminals (RT). Communication on the bus relies on a command/response principle: the BC periodically sends, in a Time Triggered fashion, a command to one RT and the RT shall respond to that command. The bus controller can also broadcast a message to all remote terminals. No RT shall speak unless asked first by the BC. By doing so, the bus controller is in fact in charge of handling the access to the single medium.

¹Note that this chapter discuss what is provided by the technology itself, not what could be achieved with a clever use of that technology.

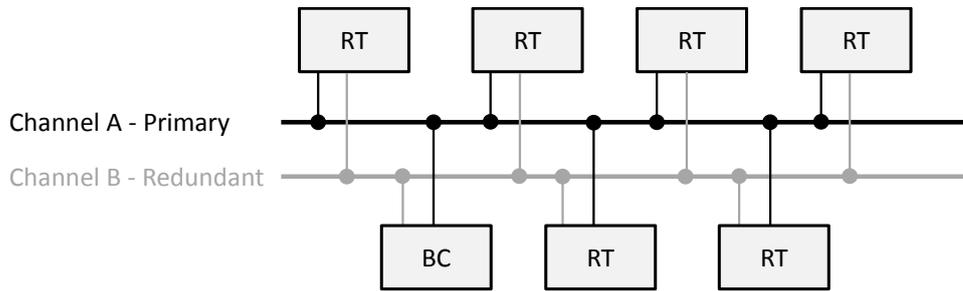


Figure 3.6: 1553 bus with one bus controller and six remote terminals

Performance

The 1553 data bus is a low data rate bus ($\sim 1\text{Mbit/s}$). The communication relies on messages composed of 1 to 32 "words" that can each bear up to two bytes of user data (in a data word). The communication pattern is simple: an exchange is initiated by a command word and acknowledged with a status word optionally accompanied by data words. The different communication patterns are standardized and can be found in [33].

Fault Tolerance

A 1553 bus is bi-directional and dual redundant. It is used in a cold redundancy scheme (see Def. 17) meaning that the communication happens on one "channel" (primary or nominal) and, in presence of a failure, messages are sent on the other (redundant) channel in the same bus. In space, 1553 is used either with single or dual bus.

Relation with the Spacecraft Industry

MIL-STD-1553B is widely used in the spacecraft industry because of its simplicity at both physical (OSI L1) and protocol (OSI L2) layers; and high bit error reliability. It is used on board in roughly 90% of Airbus satellites.

3.5 SpaceWire & SpaceFibre

3.5.1 SpaceWire

General Information

SpaceWire [42] is a point-to-point medium event triggered technology specifically designed for the spacecraft industry. Based on IEEE 1355, it was developed then standardized [42, 40] in the early 2000's under the impulsion of ESA and other major agencies (NASA, JAXA, RosCosmos). SpaceWire was first mostly used for point-to-point connexions but it is now being used as a networking technology i.e. connecting SpaceWire nodes using SpaceWire *Routing Switches*.

Medium Access

SpaceWire standard covers the first three layers of the OSI model: it relies on LVDS - *Low Voltage Differential Signalling* - used for data transmission in the double twisted pair cable at physical layer; the *character* i.e. the basic unit of a *packet*; and *wormhole* routing to exchange packets between *nodes* i.e. SpaceWire-capable equipments. The routing strategy is not further discussed in this document. It is explained in detail in [108].

Performance

The SpaceWire link is full-duplex and bi-directional. Its data rate ranges between 2Mbits/s and 200Mbits/s. Due to packet format, the theoretical number of bytes per transfer is unlimited. In space, it generally reaches 100Mbits/s with packets of 2 to 4 KiB. Higher protocols are also standardized to work with SpaceWire as for instance RMAP - *Remote Memory Access Protocol* [41], *CCSDS Packet Transfer Protocol* [39], etc.

Fault Tolerance

A SpaceWire character has a parity bit to detect error in the character as well as a link failure detection and signalization mechanism. The standard does not provide other fault tolerance capabilities.

Relation with the Spacecraft Industry

SpaceWire is widely used for connecting devices in the payload part of the satellite e.g. instruments (or instrument processing units) to mass memories. Its success is due to its robustness by design i.e. no memorization is required in the standard therefore avoiding the risk of SEU in buffers. In addition, its relatively high data-rate, the low-power consumption of its devices, its protocol simplicity and architectural flexibility make it ideal for many missions.

3.5.2 SpaceFibre

General Informations

SpaceFibre or *ECSS-E-ST-50-11C*[43] is a multi-gigabit networking technology designed for space applications. It runs over electrical or fibre-optic cables. It is a point-to-point medium time-triggered technology. It complements the capabilities of SpaceWire by improving the data rate, reducing the cable mass, providing quality of service in both performance and fault tolerance. As for SpaceWire, SpaceFibre covers the first three layers of the OSI model (L1 to L3).

Medium Access

SpaceFibre proposes an improvement of SpaceWire wormhole routing as medium access.

There are now up to 32 VC (*Virtual Channels*) per output port. These VCs work like FIFOs² the first frame to come in is the first one to go out. When several VCs have a data frame ready for emission, it is necessary to specify a medium access strategy.

SpF Scheduler

The *SpF Scheduler* offers the possibility of dividing time in 64 time-slots (of configurable fixed duration). In each time-slot, a list of VCs is allowed to try to access the medium. In Fig. 3.7, the *SpF Scheduler* is represented by the rectangle on the right. In addition, we materialized it on each VC,

² *First-In-First-Out*

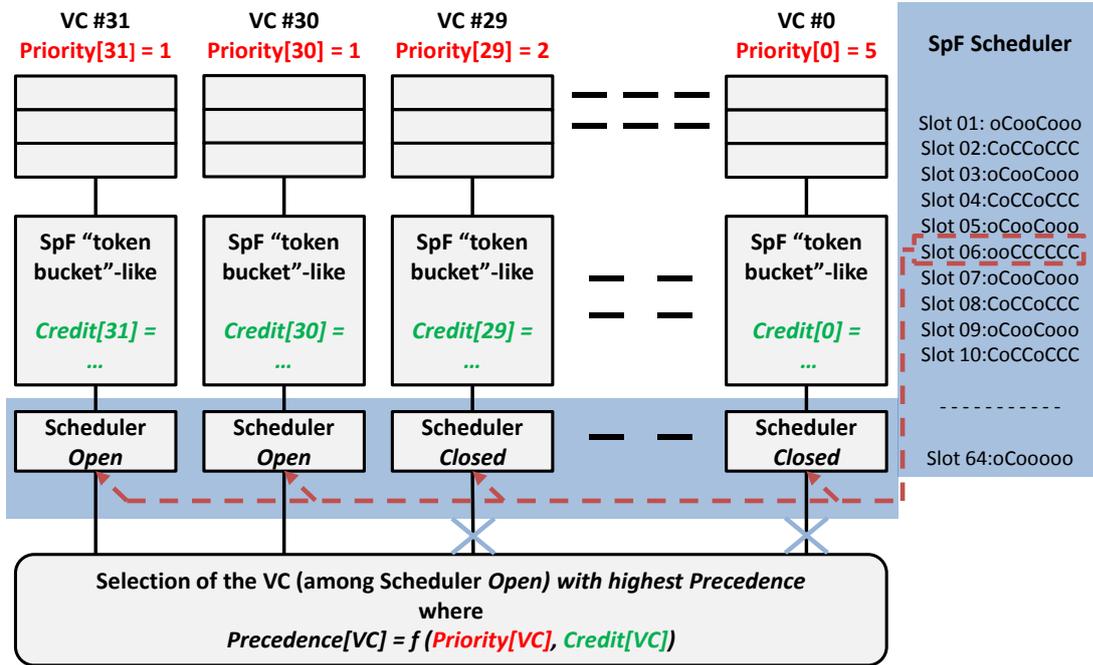


Figure 3.7: SpaceFibre Output Port Functional View

in a similar manner to TSN Gate Control Lists (cf. Fig. 7 in [18]). When the scheduler block on a VC states *Open*, it means that this VC is authorized to communicate in the time-slot, when the block states *Closed*, the VC is not configured. In the figure, VC #31 and #30 are configured in the represented time-slot.

VC Arbitration & Precedence

Then, in order to arbitrate between VCs authorized to communicate in the same time-slot, SpaceFibre defines *VC Arbitration* i.e. a rule similar to Ethernet *static priority*, based on a value entitled *precedence* per virtual channel at instant t . The precedence of a VC is computed with two values: *VC Priority* and *VC Bandwidth Credit*. The highest precedence VC will be granted access to the medium first. The *VC Arbitration* is represented by the block at the bottom of Fig. 3.7.

VC Priority

VC Priority is fixed a priori during configuration per VC. In Fig. 3.7, *VC Priority* is represented at the top, under the name of the VC. For instance, in this representation, VC #31 and #30 have the same and highest priority.

VC Bandwidth Credit

The value of the VC Bandwidth Credit evolves over time: its value is updated for every VC in a port each time a frame is emitted by one of the VC in that port. The bandwidth credit is represented, per VC, in the *SpF Token-bucket*-like block.

SpF Token-Bucket

The evolution of the *VC Bandwidth Credit* is dictated by a mechanism similar to a token-bucket (cf. [48]). This token-bucket like mechanism (let us call it *SpF Token-Bucket*) allows to reserve a portion of the available bandwidth for each VC. Examples of *VC Bandwidth Credit* computation are presented in [1].

Note that this section (including Fig. 3.7) only represents the user data channel, and ignores broadcast channels and management channels.

Performance

SpaceFibre links are bi-directional and full duplex. It can bear up to several gigabits per second. In fact, the protocols allows to rely on several *lanes* within the same link to drastically increase the data rate of a single link (in a similar fashion as PCIe³). A SpaceFibre data frame (L2) can bear up to 256 bytes of user data. The medium access rules allow SpaceFibre networks to provide bounded latency and controlled jitter.

Fault Tolerance

SpaceFibre provides fault tolerance capabilities at physical level. It is not detailed in this document. It also provides such capabilities at MAC level. In fact, error in a frame can be detected with a CRC. In addition, the non-respect of a bandwidth reservation contract for a VC can be monitored. SpaceFibre allows to isolate faults in a specific VC and prevent it from affecting other VCs. SpaceFibre does not provides redundancy mechanisms at MAC level.

Relation with the Spacecraft Industry

SpaceFibre development and standardization at ESA ECSS⁴ level is over. It is very likely to be used in future space missions as either Platform, Payload or Platform & Payload bus. In particular, it is explored by ESA in the ADHA - Advanced Data Handling Architecture - initiative.

3.6 The Ethernet Family

3.6.1 Ethernet

General Information

Full Duplex Switched Ethernet or *Ethernet* is an OSI L2 technology based on IEEE 802.3 [58] and 802.1Q [56]. It is a point-to-point medium event triggered network. In this document, we name *Ethernet* the technology defined in 802.1Q-2008. Ethernet is spread worldwide as it is the standard networking technology used at home and in ISP core networks.

³*Peripheral Component Interconnect Express*

⁴European Cooperation for Space Standardization (<https://ecss.nl/>)

Medium Access

In *Full-Duplex Switched Ethernet*, there can be no more than one device (end-stations or switch) connected to a port of a switch. Therefore there is no issue of medium access for end-stations. The medium access in switches output port is handled using *FIFO - First In First Out + Priority*.

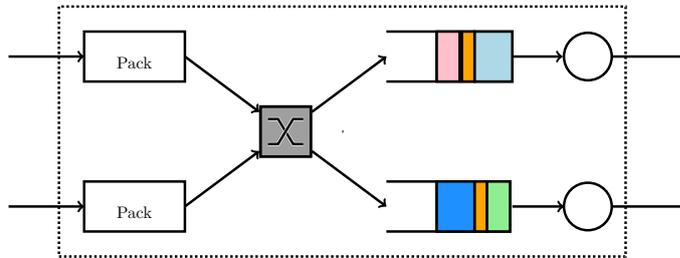


Figure 3.8: Ethernet switch

In the switches, frames heading towards the same direction through the same Ethernet output port are placed into queues, depending on their priority. The first rule applied is 'FIFO' meaning that, within a queue the first frame to get in is the first one to get out. In order to arbitrate between queues, the priority field of the 802.1Q optional tag is used. The frame having the highest priority is emitted first. Fig. 3.8 illustrates one Ethernet switch with 4 ports and one level of priority (i.e. one queue) per output port.

Performance

An Ethernet network is composed of switches and end-stations that exchange Ethernet frames (format defined in [58]) that can bear up to 1500 bytes of user data (and even more with *jumbo frames* [60]). An Ethernet frame is represented in Fig. 3.9. The Ethernet link is full-duplex and bi-directional and its data rate ranges from 100 Mbit/s to several gigabits per second.

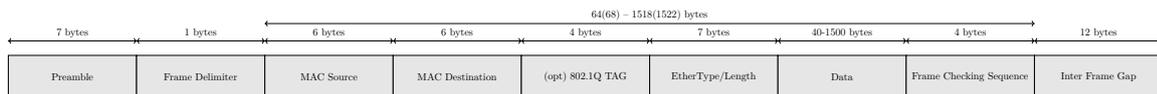


Figure 3.9: Ethernet Frame Format

Fault Tolerance

There is one mechanism defined in Ethernet for fault tolerance: the FCS -*Frame Checking Sequence*-field. This field contains a 32 bits CRC -*Cyclic Redundancy Check*- which serves to detect data corruption in a frame.

Relation with the Spacecraft Industry

Ethernet is just starting to be used on-board a spacecraft for either payload links or point-to-point dedicated links.

3.6.2 ARINC 664 (AFDX)

General Information

ARINC 664P7 [3] defines a 100Mbit/s avionic bus with a "deterministic"⁵ Ethernet protocol. It is a point-to-point medium event triggered network. It is used, among others, at Boeing and at Airbus under the name AFDX (Avionics Full Duplex Switched Ethernet). It extends Ethernet (802.1Q-2008) with bounded latency and fault tolerance capabilities.

Medium Access

The medium access in ARINC 664 is identical to Ethernet. In addition, the traffic contract defined hereafter is applied. The ARINC 664 standard defines VLs or *Virtual Links*. A VL is a traffic contract composed of a maximum frame size and a *BAG - Bandwidth Allocation Gap* i.e. a minimum time duration between two frames belonging to the same VL. The concept of BAG is illustrated in Fig. 7.2. There is one traffic type in an ARINC 664 network: VL traffic.

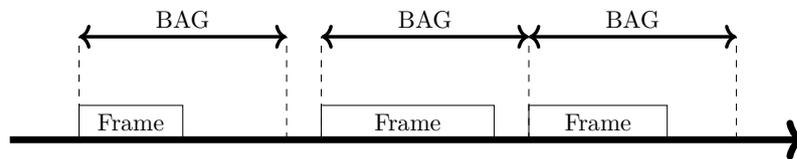


Figure 3.10: BAG concept in ARINC 664

Performance

ARINC 664 links are bi-directional and full duplex. They can operate at up to 100Mbits. The communication relies on Ethernet frames. With virtual links, an ARINC 664 network is able to provide bounded latency.

Fault Tolerance

Regarding fault tolerance, in addition to Ethernet CRC, ARINC 664 offers the possibility to duplicate frames in the emitting end-station and reassembling them in the receiving end-station in a hot redundancy fashion. In order to identify duplicates, it defines a *sequence number*: two duplicates will always share the same sequence number. A first function, entitled *Integrity Checking* eliminates invalid frames based on their sequence number. Integrity Checking is also used to detect the loss of a frame. A second function called *Redundancy Management* is in charge of the elimination of redundant frames (i.e. duplicates). In addition, the BAG protects the network against an application that would be emitting more than it is allowed or configured to.

Relation with the Spacecraft Industry

To the best of our knowledge, ARINC 664 is not used in the spacecraft industry.

⁵In ARINC 664 deterministic means bounded latency and no buffer overflow and respect of the traffic contracts.

3.6.3 TTEthernet

General Information

TTEthernet is a scalable networking technology designed for industrial automation and aerospace applications, standardized by SAE under the reference AS6802 [102] in 2011. It is a point-to-point time triggered and even triggered network. TTEthernet extends the ARINC 664 standard. It supports mixed quality of service with both synchronous (time-triggered) and asynchronous communications schemes, with the help of a fault-tolerant synchronization strategy.

Medium Access

In fact, in addition to *Best Effort* and *Rate Constrained* traffic (i.e. ARINC 664 VL traffic), TTEthernet standard defines a third type of traffic: *time-triggered* traffic or TT traffic. This traffic relies on a global schedule, computed a-priori, that defines the emission instant of all frames on all hops of the network. As explained in Section 3.2.3, this time trigger behaviour requires the use of a synchronization protocol.

Performance

TTEthernet links are bi-directional and full duplex. They can operate at up to 1Gbps. The communication relies on Ethernet frames. In terms of Quality of Service, the TT traffic class is able to provide bounded or even fixed latency and low to ultra-low jitter. The RC traffic class has the same performances than ARINC 664 traffic. The BE traffic class has the same performances as Ethernet.

Fault Tolerance

Regarding fault tolerance, in addition to Ethernet CRC and similar to ARINC 664, TTEthernet offers the possibility to do hot redundancy i.e. replicating frames in the emitting end-station, sent on disjoint paths (3 for TT traffic and 2 for RC traffic) and reassembled in the receiving end-station; in order to tolerate faults. As for ARINC 664, TTEthernet implements a *bus guardian* to make sure both the bandwidth reservation and the TT schedule are respected. Finally, TTEthernet provides fault tolerance capabilities for its synchronization mechanism (not detailed in this document).

Relation with the Spacecraft Industry

The use of TTEthernet is growing within the spacecraft industry. It is already used in some launchers and will be used in future lunar missions (e.g. Lunar Gateway [122]).

3.6.4 Time Sensitive Networking

Time Sensitive Networking is presented in the next chapter.

Conclusion

This chapter has presented key concepts of the OSI model. It has then proposed a set of definitions to agree on the vocabulary that will be used in this document. Finally, this chapter presented all the technologies, apart from Time Sensitive Networking, on which most of this study will rely on.

The next chapter will focus on introducing Time Sensitive Networking and explaining in detail one of its addenda dedicated to network performance quality of service i.e. IEEE 802.1Qbv.

Chapter 4

Focus on Time Sensitive Networking

This chapter introduces Time Sensitive Networking, the state of the art Ethernet technology from IEEE. It is composed of a large set of standards, over-viewed in Section 4.1, aiming at providing both real time and high throughput capabilities while ensuring a certain level of fault tolerance. Then this chapter proposes an in-detail presentation of TSN addenda IEEE 802.1Qbv *Enhancement for Scheduled Traffic*, entirely dedicated to network performance quality of service. It explains the different existing mechanisms and their parameters and conclude with the introduction of *Architectures of Interest* i.e. common configurations of previously presented TSN mechanisms found in the literature. Other main core TSN addenda are detailed in Appendix A.

4.1 Overview of Time-Sensitive Networking

Time Sensitive Networking (TSN for short) is a new IEEE technology that claims to be capable of supporting both real-time and high-throughput traffic. It is currently being developed by an IEEE task group, named TSN Task Group (TG). This Task Group has continued the work of the former AVB (for *Audio Video Bridging*) Task Group founded in 2012. The TSN Task Group has published more than a dozen of amendments as well as added a few new standards to IEEE 802.1 standard family in order to ensure a behaviour that is simultaneously real-time, adaptive and flexible, mixing synchronous and asynchronous approaches.

4.1.1 General Overview

Figure 4.1, originally presented by Janos Farkas, chairman of the TSN Task Group, during TSN\A Conference in Oct. 2019, and upgraded to this version in Nov. 2021, summarized the published standards and amendments as well as the on-going projects conducted by the TSN Task Group at the time. The list of standard is still growing at the time of writing this manuscript.

Definition 21 (IEEE Standard/Amendment/On-going Project) *For the record, an IEEE standard is a document that bears the knowledge on a process, a technology, or anything else, considered as a norm and common knowledge base to which anyone can refer. This document is declared a standard when it has been approved by an IEEE committee. While the document is being written/completed, once its theme has been approved by an IEEE committee, it is called an On-going*

project. A standard is generally named with a sequence of numbers (eventually separated with dots) followed by upper-case letters (ex:802.1Q standard). An amendment is a standard but it uses the reference of the amended standard plus one of several lower-case letters (ex:802.1Qbv amendment). This reference represents the classification of the document in all the IEEE standards available. An On-going project has the same naming convention, however, the sequence of numbers is prefixed with a P, for Project (ex:P802.1AS-Rev).

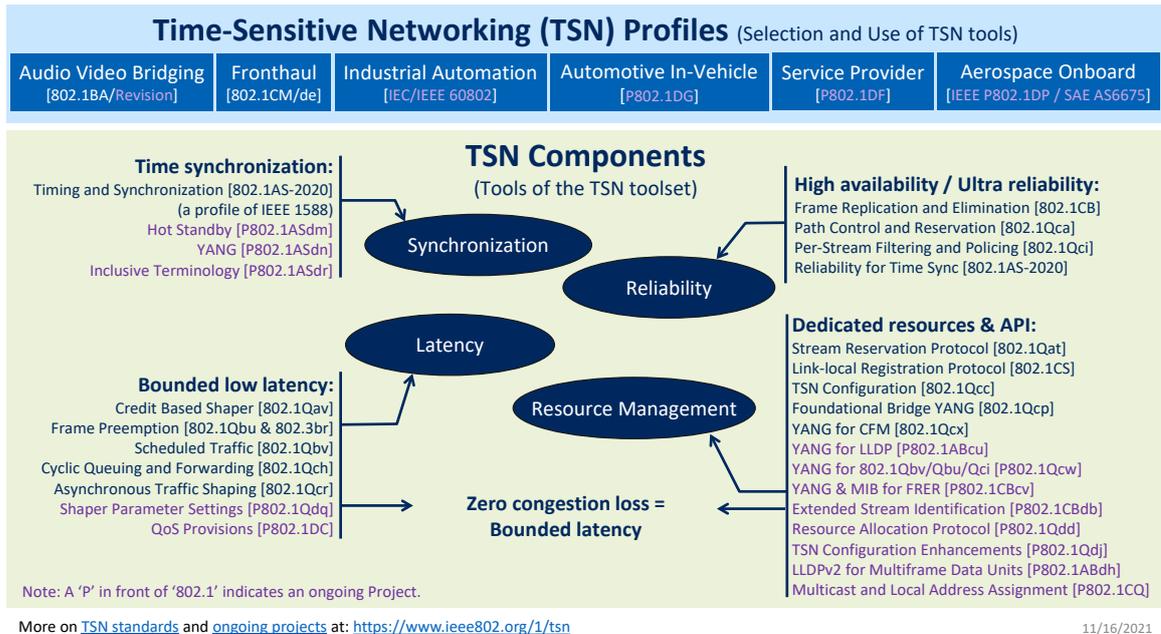


Figure 4.1: Status of TSN standards and projects in Nov. 2021

The TSN standards, amendments and on-going projects can be organized in five main families i.e. Synchronization, Reliability, Latency, Resource Management, Zero Congestion Loss.

Synchronization: The first family offers a network level *synchronization* service and it groups 802.1AS, 802.1AS-rev. 802.1AS defines synchronization and time distribution protocols for a TSN network. 802.1AS-rev defines upgrades to 802.1AS, mainly specifying a redundancy protocol for the synchronization service.

Reliability: The second family concerns *Reliability* (802.1CB, 802.1Qca, 802.1Qci, 802.1AS-Rev). It aims at preventing, as much as possible, the loss of a frame at application level by duplicating frames and/or by controlling that bandwidth reservation is respected by all streams. 802.1CB is a protocol used to support seamless network redundancy. 802.1Qca defines explicit path control and bandwidth and resource reservation protocol. 802.1Qci defines ingress policing strategies for TSN switches and end-points.

Bounded Low Latency: The third family is the *Bounded Low Latency* (802.1Qav, 802.1Qbu, 802.1Qbv, 802.1Qch, 802.1Qcr). It aims at providing protocols with a bounded end-to-end (network) latency for specific streams in the TSN network. This can be made possible through bandwidth reservation, traffic scheduling (synchronous and asynchronous) and preemption strategies. 802.1Qav defines traffic shaping strategies for TSN switches and end-points. 802.1Qbu defines a preemption protocol at ISO Layer 2 (Ethernet frame level). 802.1Qbv refines and upgrades 802.1Qav. 802.1Qch is a combination of other TSN protocols, aiming at building a TSN network with fixed latency and jitter. 802.1Qcr defines an asynchronous traffic shaping strategy for TSN switches and end-points.

Dedicated Resources and API: The fourth family is *Dedicated Resources and API* (802.1Qat, 802.1Qcc, 802.1Qcp). It defines resource management protocols as well as configuration strategies for a TSN network. 802.1Qat defines a resource reservation protocol for TSN. This can be done statically or dynamically. 802.1Qcc refines and upgrades 802.1Qat, it also defines a configuration protocol for TSN. 802.1Qcp defines a standardized model (YANG model) used to describe a TSN network, the capabilities of its devices, and potentially its configuration.

Zero Congestion Loss: The last family is *Zero Congestion Loss* (802.1Qav, 802.1Qbu, 802.1Qbv, 802.1Qch, 802.1Qcr, 802.1Qat, 802.1Qcc, 802.1Qcp). It aims at guaranteeing that no frames are lost due to congestion i.e. buffer overflow in switches. There are not much more details on this family as its protocols have already been introduced in other families.

A message to remember is that Time Sensitive Networking is not yet another new technology, it is based on the classic behaviour of Full Duplex Switched Ethernet (see 3.6.1) networks described in IEEE 802.1Q standard. It only adds protocols that amend or enhance the existing behaviours.

4.1.2 Reminder on 802.1 Switches and End-Station

In order to understand Time Sensitive Networking, we first remind/introduce the functional forwarding process of 802.1 switches and end-stations must be re-introduced.

802.1Q switch functional forwarding process An 802.1Q switch is a network device that forwards packets from input ports to output ports according to a certain number of rules. One can distinguish three steps in the forwarding process: *Ingress*, *Switching* and *Egress*.

The *Ingress* part of the process has the ability to filter frames and prepare the switching. It is located in input ports of the switch. The *Switching* part of the process actually switches the frames between input and output ports. It is located "between" input and output ports. The *Egress* part of the process has the ability to apply post-switching filtering and traffic shaping, it is located in output ports. Fig 4.2 summarizes this paragraph.

802.1Q end-station functional forwarding process This paragraph will introduce our understanding of the ES behaviour. In fact, up to this day, there is not real specification for TSN end-station in TSN, as IEEE Std 802.1Q-2018 focuses only on switches. There is currently a project for specifying the behaviour of End-Station within the TSN working group, but nothing has been published so far. Our understanding of TSN leads us to the following architecture for End-Stations in TSN (see Fig. 4.3). The dashed rectangles represents the part of our understanding which is unsure i.e the definition of the service access point above the data link layer.

Basically, in the output direction, it matches the egress part of the switch architecture whereas in the input direction, it matches the ingress part of the switch architecture. The interfaces (Service

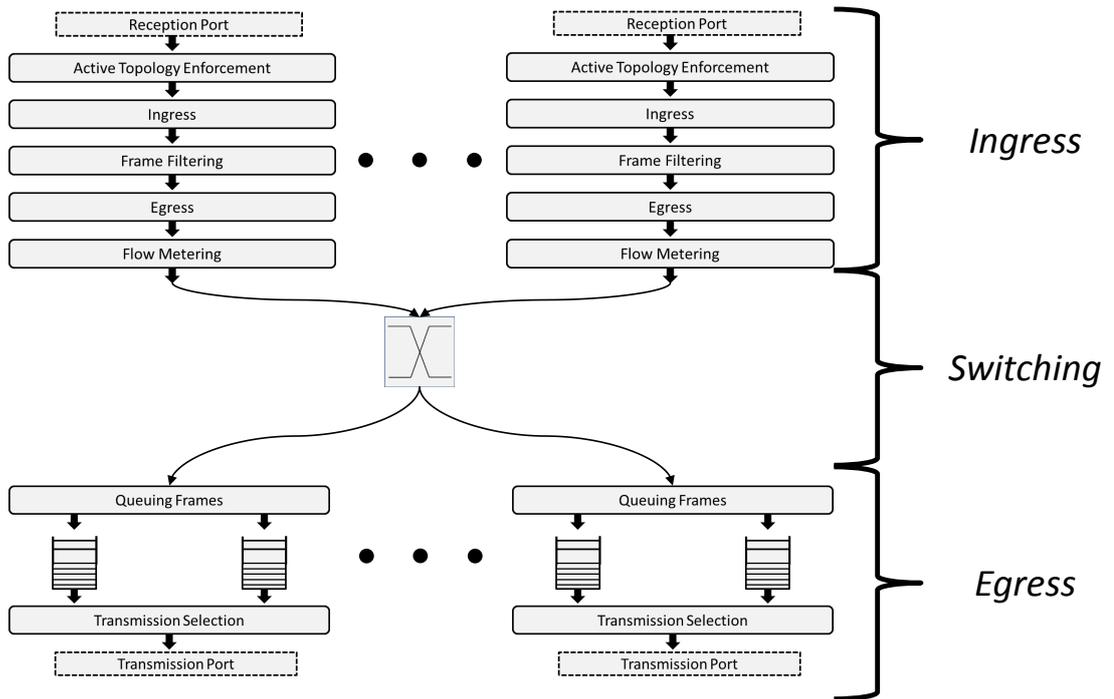


Figure 4.2: 802.1Q switch functional forwarding process overview

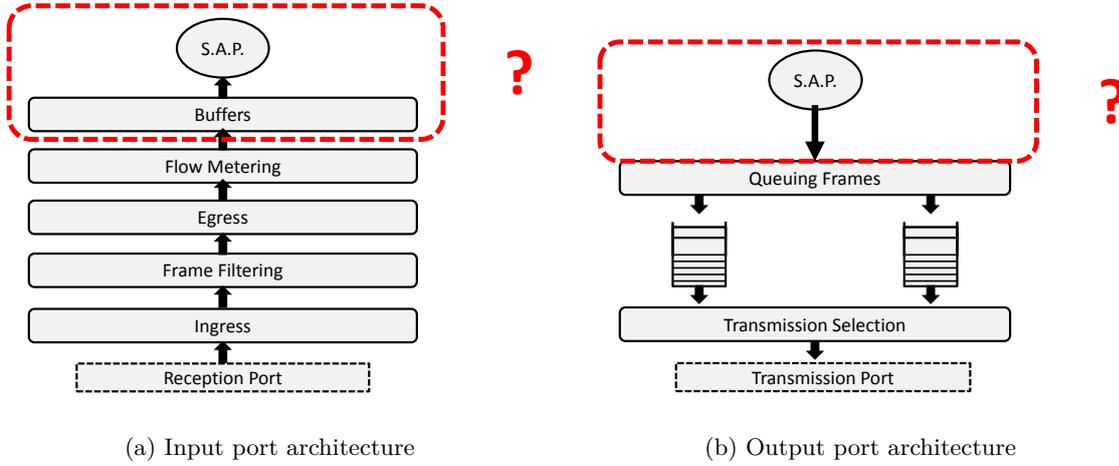


Figure 4.3: TSN End-Station Architecture

Access Points or SAP) between ISO Layer 2 and upper layers in a TSN End-System are still not specified in the standard. Nevertheless, we imagine, as represented in Fig. 4.3a, that some buffers will be necessary (close to the Service Access Point) in order to store frames while they wait to be processed by upper layers.

Local and system-wide TSN features As TSN relies on IEEE 802.1Q, the structure described above for the switches and end-stations is identical in TSN devices. There are also some added mechanisms through TSN features. The effects of all the TSN features that we have studied/considered can be categorized in two types of features :

- *Switch related features*, they have a direct impact on the switch forwarding architecture and can be configured differently from one switch to another
- *System wide features*, they have an impact on the whole network and their configuration is at system level more than per-device level.

The TSN features are sorted into these two types in Fig. 4.4.

In the previous paragraphs (regarding switches and end-stations), we only talked about *Switch related* features. We assume that the system wide features are available in both switches and end-stations. Their use, in the real system, will depend on the device capabilities and configuration.

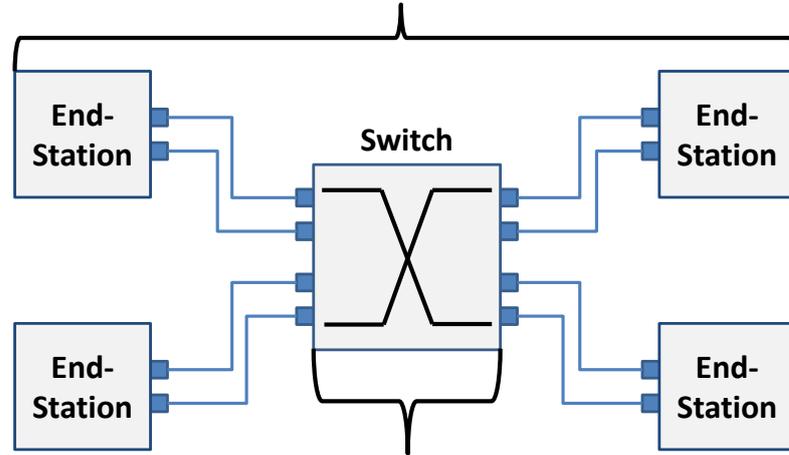
4.1.3 Vocabulary

The purpose of this section is to introduce the necessary vocabulary for the understanding of TSN standards. The most common definitions are grouped here however, some existing and new definitions will also be introduced along the document in dedicated sections. The definitions below have several external sources: IEEE 802.1Q [56] and IEEE TSN features documents [62].

Definition 22 (Feature) A feature refers to a TSN standard, amendment or on-going project. For instance, 802.1Qbv is a feature.

Definition 23 (Mechanism) A mechanism is a part of a feature. Thus a feature is a union of mechanisms.

System wide features: 802.1AS & 802.1AS-Rev (synchronization), 802.1 Qcc (TSN Configuration), 802.1Qat (Stream Reservation Protocol), 802.1CB (Frame Replication and Elimination for Reliability), YANG features (..).



Bridge related features: 802.1Q, 802.1Qbv (Enhancement for Scheduled Traffic), 802.1Qch (Cyclic Queueing and Forwarding), 802.1Qcr (Asynchronous Traffic Shaping), 802.1Qav (Credit Based Shaper), 802.1Qci (Per-Stream Policing and Filtering), 802.1Qbu (Frame Preemption).

Figure 4.4: Types of TSN features

Definition 24 (Parameter) *A parameter is a variable in a specific mechanism of a specific feature that can have several values. Thus a mechanism is a union of parameters. A mechanism has at least one parameter.*

Definition 25 (Configuration) *We consider three definitions:*

1. A **mechanism configuration** consists of a selection of parameters' values for that mechanism.
2. A **feature configuration** consists of a configuration of all the mechanism of that feature.
3. A **TSN network configuration** consists of a selection of a subset of TSN features and a configuration of these features.

Definition 26 (Streams) *A stream is multicast data transmission i.e. a unidirectional data transmission between one sender and one or several receivers. A stream is in fact a sequence of frame. It may also be called flow. In the Time Sensitive Networking and Audio Video Bridging terminology, the sender of a stream can be named Talker and the receivers Listeners.*

Definition 27 (Traffic Class) *A traffic class is a data structure introduced in TSN feature 802.1Qbv. There can be up to 8 traffic classes per output port of any 802.1Qbv compliant network device. These traffic classes have their own characteristics of the quality of service they can provide. Traffic types are associated with traffic classes at system/network design.*

Definition 28 (Architecture of Interest) *An architecture of interest is a way to configure a TSN network that is common case in the literature or in the industrial use. It is often named/described as a combination of Scheduling Types. TT-CBS-BE is an example of architectures of interest. Other architectures are introduced in 4.2.5.*

4.1.4 Configuration Challenge

As Time Sensitive Networking is composed of many features, and as each feature is composed of several mechanisms each including several parameters, there is a significant number of parameters to tune in order to configure a TSN network. It is getting more and more complicated and challenging whenever a new feature is added to the list of amendments. That is the reason why the TSN Task Group has started to promote a common and open way to describe configured TSN networks. This will help interconnect tools for configuring and analysing TSN networks.

The configuration description model is based on YANG (XML description of the network and its parameters) models and is being standardized into several TSN features (*IEEE 802.1Qcp, IEEE 802.1ABcu, IEEE P802.1Qcw, P802.1CBcv, etc.*). For now, only a few standards are included in the YANG models but the complete set of models is on its way. YANG is in fact a formal and common representation of the network and its configuration.

The configuration process itself is also standardized in TSN features IEEE 802.1Qcc - *TSN configuration*. The configuration process will not be described in this document.

IEEE Time Sensitive Networking is a wide technology, there are now more than 20 published amendments plus many on-going projects. Some industry verticals, such as automotive, industrial automation or 5G, have felt the need, for clarity and identity purposes, to reduce the number of standards in order to provide a smaller scope of TSN to the hardware/software manufacturers. These scopes are named *profiles* and also are standardized by IEEE. There are also often co-approved or co-written by other standardization institutions such as IEC or SAE.

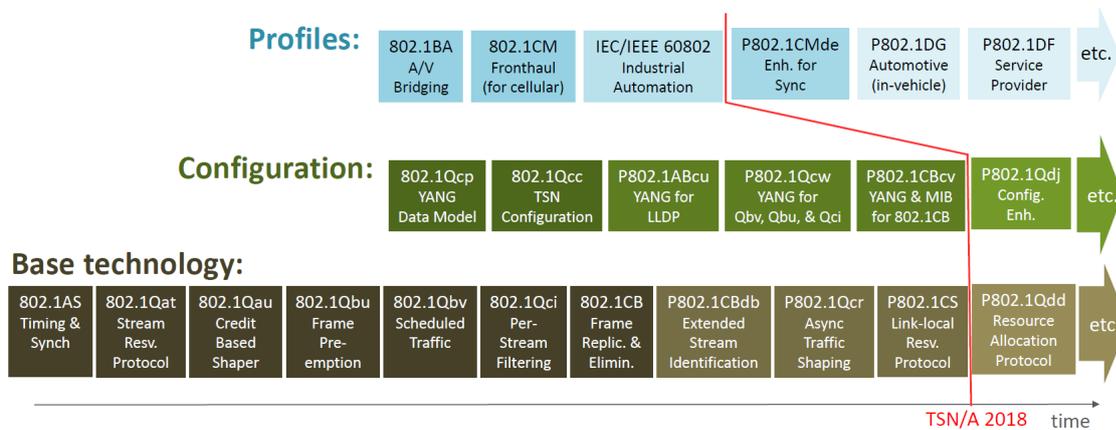


Figure 4.5: Status of TSN features, configuration and profiles

Figure 4.5, also from J. Farkas, gives a summary and status of the TSN base features, the TSN profiles (both introduced in 4.1.1) and the TSN configuration features in late 2019.

However, as good as the configuration features may be described, this does not help at all the network architect configure the TSN network. For now, the architect has to manually decide which TSN features to select and manually determine a value for the parameters of the mechanisms of these TSN features w.r.t. certain network performance and safety objectives. In this manuscript, we will propose a novel automatic configuration generation approach in Chapter 10.

4.2 Network Performance with 802.1Qbv-*Enhancements for Scheduled Traffic*

The TSN standard that needs to be introduced for this manuscript is IEEE 802.1Qbv - *Enhancement for Scheduled Traffic*. As the name suggests, this feature provides enhancements to the output queues of switches and end-systems in order to support *Scheduled Traffic* i.e. the scheduling of frames from different traffic categories so as to respect their Quality of Service requirements. In this section we will describe how the enhanced switch output port works.

Remark 4 *Although 802.1Qbv is now included in 802.1Q-2018, in this manuscript, we will continue using the acronym "802.1Qbv" to describe it so as not to confuse the reading: 802.1Q-2018 includes more than just 802.1Qbv features and therefore is not precise enough to talk about 802.1Qbv features alone.*

Fig. 4.6 is extracted from IEEE 802.1Q-2018. It represents the functional blocks associated to a switch. The part in the box is what is amended by TSN feature 802.1Qbv. Numbers in the pictures are the chapters and sections number associated to each box in the 802.1Q-2018 standard [56].

4.2.1 Introduction

In this enhanced output port, there are now up to 8 internal queues (i.e. traffic classes) to which frames can be assigned.

Parameter 1 *The mandatory number of queues really implemented in a device is not stated in the standard, let us called it $\#TC$, where $\#TC \in [1; 8]$ (maximum value in the standard). In this document, unless explicitly stated, the figures will be represented using $\#TC = 8$ queues.*

Each of these queues is associated with a Transmission Selection Algorithm (TSA) as well as a Transmission Gate (TG). Both TSA and TG will rule when frames within a specific queue will be allowed to try to access the medium. Fig. 4.7 summarizes the different mechanisms that one can see when zooming within the box (for $\#TC = 8$). These mechanisms are: the transmission queues, the transmission selection algorithm and the transmission gate blocks.

When frames arrive in the "queuing frames" part of the block architecture of the switch (Fig. 4.6), the frames are held in transmission queues. These transmission queues are the data structures that host the frames while they wait to be granted access to the medium. The queues are numbered and ordered from $\#7$ to $\#0$ and are often called traffic classes or just classes (cf. Def. 27). Each traffic class has a certain behaviour that will be defined with its transmission selection algorithm and transmission gate.

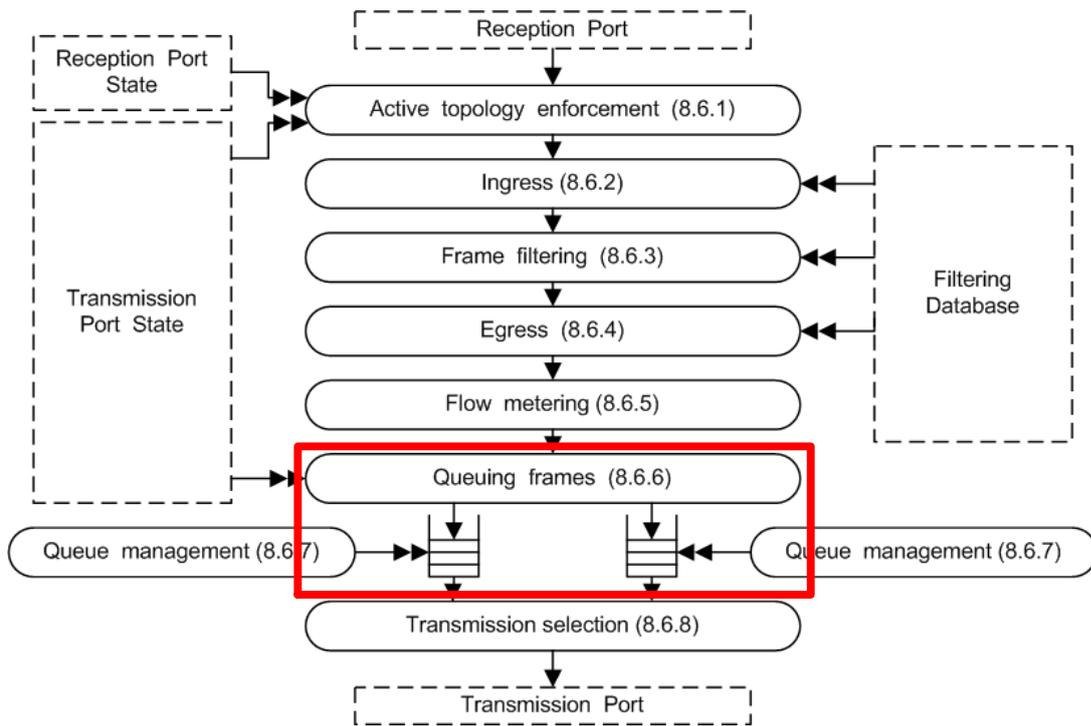


Figure 4.6: 802.1Qbv perimeter in a 802.1Q switch

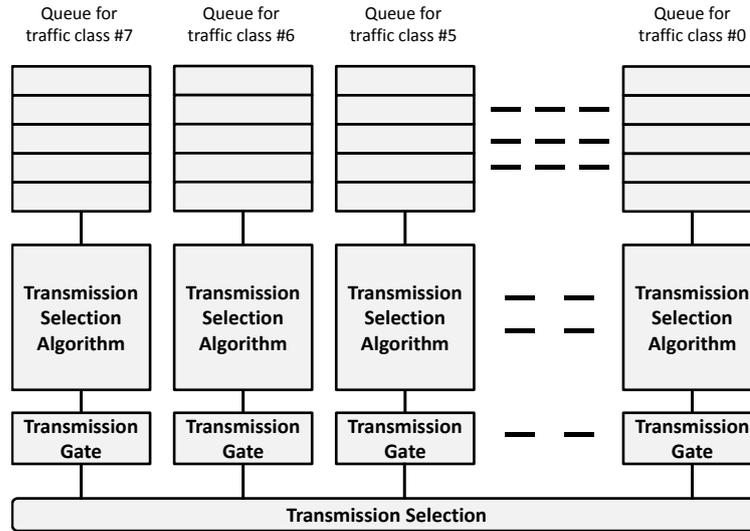


Figure 4.7: 802.1Qbv switch port

Definition 29 (Available for transmission) *In order for a frame to be allowed to try to access the medium, it must be "Available for transmission". This state requires the following conditions:*

1. *The frame shall be in head of a queue (i.e. traffic class),*
2. *The Transmission Selection Algorithm (of the queue) shall allow the transmission of that frame, marking it as "ready". By extension, the queue will also be considered as ready.*
3. *The Transmission Gates (of the queue) shall allow the transmission of the frame.*

As several queues exist in a single switch output port, several frames can be "available for transmission" in different traffic classes, hence an arbitration strategy must be defined. The transmission selection mechanism will select a frame among all frames "available for transmission" based on a static priority rule. The priority attribution implemented in TSN in the Transmission Selection block is the following: the higher the traffic class number, the higher the priority i.e. traffic class #7 has a higher priority than #0. In the end, this means that if a frame of # i and # j are allowed to access the medium at the same time with $i > j$, # i will be emitted first.

The allocation of frames (in reality streams) to traffic classes is discussed in Appendix A.2. Let us now focus on the Transmission Selection Algorithm and the Transmission Gate mechanisms.

4.2.2 Transmission Selection Algorithms

The transmission selection algorithm is one of the two mechanisms that defines whether a frame within a queue is allowed to try to access the medium. There are several transmission algorithms, or

shapers, introduced in the standard, namely : *Credit Based Shaper*, *Enhanced Transmission Selection* or even user-defined (ad-hoc). However the use of a TSA is not mandatory. Let us introduce these shapers in the following subsections.

No TSA

The use of a TSA is not mandatory in TSN. This means that one, several or all queues may choose not to use any transmission selection algorithm. In this situation, the frames in head of the different traffic classes would be immediately *ready*. The responsibility of selecting which frames goes out on the medium and in which order would be left to the transmission gates and the transmission selection.

Credit Based Shaper

The *Credit Based Shaper*, or CBS, is probably the most famous one. It defines a rule to share the bandwidth between queues based on a credit that evolves when frames are enqueued or dequeued. CBS was introduced in AVB in order to avoid starvation. Indeed, with no TSA, if the high priority queues have a lot of messages to emit, the lower priority queues will not get a chance to emit their frames. With CBS, the bandwidth shall be more equally shared between high and low priority queues.

As we will see later on in this document, the behaviour of CBS is slightly changed when combined with Transmissions Gates as well as with TSN feature 802.1bu-*Frame Preemption*. In the rest of this subsection, we will introduce CBS as it was defined in AVB.

A queue using *Credit Based Shaper* Transmission Selection Algorithm is characterized by two parameters and a counter:

- *send_slope*, which is set by configuration, represents the part of the bandwidth given to the traffic class that uses this CBS.
- *idle_slope*, is computed with *send_slope* and other variables.
- *credit*, counter.

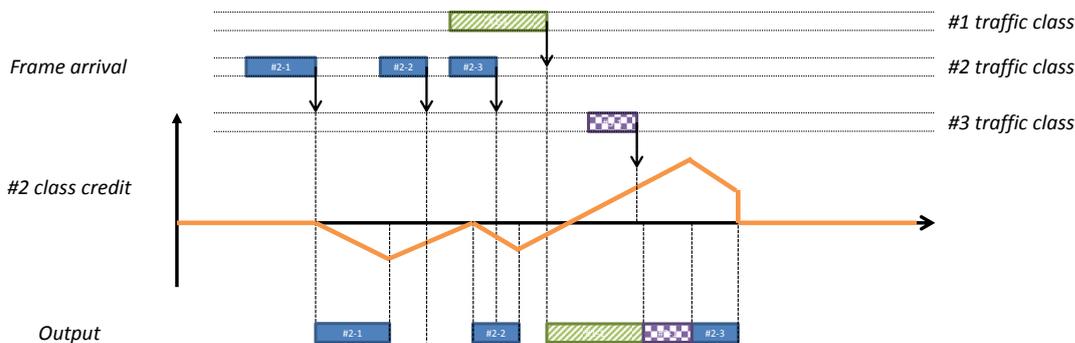


Figure 4.8: Example of CBS behaviour for traffic class #2

These parameters are used to determine, at a specific instant, whether the queue is allowed to emit or not. The credit evolves according to several rules.

1. When the queue is not empty and its *credit* is non-negative, then the message in head of the queue is selected for emission.
2. When a message is emitted, then the queue's *credit* decreases at *send_slope* rate.
3. When the queue is not empty but cannot emit (because someone else is using the medium), then its *credit* increases at *idle_slope* rate.
4. When the queue is empty and its *credit* is positive, then the *credit* is reset to zero.
5. When the queue is empty and its *credit* is negative, then the *credit* increases at *idle_slope* until it reaches zero, then stops increasing.

Example 1 *Fig. 4.8 represents the previous rules on a simple example with #TC = 3. In this example, we assume that traffic classes #1 and #3 do not have any Transmission Selection Algorithm. We also suppose that there are no Transmission Gate mechanism used so far. This means that when a frame is enqueued it has immediately the right to try to access the medium. Traffic class #2 uses a Credit Based Shaper.*

*As one can see on the figure, when the first frame of #2 is enqueued, #2 was empty hence its credit was zero. When the first frame arrives, as the credit is equal to zero (considered positive) and no one is using the medium, then this frame is emitted and the credit is decreased at *send_slope*.*

*When the frame is emitted, a new frame is enqueued in #2. However, its credit is negative, the message cannot be emitted, it has to wait for the credit to increase (at *idle_slope*) and become positive to be emitted. Again, once the frame is emitted the credit decreases at *send_slope*.*

While the credit was negative, a frame of #2 and one of #3 have arrived. As the credit of #2 is negative, the #3 frame is emitted, even if frames of #3 have a lower priority than #2. As there is a frame enqueued for #2 and it cannot be emitted (medium occupied by #3 frame), the credit increases.

When the medium is freed (#3 frame is gone), the #2 frame could be granted access to the medium, however, a frame from a higher priority traffic class (#1) has arrived in the meantime and is first granted access to the medium. The credit hence continues to increase. When the medium is freed, the #2 frame is granted access to the medium and the credit decreases.

Once the frame is emitted, as the queue is empty and the credit is strictly positive, it is reset to zero.

Enhanced Transmission Selection

The Enhanced Transmission Selection Algorithm is also introduced in 802.1Qbv. Its goal is to ensure a fair sharing of the available bandwidth to the traffic classes that implements it. There are no implementation proposed in the standard but still, one is suggested : Weighted Round Robin (WRR).

Each queue implementing WRR TSA is affected with a *weight*. This weight will determine how much the queue will be served w.r.t. the other queues. Several implementations for weighted round robin exist, we will introduce one. The WRR server follows the following rules (as described by Algorithm 1, c.f. [12]):

1. When a queue is served, if its weight is positive, a frame of this queue can be emitted and its weight is reduced by one.
2. As long as the weight of the queue is positive and the queue contains frames to emit, then the frames are emitted and the weight is decreased by one for each.

Algorithm 1: WRR algorithm

```

Input: Number of queues, #TC ∈ [1; 8]
Input: Queue weights, w1, ..., w#TC ∈ ℕ
1 while True do
2   for i = 0 to #TC - 1 do
3     currentWeight ← wi ;
4     while (not empty(#i)) and currentWeight > 0 do
5       MarkAsReady(head(#i)) ;
6       currentWeight ← currentWeight - 1 ;

```

3. If the weight reaches zero, then the next queue (in the list of queues implementing the algorithm) is served (with the same rule w.r.t. the queue's weight).
4. If a queue has a weight positive but no frames to emit, then the next queue is served.
5. When the queue has been served, its weight is reset to its configured weight. The remaining weight that may exist, because the queue did not have enough frames to emit during the previous round, is lost.

Example 2 Fig 4.9 instantiates these rules on a simple example with 3 traffic classes.

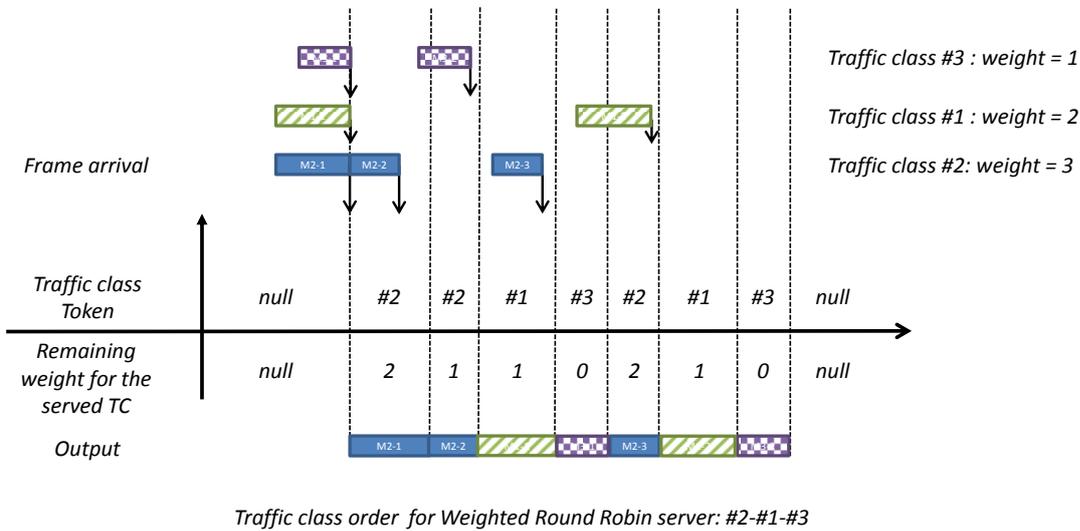


Figure 4.9: Example of WRR behaviour on #1-#2-#3 traffic classes

Vendor Specific

We have introduced in the previous subsection common TSA. But that standard also leaves the door open for vendor specific transmission selection algorithm. The only constraint that all transmission selection algorithms must respect is named *Ordering requirements* in the standard. All TSA must respect the ordering requirements of 802.1Q i.e. the order of frames received on the same Bridge Port shall be preserved for:

- Unicast frames with a given VLAN identifier (VID), frame priority, stream identifier, MAC destination address and MAC source address combination
- Multicast frames with a given VLAN identifier, frame priority, stream identifier, and MAC destination address.

These elements (VID, frame priority, stream identifier, MAC destination and source address) will be introduced later.

4.2.3 Transmission Gates

The Transmission Gate mechanism is the second element that comes into play for deciding when frames in traffic classes are allowed to try to access the medium i.e. *available for transmission*. It is also often called *Time Aware Shaper - TAS*. Each of the #TC TSN queues is associated with a transmission gate. In fact, this transmission gate may prevent a frame from a specific traffic class to access the medium even if its transmission selection algorithm allows it. Let us explain how these transmission gates work. A transmission gate can have two states : Open (o) or Closed (C). When the gate of traffic class #a is:

- Closed → Even if the TSA allows it, #a frames cannot try to access the medium,
- Open → If the queue is *ready* w.r.t. to its TSA, #a frame can try to access the medium i.e. is declared "*available for transmission*"

It appears clearly that the gate state must evolve over time, otherwise, if a gate is closed and stays closed, it would mean that the frames of the associated traffic class will never be emitted and would end up being lost. To this extend, a Qbv switch output port has a data structure called a *Gate Control List*.

Definition 30 (Gate Control List) *The purpose of the Gate Control List or GCL is to provide the state of gates of each traffic class of a specific transmission port of a network device, composed of #TC queues. More precisely, we propose to describe a GCL as follows:*

$$GCL_{port} = \langle Tick, TimeInterval, OperControlListLength, CycleTime, Schedule \rangle \quad (4.1)$$

Where :

$$\left\{ \begin{array}{ll} Tick : & \text{base clock ,} \\ TimeInterval & \in \mathbb{N}^+, \\ OperControlListLength & \in \mathbb{N}^+, \\ CycleTime & \in \mathbb{N}^+, \\ \text{with} & OperControlListLength * TimeInterval \leq CycleTime, \\ Schedule : & V = \langle v_e \rangle, e \in [1, OperControlListLength], \\ \text{with} & v_e = \langle l_k^e \rangle, k \in [1, \#TC], \text{ and } l_k \in \{o, C\} \end{array} \right. \quad (4.2)$$

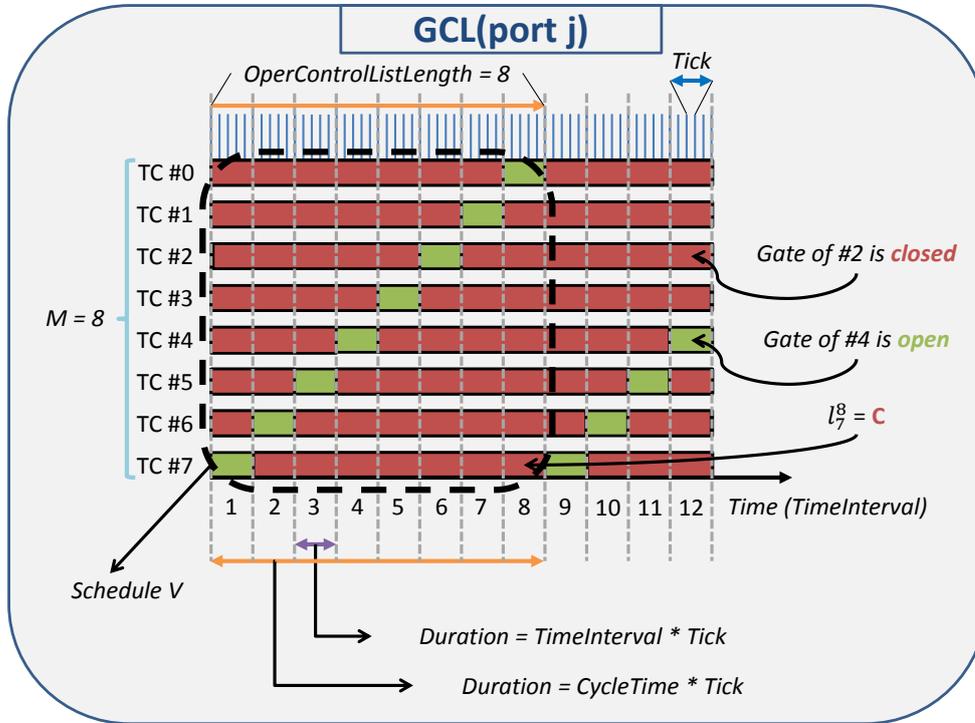


Figure 4.10: 802.1Qbv Gate Control List formalization

Example 3 Figure 4.10, provides a visualization of these formal parameters on a simple example of a port labelled j with $\#TC = 8$ queues and $OperControlListLength = 8$ states. The behaviour of all traffic classes is based on the same sequence of closing and opening: the gate is closed during a duration of $7 * TimeInterval$ and open during one $TimeInterval$. Without loss of generality, let us focus on v_0 , first element of the schedule. The other elements of the schedule can immediately be obtained by circular permutation of v_0 . In v_0 , the transmission gate of #7 is open in the first slot (of duration $TimeInterval * Tick$) of the schedule and the transmission gates of all the other queues are closed hence:

$$v_0 = \langle l_k^0 \rangle, k \in [1, 8] = [o, C, C, C, C, C, C, C] \quad (4.3)$$

The expression "a port uses the transmission gate mechanism" will be understood as "this port has scheduled closing and opening sequence for its transmission gates". The expression "a port does not use the transmission gate mechanism (or does not use Time Aware Shaper)" will be understood as "the transmission gates of this port are always open". In other words, the default behaviour of the Time Aware Shaper for port j is to have all its gates open all the time i.e.:

$$\begin{aligned}
& \text{for } e \in [1, \text{OperControlListLength}], \\
& \text{for } k \in [1, \#TC], \\
& l_k^e = o,
\end{aligned} \tag{4.4}$$

Although 4.3 proposes a schedule with more than one element ($e > 1$), when for each schedule element, all the transmission gates are open, *OperControlListLength* could be set to 1 when the port does not use the transmission gate mechanism.

Configuring the Time Aware Shaper on a transmission port is equivalent to finding a schedule and its duration or number of states for the transmission gates of all queues of this port i.e.

$$\left\{ \begin{array}{l}
\text{TimeInterval} = ?, \\
\text{OperControlListLength} = ?, \\
\text{CycleTime} = ?, \\
\forall e \in [1, \text{OperControlListLength}], v_e = ?
\end{array} \right. \tag{4.5}$$

Remark 5 (TimeInterval) *Contrarily to the above explanation, we recently discovered that in the standard, TimeInterval is not constant but variable for each gate event. While it does not change the general principle of 802.1Qbv gate control list that we have just explained, we felt it was important to correct our understanding with this remark.*

Remark 6 (Tick implementation interoperability) *Although it seems rather clever to have such abstraction using a Tick for defining the length, in time, of the events in the GCL. It might create some limitations in the choice of devices or interoperability between manufacturers limitation if the minimum supported value of Tick differs from one manufacturer to another. There is not standardized minimum value of Tick in 802.1Qbv, a value is suggested (1 nanosecond) but it is not mandatory to support it. The network architect shall be aware of this issue when selecting its switches and end-stations.*

Remark 7 (Impact of TAS on CBS) *The use of the Transmission Gate mechanism has an impact on the rules defined for credit based shaper. In fact, the standard defines rules for CBS regarding the evolution of the credit when the transmission gate is closed or what would happen to it when the gate is open but is about to close. The analysis and modelling in network calculus of this CBS behaviour when combined with TAS was done in [28]. The new credit evolution rules for CBS with TAS are the following (cf. [30] for the in-detail analysis):*

1. *When the queue is not empty, its credit is non-negative, its gate is open and there is enough time before the next gate closing event for transmitting the frame in head of the queue, then the queue is considered ready, and the message in head is selected for emission.*
2. *When a message is emitted, then the queue's credit decreases at send_slope rate. This does not apply to the overhead induced by frame preemption (see Appendix A.1).*
3. *When the queue is not empty but cannot emit (because someone else is using the medium), or emit and overhead due to preemption, then its credit increases at idle_slope rate.*
4. *When the queue is empty and its credit is positive, then the queue is reset to zero,*
5. *When the queue is empty and its credit is negative, then if its transmission gate is open, then its credit increases at idle_slope rate until it reaches zero, then stops increasing.*

- 6. When the transmission gate of a queue using CBS is closed, then this queue cannot emit frames and its credit remains constant.

Fig. 4.11, illustrates these rules on a simple example.

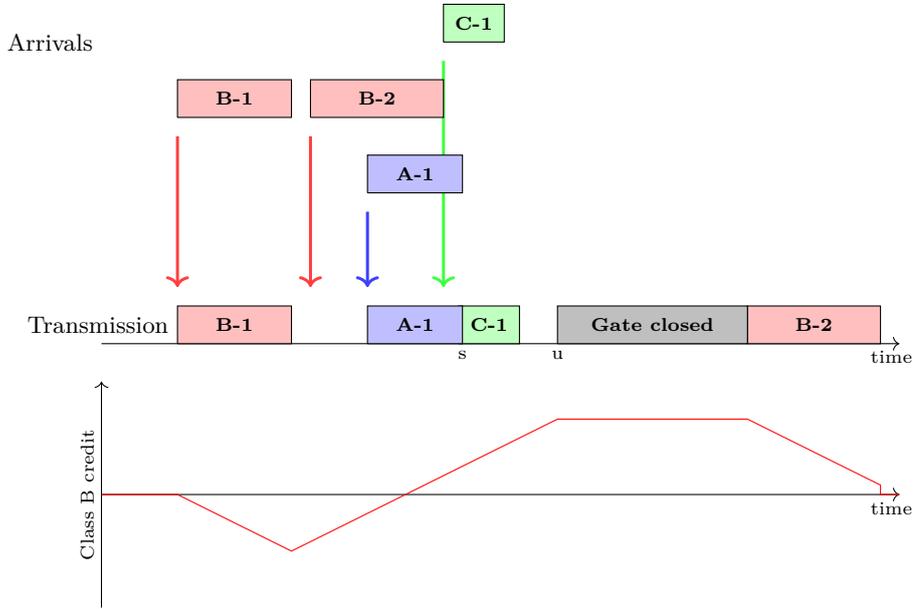


Figure 4.11: CBS amendment for use with Transmission Gates

4.2.4 Configuration Problem for 802.1Qbv

The field of possibilities for the configuration of 802.1Qbv is huge and imagining a random configuration is straightforward. However, finding a configuration, i.e. a selection of parameters' values for the Transmission Selection Algorithms, Transmission Gates and number of queues, in the goal of satisfying Quality of Service requirement is not an easy problem.

4.2.5 802.1Qbv Architectures of Interest

In this section, we introduce several architectures of interest for a 802.1Qbv port that can be found in the literature. For all the architectures below, we assume without loss of generality that all ports use $\#TC = 8$ queues.

Static priority

This subsection will describe the behaviour of a 802.1Qbv output port when there are no TSA configured and the Transmission Gate mechanism is not configured either. In this situation, the output port would look like Fig.4.13.

In the absence of Transmission Selection Algorithms and Gate Control Lists, the queues are directly "connected" to the Transmission Selection block. When a frame is enqueued in one of the

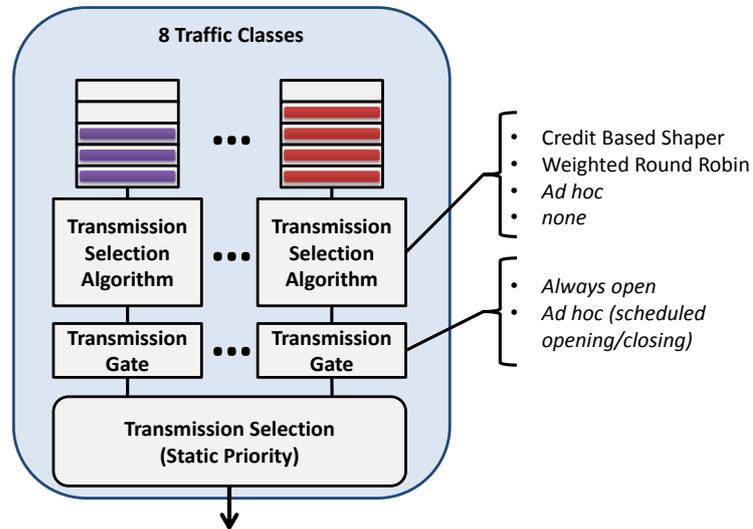


Figure 4.12: Summary of configurable mechanisms in 802.1Qbv

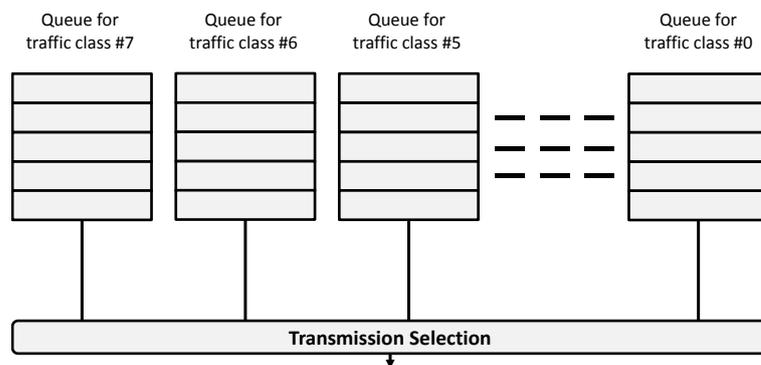


Figure 4.13: Static priority on 8 queues

queue it is immediately available for transmission. If several queues have some frames available for transmission at the same time, the Transmission Selection does its job : it arbitrates between queues with a static priority rule. In this architecture, the Scheduling Type is "Static Priority".

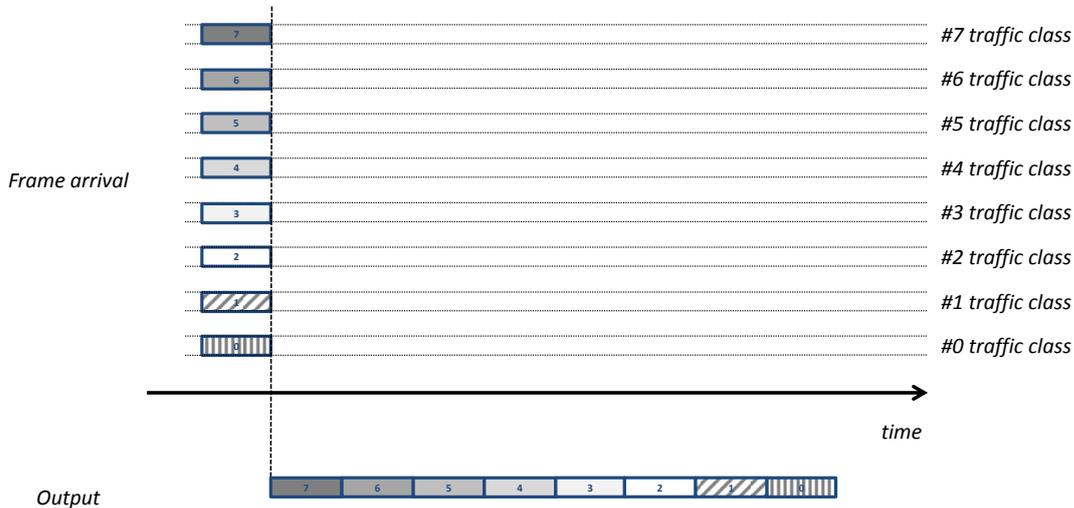


Figure 4.14: Example of static priority on 8 queues

Example 4 Fig 4.14 illustrates the behaviour of such port on a simple example. We suppose that all the queues have a frame ready for transmission at the exact same time. Applying the static priority of Transmission Selection, frame from #7 is emitted first (priority order is decreasing, the lower the queue's number, the lower the frame's priority), followed by the #6 frame, followed by the #5 frame, [...] ending by the #0 frame.

AVB - Audio Video Bridging

This subsection will describe the behaviour of a 802.1Qbv output port when the 2 most important queues use CBS and the other use either nothing, Enhanced Transmission Selection Algorithm (ETS) or Vendor Specific (VS) algorithms and will be named BE (for *Best Effort*). Interactions between queues are arbitrated by Transmission Selection. This architecture is called AVB - Audio Video Bridging, and is widely spread in the Audio-Video world (concert halls, television sets, etc.). Fig. 4.15 introduces an AVB output port architecture.

Example 5 In this example, illustrated by Fig.4.16 (extracted from [28]), we have considered that Queue #A and #B are CBS queues and #C is a best effort queue with no transmission selection algorithm. As a reminder, the interaction between queues #A, #B and #C are arbitrated using strict priority and #A has a higher priority than #B, and #B has a higher priority than #C. On the first part of the drawing we can see frames from #B arriving and getting sent, accordingly to the evolution of B's credit. When a #C frame arrives, #B does not have enough credit to send its frame, hence #C frame is selected and sent. At that time, a #A frame has arrived, we assume that it has credit to emit its frame. As #A has a higher priority than #B, the #A frame is sent before #B-3. Finally #B-3 frame is sent.

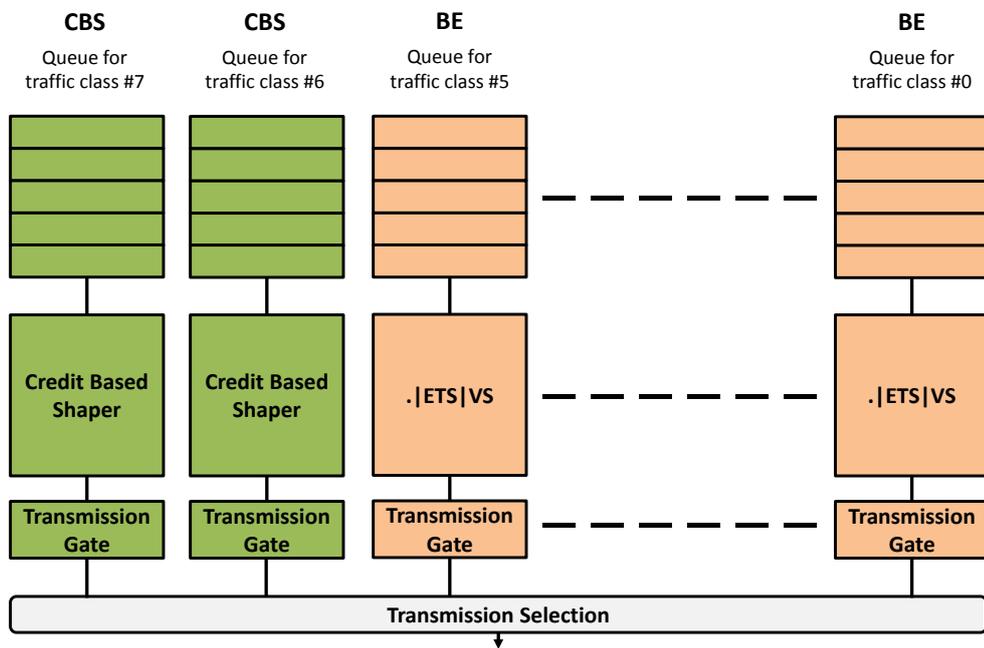


Figure 4.15: AVB port architecture (CBS-BE)

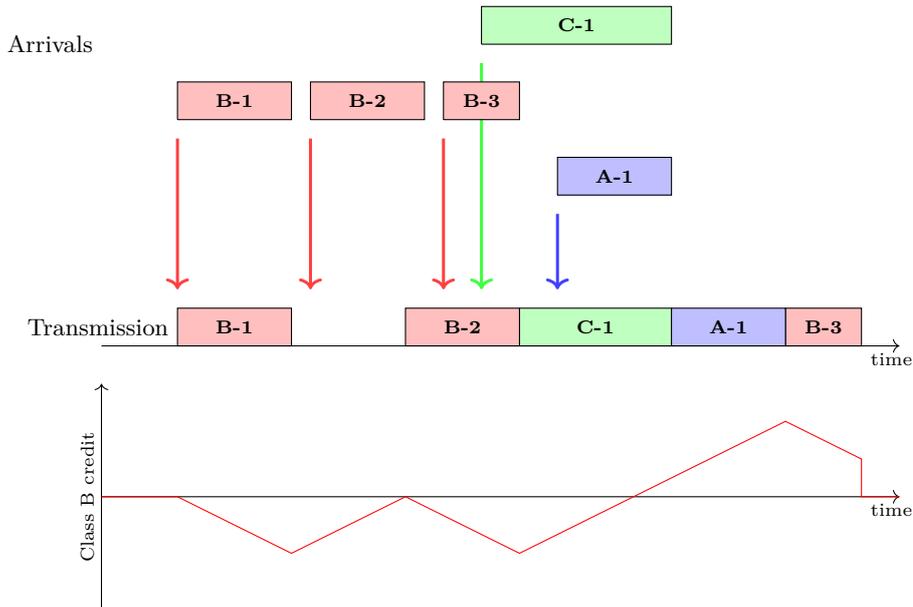


Figure 4.16: AVB Example

End-to-End TT

This subsection will describe the behaviour of a 802.1Qbv output port where all queues use a Transmission Gate but not Transmission Selection Algorithm. In this architecture, the output port would look like Fig. 4.17.

As there are no Transmission Selection Algorithms in any of the queues, all the frames are considered ready all the time. The transmission gates, with their schedule, will decide which frames will be ready for transmission.

Example 6 Fig. 4.18 introduces a "random" configuration of the Transmission Gates (i.e. random schedule). In this configuration, several gates are open in the same time. If frames are ready in different queues the gate of which is open i.e. are available for transmission, Transmission Selection will decide which frame gets emitted first. If only one queue has its gate open, then the queue immediately gets to emit on the medium.

A very common implementation of the Time Aware Shaper is the *exclusive gating principle*.

Definition 31 (Exclusive Gating) In this implementation, all gates have a transmission gate managed in a Gate Control List. When the gate of one specific traffic class #a is Open, the gates of all the other traffic classes, named #[other] are Closed. When the gate of the specific traffic class is Closed, the gates of all the other traffic classes are Open. This will prevent traffic from #[other] to interfere with #a traffic.

One issue remains : what to do when the gates of #[other] traffic classes are about to close ? In the very likely case where a frame is being emitted when the gate has to close, it will interfere with

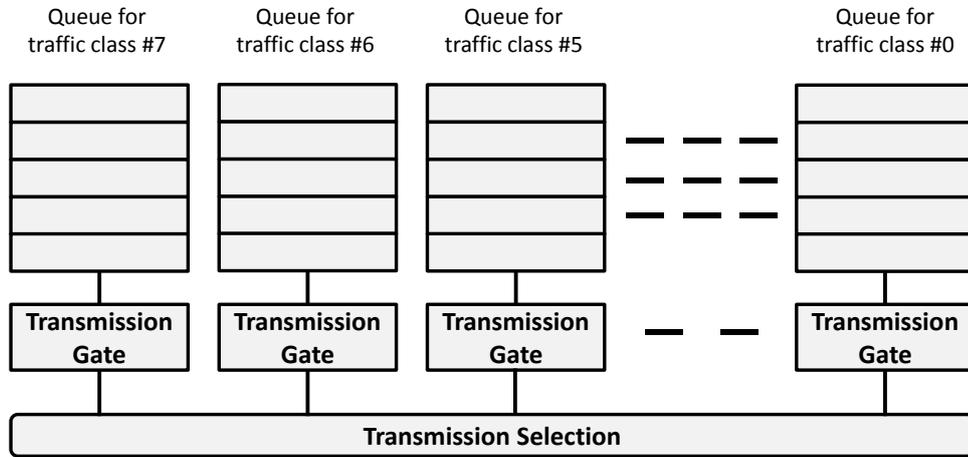


Figure 4.17: End-to-End TT port architecture

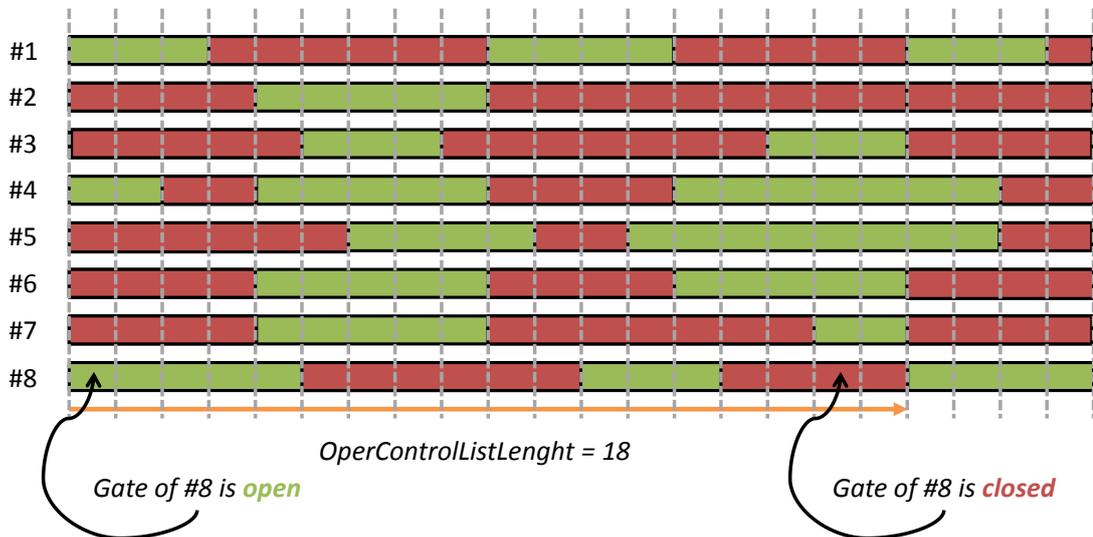


Figure 4.18: Random TAS configuration

the emission of #a traffic. TSN features 802.1Qbu and 803.br proposes two integration methods in order to deal with this issue. It will be detailed in Appendix A.1.

The exclusive gating concept, applied above on #a, can be generalized to be applied to 2 (see Fig. 4.19), 3 and up to 8 queues. In the case where each queue is in exclusive gating with the other queues, the architecture of the output port is equivalent to a End-to-End TT port in which each queue is given in timeslot to emit its frames without any interaction from the other queues. It is not mandatory that the timeslots given to each queue are equals. This End-to-End TT architecture is represented in Fig. 4.20.

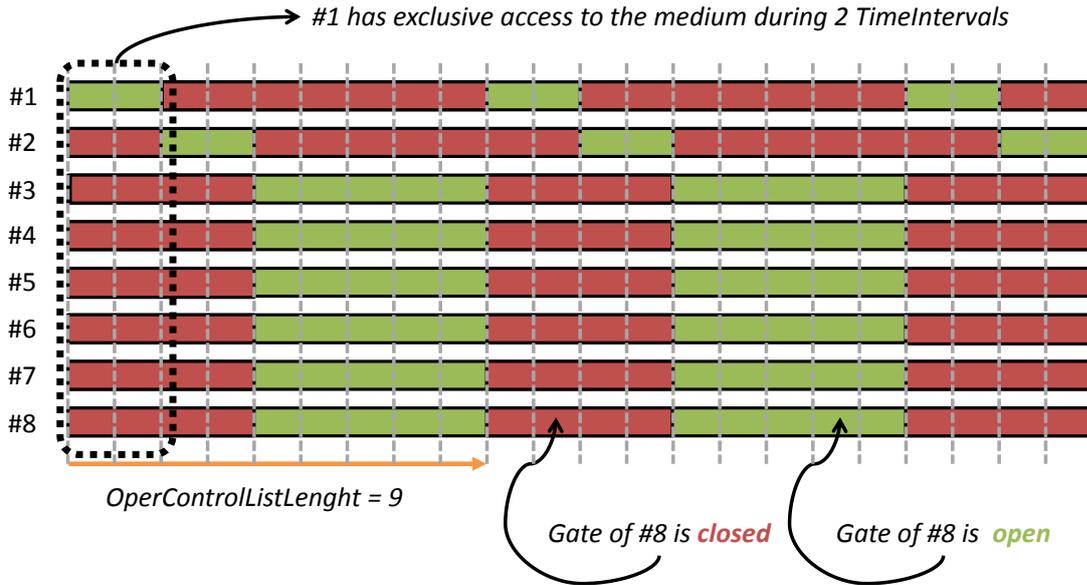


Figure 4.19: TAS Configuration with 2 queues applying the exclusive gating principle

TT-CBS-BE

This subsection will describe the behaviour of a 802.1Qbv output port when the highest priority queue uses no TSA and a TG in exclusive gating with the other queues. The two next higher priority queues remaining will use CBS and the rest will use either nothing, ETS or Vendor Specific algorithms and will be named BE. With such organization, an output port would look like that Fig.4.21. We will not develop further the behaviour and performances of *TT-CBS-BE* as we have already deeply explained the other architectures of interests and all the concepts before. In-depth analysis and modelling of this architecture was realized in [30].

It is not all !

In the previous subsections, we have introduced several architectures of interest regarding the configuration of TSN features 802.1Qbv. But these are not the only ones, the field of possibilities is nearly infinite as anyone can choose for each traffic class of a single port a different number of queues, a different Transmission Selection Algorithm and a different Gate Schedule.

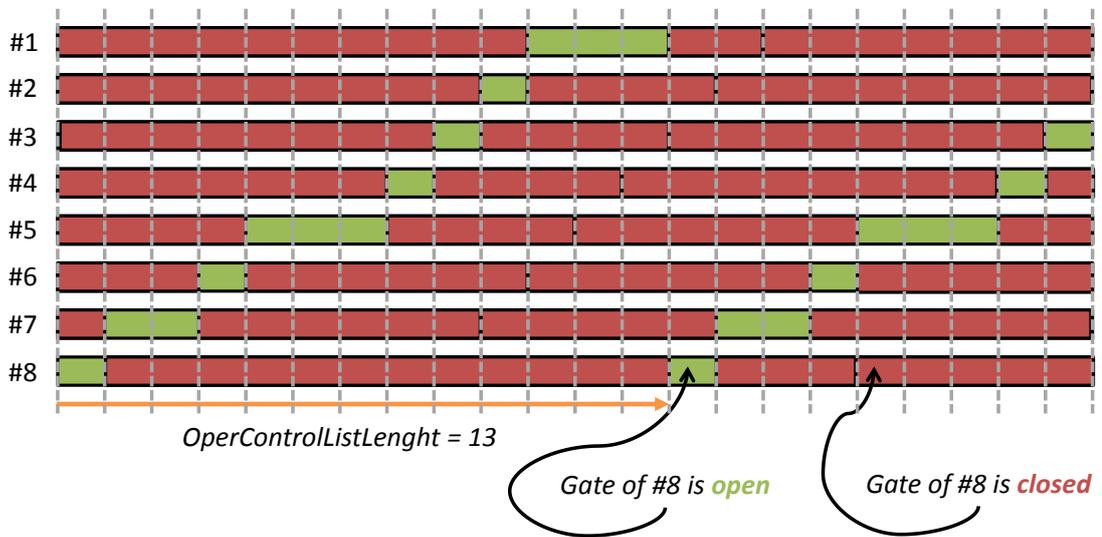


Figure 4.20: End-to-End TT architecture

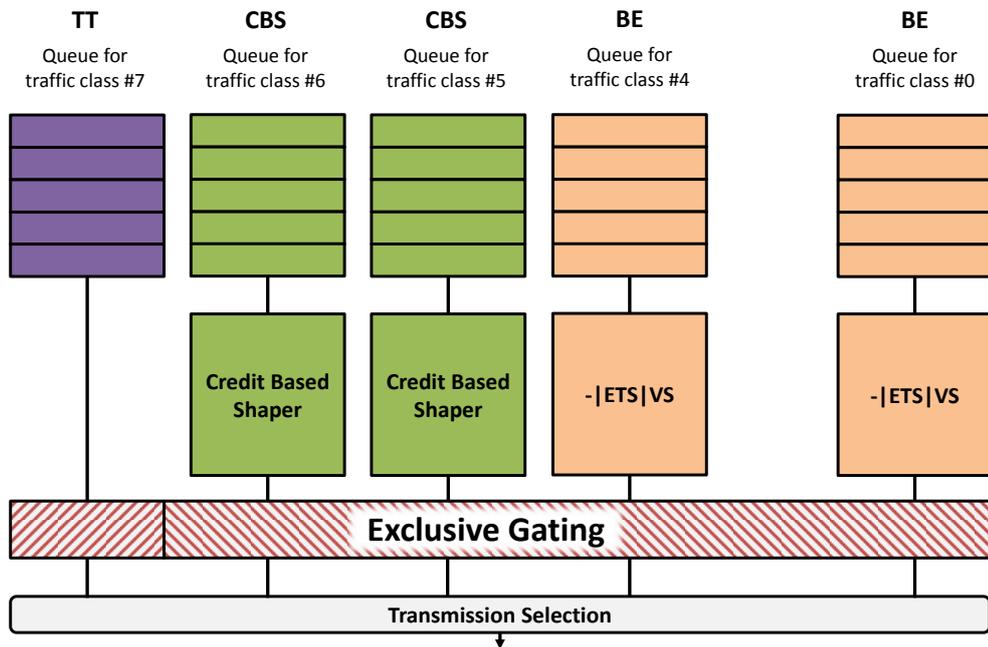


Figure 4.21: TT-CBS-BE port architecture

4.2.6 Summary

To summarize, TSN feature 802.1Qbv - *Enhancement for Scheduled Traffic* add new bandwidth sharing mechanism through Transmission Selection Algorithm (Credit Based Shaper, Weighted Round Robin, Static Priority, etc.) as well as Time Triggering capabilities through Time Aware Shaper to the TSN switches output ports. The same elements are added to the output ports of the end-stations. Fig. 4.12 summarizes the configurable mechanisms of 802.1Qbv. With this set of mechanism, TSN is able to achieve bounded or even fixed latency as well as very low reception jitter.

Conclusion

This chapter has introduced some generalities on IEEE Time Sensitive Networking as well as TSN addenda for network performance quality of service and fault tolerance quality of service. In chapter III, we will aim at finding a configuration of the mechanisms of 802.1Qbv. Before doing so, we propose in the next part to make sure that TSN is indeed an interesting technology for a next-generation satellite network.

Part II

Contribution: Selection of Candidate Technologies Compliant with the Quality of Service Requirements of a Next-Generation Satellite Network

Chapter 5

Problem Statement 1

In accordance with the ever-expanding volume of data generated and handled by ground-level equipments (telephone, cars, scientific instruments, etc.), satellites must be capable of *producing* and *transmitting* massive amounts of data in order to meet their users' requirements. While improving the performance of data production is straightforward since instruments with such capabilities are already available on the market (e.g. COTS multi-gigabit camera, etc.); improving the performance of the on-board networks carrying that data remains complex.

The current networking technologies (mostly 1553 and SpaceWire) will not be able to support that trend for long. Therefore, the spacecraft industry has started to consider new technologies so as to keep up with the demand. The technology supporting the next generation satellite network is expected to create a "*unified*" network, meaning that *Platform* and *Payload* traffic will co-exist in the same network, while proposing improved quality of service (performance and fault tolerance) at a reasonable cost.

There are not one but several technologies currently considered by the industry as prospective solutions for next-generation satellite networks. The first section of this chapter presents the list of pre-selected technologies including a rationale for their pre-selection. The goal of this first problem will be to identify one or several candidates capable of fulfilling the satellite requirements. To do so, expectations on the technologies supporting the communication network have to be refined meaning that it is necessary to formalize further the behaviour of space applications and legacy design paradigm. For that purpose, we present an abstract model of the equipments (or devices) and application, from which we derive a first set of requirements that we call *Application Level Properties*.

5.1 Preselected Technologies

Although the actual architecture works perfectly fine, it has started to show its limits: new instruments and more generally new equipments are capable of generating gigabits of data that the network cannot handle in its current version i.e. 100Mbits/s on a SpaceWire network. Using gigabits-capable network could allow satellite users to access this huge amount of raw data. Moreover, adding more mechanisms at network level (ISO Level 2) could ease the integration of an increasing number of equipments on-board and reduce the development effort to be done at application level.

At European Space Agency level, SpaceFibre is the high-throughput networking technology successor of SpaceWire, it is hence naturally considered in this study. Nevertheless, SpaceFibre is only

used in the spacecraft industry, thus its development and updates are quite expensive, in particular in terms of non-recurring costs.

Using a technology based on COTS - *Commercial-off-the-shelves* - components, or IP Cores - *Semiconductor Intellectual Property Cores* (instantiated into specific space oriented hardware), shared, for some parts or as a whole, with other industrial sectors (automotive, industrial automation, aeronautics, etc.) could help lower the overall cost of the satellite network. Having a wide-spread technology could also facilitate the interaction between the spacecraft industry and the academic world. This is the reason why the focus is steered towards Ethernet-based technologies.

In fact, Ethernet has started being considered/used in industrial systems since the early 2000's. In the spacecraft industry, Ethernet is being used on board the International Space Station (ISS) and is starting to be embedded in satellites to convey payload traffic between the instrument and its associated computing device on a point-to-point fashion. In the aircraft industry, an enhanced version of Ethernet i.e. ARINC 664 is used to convey flight control traffic (with stringent quality of service requirements). It is an on-board networking technology which offers real-time and fault tolerance guarantees. ARINC 664 is widely used in Airbus planes, it is therefore logical to try to re-use it in Airbus satellites. That is why this technology is preselected for our study. In the launcher industry, TTEthernet will be used on board Ariane 6 rocket as on-board network (with quality of service requirements equivalent to our platform side in the satellite). It is also planned to be used in space for, among others, the space gateway of the ARTEMIS mission towards moon [122]. TTEthernet also offers real-time and fault tolerance guarantees. In addition, there are already space hardened components available for TTEthernet. That is why TTEthernet is preselected for our study. Finally, one opportunity has appeared with Time Sensitive Networking, the state of the art IEEE Ethernet technology. It is being considered by several industry verticals e.g. automotive industry [86], aircraft industry [113], industry automation [95], [81], 5G [76], train industry [72], spacecraft & launcher industry [103], etc. Hoping that the development effort could be shared between industries and that the spacecraft industry could benefit from a scale effect due to the other industries buying the same components on a much larger scale, Time Sensitive Networking is preselected for our study.

To summarize five technologies are pre-selected for this study: SpaceFibre and four technologies from the Ethernet family: Ethernet, ARINC 664, TTEthernet and Time Sensitive Networking. These technologies are introduced in Chapters 3 and 4.

5.2 Satellite On-Board Applications Modelling

To support the definition of *Application Level Requirements*, it is necessary to detail the behaviour of applications and the temporal parameters associated to it. Let us first model the end-stations and applications of the satellite system introduced in Chapter 2. The definitions and models of this chapter have been retrieved from [17].

5.2.1 Devices

In the model, we define end-stations as *devices*.

Definition 32 (\mathcal{D} , Devices) *Let \mathcal{D} denote the set of devices. The devices ($Dev \in \mathcal{D}$) can be part of four different families:*

- *Platform Computing: The platform computing devices process data from sensing devices and generate commands sent to actuating devices.*

- *Sensing*: The sensing devices collect data through analog or digital sensors.
- *Actuating*: The actuating devices execute commands received from a computing device so as to control the attitude and the orbit of the satellite.
- *Payload Computing*: the payload computing devices collect useful data for the mission, for example: antennas and transponders for telecommunication satellites, telescopes and cameras for Earth/Space observation satellites or basically any type of scientific instruments for scientific missions. They also process this data before storing in an on-board mass memory and sending it to Earth.

Definition 33 (Remote Interface Unit, a data concentrator) *Some of the sensing or actuating devices are analog devices and hence are connected to a RIU for Remote Interface Unit that acts as controller for these devices. This RIU will also be considered as an Actuating or a Sensing node.*

Property 1 (Master/Slave communication paradigm) *Both Actuating and Sensing devices act as slaves of the Platform Computing devices. This entails that Actuating and Sensing devices can only use the communication system if they were previously asked to by a Computing device.*

5.2.2 Motivating Example

All along the problem statement, we will use a motivating example to illustrate the given definitions, constraints, etc. Every now and then, we will use this example or add to it more details so as to ease the understanding of the concepts of this document. This example, for now, is composed of three devices:

- One Platform Computing Device that we will name *OBC*¹,
- One Sensing Device that we will call *High Performance Sensor*,
- One Actuating Device that we will call *High Performance Actuator*, that is, in reality, connected through a *RIU - Remote Interface Unit*.

5.2.3 Applications

Applications (e.g. *Command & Control*, *Vision Based Navigation*, *Payload Processing*, etc.) run on the Computing devices. On Platform Computing devices, applications gather, process, and produce data that serves for the control of the satellite, for instance the control of gyroscopic actuators to orient a satellite in order to take a picture of a specific spot on Earth, or the use of on-board cameras to detect any incoming obstacle, and control the thrusters of the satellite to avoid the threat. On Payload Computing devices, applications also gather, process, and produce data but for the payload part of the satellite. For instance, they can process the raw stream of data retrieved from the camera sensor of the satellite and build images or videos that will then be retransmitted to Earth via the communication subsystem of the satellite connected to ground stations. Several time constants are used to define the behaviour of these applications.

Definition 34 (MIF Cycle, P_{MIF}) *Each application follows a pattern, named a cycle, shown in Fig. 5.2. The duration of this cycle is constant per satellite and is called the MIF - MInor Frame - Period, denoted P_{MIF} . During this cycle, a reserved time quantum of application time is dedicated to gathering input data coming from sensing devices through the on-board network, then another quantum of its time is dedicated to processing this data and one last quantum is dedicated to sending output data to actuating devices according to the output of the processing via the on-board network.*

¹On-Board Computer

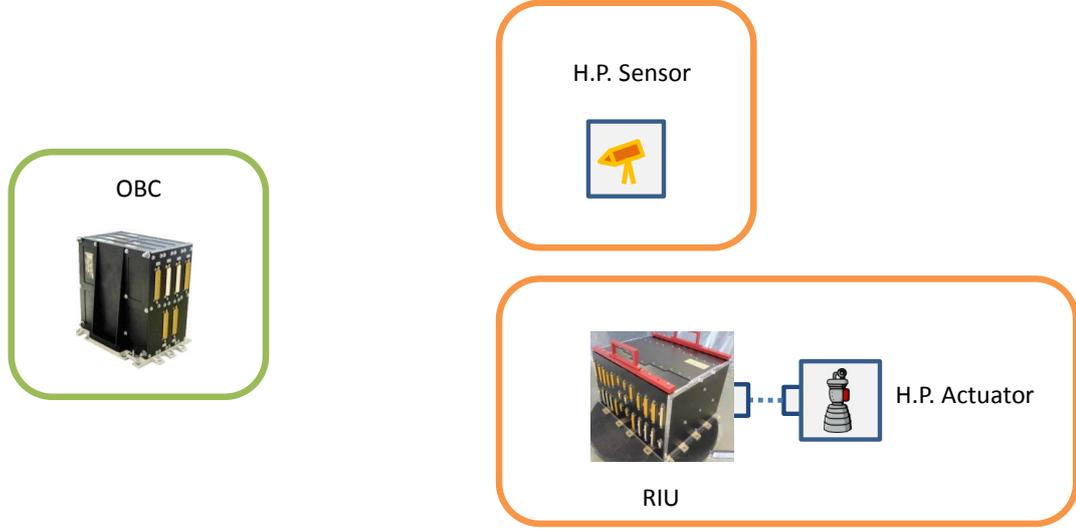


Figure 5.1: Motivating Example with three devices

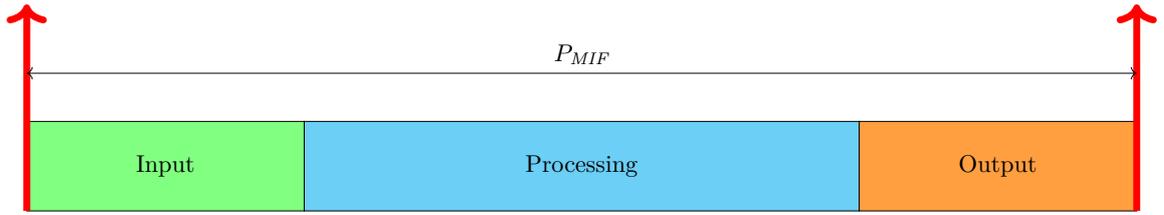


Figure 5.2: Typical application pattern

This above definition is illustrated with Fig. 5.2. In the figure, the duration P_{MIF} is represented by the distance between the two upward-facing arrows. Then, the green rectangle represents the part where application potentially gathers input, the blue part represents the application processing time and the orange one represents the part where application potentially emits its messages.

Remark 8 *Although it may appear constant in the figures of this document, the duration of the Input, Processing and Output phases is not required to be constant in the satellite system.*

Rule 1 *All values for periods in this document will be expressed in milliseconds.*

Definition 35 (MAF Cycle, P_{MAF}) *In the satellite, the MIF period represents the duration of one applicative cycle. As several applications co-exist in the satellite, a system-wide period or hyperperiod is defined. This period is call a MAF - MAJor Frame - Period (or MAF Cycle) and its duration is denoted P_{MAF} .*

The MIF cycle of duration P_{MIF} is repeated k ($\in \mathbb{N}^$) times during a MAF Cycle so that:*

$$P_{MAF} = k * P_{MIF} \quad (5.1)$$

Hypothesis 1 In the system of our industrial partner, the usual value of P_{MAF} was 1000ms. Tab. 5.1 gives usual values of k and corresponding usual values of P_{MIF} with $P_{MAF} = 1000$.

k	P_{MIF}
8	125ms
16	62.5ms
32	31.25ms

Table 5.1: Usual values of k and P_{MIF}

This hypothesis is illustrated in Fig. 5.3.

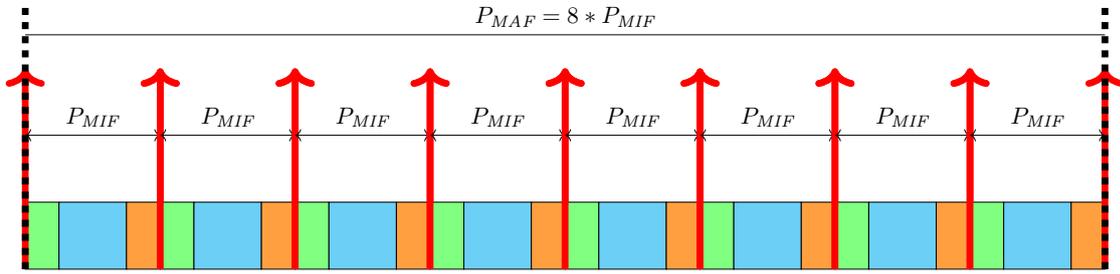


Figure 5.3: Link $P_{MIF} - P_{MAF}$ with $k = 8$

5.3 Application Level Properties

Because of the gathering of input at the beginning of the MIF cycle and the output at the end of the MIF cycle, multiple messages for several applications must share the same medium at the same time. Those applications probably have different requirements on network performance quality of service. As a consequence, the on-board network shall be able to satisfy these performance QoS requirements. In addition, in order for applications scattered all across the network to work properly, they all require to have the same *understanding* of time (by for instance being synchronized). Finally, since resources are shared between applications, in order to prevent a fault coming from one application to have an impact on the nominal behaviour of the other applications, a fault tolerance quality of service requirement is added. Therefore, at this stage, we identify three requirements or *application level properties* that the on-board network shall fulfil: *Mixed Traffic Types*, *Time Management* and *Fault Tolerant Operations*.

5.3.1 Mixed Traffic Types

Application-Level Property 1 (Mixed Traffic Types) *Capability of the network system to convey, at the same time, several flows with different characteristics for instance, low data rate with low jitter and high data rate traffic but to prevent traffic with different criticality to affect each other's performances.*

For instance, a network satisfying Application Level Property. 1 shall be able to convey, with the same equipments, low data rate traffic with network performance requirements (e.g. messages for the command & control of a thruster requiring very low reception jitter); and high data rate traffic with no network performance requirements (e.g. data from a Payload Computing device sent to the communication sub-system for its transmission towards Earth using most of a 1Gbits/s link).

5.3.2 Time Management

Application-Level Property 2 (Time Management) *Capability of the network system to manage time, i.e. ensuring either a global common clock of all network elements or at least applicative time distribution.*

Remark 9 *While the industrial partner would be ready to consider that part of the time management would remain at application level, in this PhD, we took the hypothesis and requirement that Time Management should be done at MAC level (ISO level 2).*

5.3.3 Fault Tolerant Operations

Before defining the third *application level property*, let us first explain that we consider a faulty behaviour as either incorrect, lost, out of time constraints or out of traffic contracts.

Example 7 (Examples of Faulty Behaviours) *An example of incorrect faulty behaviour would be a message sent by an application and received by another which content has been altered (due for instance a bit-flip, which is quite common in space environment). An example of lost faulty behaviour would be a message sent by an application that never reaches its destination application (due for instance to the message being discarded after a CRC error). An example of out of time constraints fault behaviour would be a message sent by an application expected to be received within a specific time window and received outside of this time window (due for instance to a synchronization error between the sending and receiving devices). Finally, an out of traffic contract faulty behaviour would be a message sent by an application while said application has already used all the resources that were allocated for it (due for instance to a device turning into a babbling idiot).*

Application-Level Property 3 (Fault Tolerant Operations) *Capability of the network system to operate in a faulty context by preventing faults, by detecting, isolating and recovering from certain faults and by generating failures report/indicators for higher level fault management in case fault cannot be dealt locally, in a seamless manner.*

Remark 10 *The above paragraph only addresses a selection of all types of faulty behaviour that could occur in the system. Examples of such faulty behaviours are impersonation (i.e. an application emitting message in place of another application), routing error (i.e. a message not taking the correct path), etc.*

5.4 Contribution Overview

The applications running over the devices require a network to communicate with each other. This network can be implemented with several technologies. Therefore, we wondered :

Definition 36 (Problem 1) *Given the set of technologies preselected by the industrial partner, is it possible to identify one or several technologies that would be good candidates for a next-generation*

satellite network based on their compliance to the application level properties on a qualitative approach ?

To confront this first problem, we proposed to compare the preselected technologies on a qualitative manner. To do so, we defined a set of criteria per application level property that would serve as metric for the comparison. The criteria, the comparison and its results have been published in [19] and in [20]. The first paper only discusses the suitability of Ethernet technologies (i.e. Ethernet, ARINC 664, TTEthernet and Time Sensitive Networking) while [20] complements the previous comparison by adding SpaceFibre. Part II will detail the identification of the criteria for the comparison followed by the comparison itself, criteria per criteria, on a qualitative basis. It will conclude by the identification of three suitable candidates for the next generation satellite network.

Chapter 6

Methodology: Qualitative Comparison Relying On Criteria Regrouped Per Properties

In order to select one or several technologies suitable for a unified *Platform & Payload* satellite on-board network, we propose to compare the preselected technologies (presented in Chapters 3 and 4) in a qualitative fashion. Willing to reuse as much as possible the bibliography, we propose in the first section a state of the art regarding the selection of a next-generation network. The state of the art focuses on two aspects: definition of criteria to compare technologies and the existing comparison of technologies. The second part of this chapter is the presentation of the criteria we will use for our comparison.

6.1 Related Works: Selection of a Next Generation On-Board Network

This state of the art deals, in the first section, with the space sector, mostly in the satellite domain, and then opens the focus, in the second section, to other industrial sectors such as automotive, industrial automation, etc. We will reuse the methodology of the state of the art for the comparison. In fact, authors define both a set of technologies and a set of criteria representative of the requirements of the system they consider (e.g. a satellite, a car, a train, etc.). Then, for each criteria, they evaluate how a specific technology performs w.r.t. to said criteria. Then, they analyse the "value" of every criteria w.r.t. to their system requirements.

Example 8 *For instance, with a criteria Maximum Data Rate, and a set of technologies like {Ethernet, Spacewire}, Ethernet will be able to perform at 1Gbit/s or more, Spacewire will be able to perform at about 200Mbit/s. Considering a system requirement at 800Mbit/s, the analysis of the value of the criteria Maximum Data Rate for the set of technologies {Ethernet, Spacewire}, leads to the conclusion that Ethernet is suitable and Spacewire is not for this particular criteria.*

Eventually, a summary of the analysis of all the defined criteria is provided so as to identify the best choice(s) for the considered system.

6.1.1 In the Space Domain

Spacewire and MIL-STD-1553 are the most common technologies used as on-board bus/network in the spacecraft industry today. However, their limits in terms of performance, scalability, flexibility, etc. have left space industrials wonder about new technologies. In [18], we have introduced the need for an upgrade of the satellite network and identified several challenges on the use of TSN for space applications.

Closest works

The two closest works to our first contribution are [49] and [97]. [49] proposed the very first comparison of a huge set of technologies for Spacecraft Avionics Systems but only for the *Platform* network. The goal was to provide a comparison of these technologies for man-rated spacecraft with low jitter and low data rate traffic. The authors considered 11 technologies: MIL-STD-1553, SAFEbus, Time-Triggered Communication Protocol (TTP), FlexRay, Time-Triggered Controller Area Network (TT-CAN), IEEE 1394b, SpaceWire, Ethernet 10/100 Base-T, Avionics Full-Duplex Switch Ethernet (AFDX i.e. ARINC 664), Fibre Channel and Gigabit Ethernet (i.e. Ethernet). Apart from an extensive presentation of each technology, the major contribution was the table provided in appendix which is a comparison matrix of all technologies on 45 criteria. These criteria, such as "*Maximum Data Rate, Latency, Jitter, Clock Synchronization, Fault Containment, Babbling Idiot Avoidance*" . . . , include most of ours. The others are either a subdivision of the criteria we introduce in the next chapter or out of the scope of the coming qualitative comparison (for instance physical-layer related criteria). For man-rated applications, the technology they selected was TTP/C. We do not reuse their conclusion as the requirements of our satellite system differ from their spacecraft's requirements (i.e. we consider high data rate traffic in addition to low jitter traffic). [97] proposed in 2016 a first high level analysis comparing ARINC 664, TTEthernet and Ethernet for Space (a tailoring of ARINC 664 for space applications), without really relying on clearly defined criteria. They identified the opportunity of an Ethernet-based network as a next-generation satellite network without really selecting one specific technology. We add more technologies to the comparison they have started, namely Ethernet and TSN, and provide a more in-depth analysis of application level properties for each technology.

Switched Ethernet and ARINC 664

In 2002, [121] was most likely the first work to address a detailed implementation of a full-duplex switched Ethernet for space applications. At the time, it was already considered as both a *Platform & Payload* network candidate and the authors compared it to 1553. The authors concluded that Ethernet was indeed interesting but that further studies were required to precisely quantify the bounded latency capability of Ethernet and the possible implementations of redundancy protocols over Ethernet.

[91] introduced several requirements oriented towards space robotics needs and provided a high level analysis of the capabilities of SpaceWire-2/SpaceFibre w.r.t. to said requirements. Their analysis comprised most of the criteria for Application Level Properties 1, 2, and 3 that we consider as well as some additional ones (e.g. considerations on configuration process of a SpaceFibre network). The authors concluded that SpaceWire-2/SpaceFibre could be a potential candidate for the upgrade of the satellite network.

[114] proposed a detailed analysis of ARINC 664 for "*Space On-board Data Networks*". In the analysis the authors only considered part of Application Level Property 1 as they only discussed the latency and jitter criteria. Typical values for latency and jitter were assessed via simulation on a network working at 100Mbps/s. They concluded that on a first approach, ARINC 664 could be considered as a suitable candidate. While for the use case presented in their paper, we agree with the conclusion proposed in the paper, in our contribution we disagree with the authors and declare ARINC 664 not suitable for a next-generation satellite network based on the analysis of Application Level Properties 2 and 3 as well as more stringent requirements on jitter in our case.

TTEthernet and Time Sensitive Networking

In [112], the authors described TTEthernet and showcased its interest for the spacecraft industry by identifying several interconnections possibilities between TTEthernet and Spacewire for a satellite backbone network. Most recently, [74] discussed of the interest of Ethernet based networks and in particular of TSN based networks for next-generation on-board computer and data handling architectures. The discussion was high level but went through the same application level properties that those of our contribution where we address them in detail.

The launcher (i.e. rockets) industry has not been spared from the network evolution trend of the space domain. In Europe, the interest of ARINC 664, TTEthernet and Time Sensitive Networking has been discussed for the newest launchers and some relying on TTEthernet and Time Sensitive Networking will lift-off in 2022.

In [22], the authors proposed a discussion on the network for ARIANE 6 launcher. In the paper, they selected TTEthernet as a suitable candidate among several technologies including Spacewire, Flexray, TTP, Ethernet, ARINC 664, TTEthernet and Fibre Channel. While the comparison process and results are not available in the paper, the criteria used by the authors are the following: Cost, RAMS, AIT-OPS (Assembly Integration & Test, Operations), Environment withstanding, Flexibility/Adaptability to design modification or obsolescence and Maturity (described by a TRL level). We assume that performance (Application Level Property 1) was also addressed but not listed. Application Level Property 2 does not seem to have been addressed. Two candidates were actually identified ARINC 664 and TTEthernet. However, TTEthernet was favoured because it was tailored for their applicative needs, and because of its flexibility and its simplicity for software validation. With the in-detail comparison we propose in the following chapters, we will also identify TTE as a suitable candidate for a next-generation satellite network.

Later, authors from the same group started to consider TSN in place of TTEthernet. In paper [94], the authors mentioned, among other things, the interest of the launcher industry for going towards an Ethernet-based technology relying on COTS components so as to reduce costs. The technology they chose was TSN (instead of TTEthernet). However, they raised the reader's awareness on the additional studies required to assess the behaviour of COTS components in space. We share the authors' view on COTS components and we actually detail it in the last chapter of this first contribution.

Finally, [103] presented the TSN implementation used for the network of MIURA 1 micro-launcher that will go to space in the coming years. The networking technologies considered by the authors are really close to ours, the only missing technology being Spacefibre. Again, as for the previous paper, the authors identified the interest of the use of COTS components for the launcher networks. TSN has been selected for its ability to handle critical data (Application Level Property 1) and redundancy (Application Level Property 3), as well as the expected availability of TSN COTS components. In the coming comparison, we agree with the authors on the interest of TSN and COTS components.

There is therefore, in the space domain (satellite and launchers) a real interest in the selection of a next generation on-board network. The trend seems to predict that the technologies that were indeed pre-selected for our contribution would be good candidates. In the coming contribution, we will reduce this set of potential candidates with the help of the qualitative comparison.

6.1.2 Outside the Space Domain

Outside of the space domain, the race for evolving from an industry specific legacy bus towards a faster, better and more common technology is also raging, in particular in the automotive and factory automation domains, but also in the avionic industry. A good illustration of this "race" would probably be [90] where the authors propose a historical list of every, used or emerging at the time (2013), in-car embedded networks. This list does include Ethernet, and TTEthernet but lacks TSN which did not exist at the time and SpaceFibre which is out of scope of the automotive industry.

Most of the works presented hereafter use simulation to obtain latency and jitter value for several network configurations. Moreover, in the following papers, the definition of performance quality of service requirements is not identical to ours. While most related works chose their technologies based on the best performance the technologies are able to provide (i.e. very low jitter and smallest network latency possible), our comparison is motivated by a satellite use case where jitter shall be kept low but latency constraints are not strong. In fact, in our industrial use case, latencies can be as large as possible as long as no deadlines are missed.

Automotive Industry

[27] proposed a comparison similar ours, but applied to the automotive industry. Their analysis targeted the evaluation of the use of AFDX (i.e. ARINC 664), TTEthernet, EtherCAT and AVB in an automotive context. They considered Application Level Property 1 and 3 but also addressed physical layer, system start-up and costs aspects. We consider these aspects out of scope of our comparison since we only wish to discuss the protocol theoretical capabilities and not the performance of one implementation at this stage. With the qualitative comparison of Chapter 7, we complete their work with the comparison on Application Level Property 2 and additional analysis on Application Level Property 3 applied to a space use case. Still in an automotive context, [86] proposed a comparison of Ethernet, AVB and TSN. The comparison only focused on Application Level Property 1 and more specifically on latencies and their bounds in an automotive use case. Latencies were obtained via simulation and their bounds via worst-case schedulability analysis. The authors concluded by identifying that, among their candidates, TSN was the most suited for the automotive use case requirements. However, the authors rightfully claimed that the configuration and timing validation process of a TSN network are complex and that tools are needed for it.

TSN, the successor of AVB, can reproduce, via configuration, the behaviour of AVB. Hence, the results observed in AVB are transposable to TSN when configured in AVB mode. [104] proposed a comparison between AFDX and AVB in an avionic context. The authors then, based on network calculus, declared that the observed bound on latencies obtained with an AVB network were suitable for aircraft avionics' network requirements. Their paper also stated that the new mass market controllers and switches developed for AVB (now TSN) could be a cost-effective alternative for COTS-based low criticality systems. We share their view on the performance and cost analysis. In particular, the mass market of COTS component could benefit to satellite production in a "New Space" context (business analysis pending). However, we do not discuss AVB in our comparison

since, for our specific application, it does not match our needs in terms of jitter.

[109] proposed a competitive performance evaluation of AVB and TTEthernet. The evaluation was performed on an automotive use case via simulation in OMNET++. Results of the evaluation showed comparable performances for both technologies in terms of latencies (Application Level Property 1). However, the authors identified that background traffic (without deadline and jitter constraints) had a significant impact in AVB and planned to make more evaluation while adding background traffic to quantify its impact on performances.

[5] also did simulative assessment of the performances of AVB and TTEthernet in an automotive context that lead them to the same conclusions than [109].

[79] compared the performances of AVB and Ethernet in an automotive use case at 100Mbit/s. In our analysis, we compare the performances of TSN and Ethernet at 1Gbits/s.

[11] compared the performance in terms of latency and jitter (Application Level Property 1) of AVB and "AVB-ST", for Scheduled Traffic (which would become TSN configured in TT-CBS-BE mode cf. Section 4.2.5), on an automotive use case via simulation using OMNET ++. The authors concluded that the performance of both technologies seemed suitable for the automotive use case but the introduction of a scheduled traffic class in AVB-ST would, in combination with other mechanisms, provide low and bounded latencies to critical traffic even with a high bandwidth traffic using the same resources. Again, this confirmed already at the time the interest of the automotive industry for TSN.

Railway Industry

[72] addressed the use of ARINC 664, TTEthernet and Time Sensitive Networking for deterministic (understood as bounded latency and low jitter) data delivery and briefly compared them without really choosing a winning technology at the end of the comparison. This paper was coming from a railway context, and the networking technology was discussed as a building block of a Train Control/Management System (TCMS). ARINC 664, TTEthernet and Time Sensitive Networking were compared on Application Level Properties 1 and 2. The criteria for the comparison included the ones we use in the coming comparison at the exception of one i.e. whether the synchronization capabilities of TTEthernet and Time Sensitive Networking are available at ISO L2 or not. We discuss it in the next chapter.

Aircraft Industry

[124] proposed a comparison between TTEthernet and TSN. The comparison was very thorough. Several properties, similar to ours were used for the comparison i.e. *synchronization, bandwidth allocation, traffic shaping and traffic scheduling and redundancy*. The authors computed worst case delays in a specific TSN configurations based on network calculus. We redo the qualitative protocol comparison with our specific performance requirements and our use case and add a more detailed redundancy analysis.

Helicopter Industry

Most recently, [84] proposed an evaluation of the suitability of TSN for an Helicopter network. The performance requirements were slightly different: while we consider deadline and jitter requirements, the helicopter network traffic is only subject to deadline requirements. Therefore, based on coarse-grained analysis using RTAW Pegase commercial tool [99], they declared that TSN might not be the number one solution for a next-generation helicopter network.

Conclusion

The previous sections have presented a set of papers discussing the interest of certain technologies for the next-generation on-board networks in different fields of applications (e.g. satellites, launchers, cars, etc.). Several papers already compared some of our preselected technologies together but the comparison was either too light or not compatible with our system model. For instance, the definition of jitter presented in Section 8.2 describes the end-to-end jitter as latency variability as it is the case in the common jitter definitions in the state of the art. However our latency concept differs: we consider the duration between the reception instant and a reference instant for any frame (see Chapter 8.1) instead of the difference between reception and emission instants. Therefore, while we do not contradict any of the papers in the state of the art, our conclusion might still differ. As a consequence, we reuse the criteria or the approach of certain papers but apply them to the specific requirements of a satellite next-generation on-board network. In fact, the comparison of the following chapter has never been done before on the full set of technologies presented in Chapter 3 and 4, as well as on the full set of considered criteria, with the goal of discussing the suitability of a technology for a next-generation network supporting both *Platform* and *Payload* requirements.

6.2 Definition of Our Criteria

In order to select, based on a qualitative comparison, one or several candidate for a next generation satellite network, it is necessary to define the extensive list of criteria on which the comparison will be based on. In the literature, the Federal Aviation Administration proposed in a report a list of criteria that could be used for "assessing the design of existing and new data networks for their applicability to safety-critical aviation digital electronics systems"[2]. The criteria presented hereafter are inspired from this report and driven by the requirements of the satellite system. Therefore, in this section, we list and define our criteria for the comparison, regrouped per application level properties. Section 6.2.1 introduces three criteria related to *Mixed Traffic Types* Application Level Property. Section 6.2.2 introduces three criteria related to *Time Management* Application Level Property. Section 6.2.3 introduces three criteria related to *Fault Tolerant Operation* Application Level Property.

6.2.1 Criteria for Application Level Property 1

To evaluate the suitability of the preselected technologies i.e. Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking and SpaceFibre, presented in Chapter 3 and 4, with Application Level Property 1 - *Mixed Traffic Types*, four criteria are identified: *High Data Rate*, *Bounded Latency*, *Very Low Jitter* and *Mixed Criticality*.

Remark 11 *One very important remark regarding this comparison and its associated criteria, is that the criteria evaluate what it is possible to achieve using the protocol at L2 level, not what could be achieved by a clever implementation of the protocol in the whole system.*

Criteria 1 (High Data Rate) *Is the considered networking technology capable of handling 1Gbit/s or more ?*

This criteria is used to eliminate potential candidates that would be too slow for next-generation requirements. Note that neither SpaceWire or 1553 satisfies this criteria.

Criteria 2 (Bounded Latency) *Is the considered networking technology capable of providing bounded latency for one or several data exchanges ?*

This criteria is used to identify potential candidates that would not be able to provide performance quality of service to data exchanges.

Criteria 3 (Very Low Jitter) *Is the considered networking technology capable of ensuring less than 1 μ s reception jitter for one or several data exchanges ?*

This criteria is used to identify potential candidates that would not be able to handle the low jitter command and control traffic from the platform network.

Criteria 4 (Mixed Criticality) *Is the considered networking technology capable of handling in the same medium platform traffic (with bounded latency and very low jitter constraints) and payload traffic (with high throughput demand) without these two types of traffic affecting one another ?*

6.2.2 Criteria for Application Level Property 2

Three criteria are identified for property 2 *Time Management: Time Synchronisation at MAC Level, Time Management Algorithm Robustness and Interaction with higher layer capabilities.*

Criteria 5 (Time Synchronization at MAC Level) *Does the technology provides a synchronization protocol at MAC Level ?*

This criteria is used to specify that the network shall be in charge of time synchronization in the satellite system.

Criteria 6 (Time Management Algorithm Robustness) *Does the Time Synchronization/Management Algorithm come with any robustness mechanism ?*

This criteria is anticipating the high criticality of the time synchronization function. If most network operations are driven by a timing information known across the whole network, the mechanism in charge of distributing and maintaining such information shall be fail-safe.

Criteria 7 (Interaction with higher layer capabilities) *Does the Time Synchronization/Management Algorithm come with standardized way to interact with applications, on both L2 to L7 and L7 to L2 directions?*

This criteria is related to the first criteria for Application Level Property 2: if the network is in charge of time synchronization in the satellite system (and synchronization happens at L2 level), it would be nice to have a standardized way to share this timing information with applications running over the network. For instance, this timing information could be used to trigger actions at application level.

Remark 12 *We take the hypothesis in this contribution that Time Management should be done at MAC level (ISO level 2). In effect, a higher level management of time could also be considered in the next-generation network.*

6.2.3 Criteria for Application Level Property 3

Before defining the four criteria associated with Application Level Property 3 - *Fault Tolerant Operations* i.e. *Error Detection Capabilities, Error Reporting Capabilities, Redundancy Capabilities and Fault Containment Capabilities*, let us first remind the reader that we consider a faulty behaviour as either incorrect, lost, out of time constraints or out of traffic contracts.

Criteria 8 (Error Detection Capabilities) *Is the technology able to detect the following errors:*

- *Incorrect Message?*
- *Lost Message?*
- *Out of Time Constraints Message?*
- *Out of Traffic Contract Message?*

This criteria serves at identifying the capability of candidate technologies of detecting incorrect messages (e.g. messages with incorrect checksums), lost messages, messages arriving too early or too late (e.g. a flow emitting messages out of its allocated time slot) or messages out of traffic contract (e.g. a flow emitting messages that exceeds its bandwidth reservation).

Criteria 9 (Error Reporting Capabilities) *Is the technology able to report the errors that it has previously identified, in either a direct and an indirect way ?*

The next generation network is expected to be able to report, in both either a direct manner with for instance interruptions or in an indirect manner with for instance statistics counter periodically read by a fault management entity, the faults that it has detected.

Criteria 10 (Redundancy Capabilities) *Does the technology provide a redundancy mechanism?*

Redundancy was identified by the industrial as the way to satisfy the safety requirement i.e. tolerance to the loss of a message. It is therefore expected that the next-generation technology provides such capability.

Criteria 11 (Fault Containment Capabilities) *Is the technology able to isolate/contain and even eliminate the errors it has detected ?*

This behaviour is classical in industrial real-time network. The goal is to prevent the fault from propagating into the network and affecting the nominal behaviour of other devices/applications using the network. For instance, a switch connected to a *babbling-idiot* device (i.e. a device constantly sending messages out of its traffic contract) should eliminate the faulty messages (or all the messages) coming from that device to prevent them to propagate into the network and overcrowd the links and the buffers.

Conclusion

In this chapter, we have first presented the different initiatives for the selection of a next-generation network from the state of the art both in the space domain and outside the space domain. It is a hot topic growing in importance with the emergence of Time Sensitive Networking. Inspired from several papers, we have conceptualized several criteria for each of the application level properties defined in Chapter 8. These criteria could have been more detailed as it was done by the authors of [2]. However, it was deemed sufficient with respect to the satellite use case we were dealing with. These criteria will be used in the comparison of the next chapter in order to select one or several technologies for the next-generation satellite network.

Chapter 7

Qualitative Comparison of the Pre-Selected Technologies with respect to the Identified Criteria

This chapter is the first contribution. We apply the methodology so as to compare the preselected candidates and then detail the output resulting from this methodology. In fact, we propose a qualitative comparison based on a set of high level criteria representative of the requirements of future satellite systems. In the previous chapter, we have defined our set of criteria for each Application-Level Property (cf. Chapter 5). Therefore, we can now proceed with the comparison. In fact, the goal of this comparison is to assess the theoretical capabilities of the technologies i.e. what is provided in their definition documents by their mechanisms and not what would be achievable with a clever use of these mechanisms. This chapter starts by discussing the suitability of Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking and Spacefibre with respect to space requirements. Then, in a second section, suitable candidates are identified. Finally, a last section discusses third-party arguments for the selection of a next-generation satellite network among the suitable candidates.

7.1 Technologies Capabilities w.r.t. space requirements

In the following paragraphs, the symbol  signifies that the technology is compatible with the considered criteria,  that it is not compatible and  that part of it is compatible.

7.1.1 Ethernet

Let us now discuss the compatibility of Ethernet with the three application-level properties of Section 5.3.

Application-Level Property 1 - Mixed Traffic Types

Criteria 1 - High Data Rate  Ethernet is widespread both at home and in ISP¹ core networks. Several physical media are available (e.g. twisted pair, optical, ...) and allow to achieve

¹Internet Service Provider

data rates from 10Mbit/s to 100+Gbit/s. Therefore, Ethernet, on criteria of *High Data Rate*, seems to be suitable.

Criteria 2 - Bounded Latency ✘ To provide some sort of quality of service (maybe bounded latencies?), Ethernet is equipped with only one mechanism: Static Priority.

Definition 37 (Static Priority) *Static Priority relies on the 802.1Q optional tag of the Ethernet frame (cf. Fig. 3.9). This tag contains a 3 bit field called Priority, that can be used to define, at MAC level, a priority between frames on 8 (2^3) levels. This priority is used to decide, in case of medium access conflict, which Ethernet Frame shall be emitted first (i.e. the one with the highest priority).*

Property 2 (Static Priority and delay guarantees) *Static priority does not provide any guarantee on delays as far as there are no traffic contracts in place [13].*

Since Ethernet does not provide any traffic contract by itself (i.e. at MAC level), in order to provide guarantees, traffic contracts are left to be dealt with at application level. Therefore, at MAC level, Ethernet alone cannot provide any guarantees on latencies hence it does not satisfies the *Bounded Latency* criteria.

Criteria 3 - Very Low Jitter ✘ Jitter is understood as latency variability. Let us roughly analyse the different parts that affect latency to see which part will be taken into account in jitter. Any constant element in the latency computation can be ignored since it will by definition no vary between two frames. The only elements that need to be taken into account are the variable elements.

$$Lat_{f_i} = AppEmOffset + \sum_{link \in Path(f)} \Delta_{link} \quad (7.1)$$

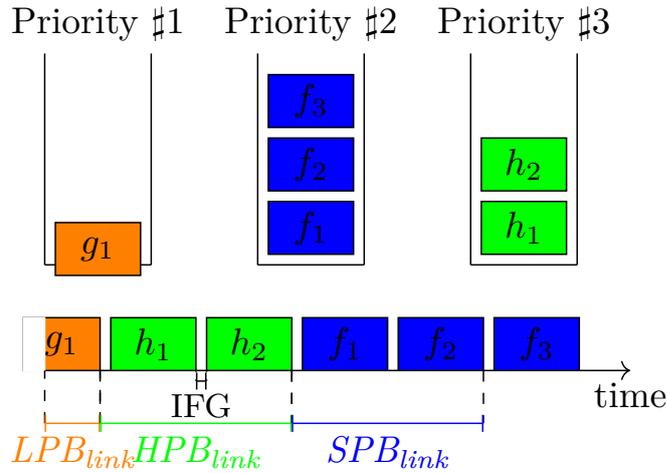
Where $\Delta_{link} = HPB_{link} + SPB_{link} + LPB_{link} + \tau_{emit} + \tau_{propag}$

Where:

- *AppEmOffset* represents the duration between the reference date of f_i and its deposit in the queue of the emitting end-system i.e. $T_{SAP}(f_i) - Ref(f_i)$.
- HPB_{link} , SPB_{link} and LPB_{link} represent the delays induced per hop, in the path of f_i , by higher, same and lower priority traffic.
- τ_{emit} and τ_{propag} represent the duration of emission and propagation of f_i on a link.

Hereafter, we focus on the variability of the delay at emission only to explain why Ethernet is not able to provide very low reception jitter, but this issue may appear on every hop within an Ethernet network, not just on the first hop. With several frames of different priorities competing for the access to the medium, some additional and variable delay can occur.

Example 9 (Jitter issue with Ethernet) *Let us illustrate the variability with an example represented in Fig. 7.1. A frame f_3 of medium priority can be delayed by the a frame g_1 of lower priority finishing being emitted, for a duration LPB_{link} , then by one or several frames (h_1, h_2) of higher priority also waiting to be emitted, for a duration HPB_{link} , and finally by frames of same priority (f_1, f_2) that arrived before f_3 , for a duration SPB_{link} . Depending on the periodicity of all these frames, the waiting delay for the emission of f_3 can be variable. Let us now assume that this frame has low jitter requirements. In the worst case, the lower priority blocking alone adds up to an additional 12,304 microseconds (see Table 7.1) delay, way above the 1 microsecond requirement of a low jitter flow.*

Figure 7.1: LPB_{link} , SPB_{link} and HPB_{link} example

Therefore, only one priority blocking at emission happening for one message and not for the next one of the same flow is enough to make this flow miss its jitter requirement. And this blocking (plus the other ones) could happen on all hops ! Therefore, Ethernet, on *Very Low Jitter* criteria, does not appear to be suitable.

Remark 13 *One might say that not assigning the highest priority to the low jitter flow is not correct. However, doing so is not always a good idea either. Indeed, if there is enough low jitter traffic in the highest priority queue, it could prevent the lower priority traffic being emitted (i.e. starvation) and hence prevent them from satisfying the bounded latency requirements.*

Table 7.1: Usual frame emission delay at 100Mbits/s and 1Gbits/s

	100Mbits/s	1Gbits/s
Max. size frame (1518 + 20 bytes) ²	123,04 μ s	12,304 μ s
Min. size frame (64 + 20 bytes)	6,72 μ s	0,672 μ s

Criteria 4 - Mixed Criticality ✖ Regarding Mixed Criticality, Ethernet proposes with static priority a way to differentiate the importance of certain flows from others on eight levels. In order for platform and payload traffic to coexist, some priorities could be given to platform flows and the rest to payload flows. However, as already explained above, these priorities shall be cleverly allocated so as to avoid the medium being monopolized by the highest priority level. For instance, since payload traffic requires a lot of bandwidth it might not be a good idea to give it a too high priority otherwise the platform traffic could suffer from starvation. Nevertheless, as explained in the *Very Low Jitter* criteria paragraph, it would only require one low priority payload frame to mess up with the transmission delay of a higher priority low jitter frame. Therefore, Ethernet is not suitable with this Mixed Criticality Criteria.

²The 20 bytes correspond to the sizes of Preamble, Frame Delimiter and Inter-Frame Gap, commonly denoted SFD + IFG

To conclude on Application-Level Property 1, Ethernet is not deemed suitable.

Application-Level Property 2 - Time Management

Criteria 5 - Time Synchronization at MAC level ✘ At ISO level 2, Ethernet alone does not provide any mechanisms for time management. However, the use of higher level protocols over Ethernet for time distribution and/or time synchronization is very common. These protocols often require a lower level layer support, at either MAC or PHY level. Hardware supporting these protocols shall be able to timestamp frames in emission and reception and retrieve these timing informations at higher layer for these protocols to use. The most mainstream synchronization protocol over Ethernet (but not at level 2) is PTP - *Precision Time Protocol* [55]. However, this is out of scope of this study. Therefore, Ethernet does not seem to satisfy this criteria.

Criteria 6 - Time Management Algorithm Robustness ✘ There is no Time Management Algorithm in Ethernet therefore discussing of its robustness is not possible. Hence Ethernet does not satisfy this second criteria.

Criteria 7 - Interaction with Higher Layer ✘ There is no Time Management Algorithm in Ethernet therefore discussing of its interactions with higher layers is not possible. Hence Ethernet does not satisfy this third criteria.

On behalf of the suitability with the above criteria, we conclude that Ethernet does not satisfy Application-Level Property 2.

Application-Level Property 3 - Fault Tolerant Operations

Criteria 8 - Error Detection ✘ Regarding error detection, Ethernet only offers one tool: Ethernet frame CRC i.e. the *FCS - Frame Checking Sequence*. This 16 bits field allows checking the integrity of the content of the Ethernet frame. A CRC error usually leads to the erroneous frame being dropped, which coincides with the fault isolation criteria. This CRC will help detecting and preventing the incorrect faulty behaviour. There is no way of detecting the other types of faults (i.e. lost message, out of time constraint and out of traffic contract).

Criteria 9 - Error Reporting ✔ In addition to the Ethernet frame CRC, Ethernet devices usually hold counters, called MIB - *Management Information Base* counters, that describe the behaviour of the device through the number of received frames, number of emitted frames, number of CRC errors, etc. These MIBs are, for most parts, standardized by IEEE (e.g. MIB section in IEEE 802.3 [58] and IEEE 802.1Q [56]) or IETF (e.g. RFC 3635). The information included in these MIBs, updated in every device, could be gathered by a higher layer entity (with the help of SNMP protocol for instance) and serve as error detection mechanism. MIBs will help higher layer fault management entities detect incorrect, lost and out of traffic contracts faulty behaviours.

Although Ethernet has some error detection mechanisms, there is no real mean to prevent faults, in particular, lost faulty behaviour (like single points of failure). Moreover, there are no mechanisms available to detect and or prevent the out of time constraints faulty behaviour.

Criteria 10 - Redundancy ✘ In Ethernet, at ISO level 2, there are absolutely no mechanisms for redundancy. However, Ethernet, is widespread and hence is used as MAC layer for several higher layer protocols like, for instance, UDP/IP. Although Ethernet does not provide any redundancy

mechanisms or more error handling mechanisms, such mechanisms could be provided with higher level protocols. Nevertheless, this option is out of scope of this comparison.

Criteria 11 - Fault Containment ✘ Regarding fault containment capabilities, Ethernet provides weak traffic segregation through Static Priority. In fact, if a device sends too many messages (i.e. out of traffic contract faulty behaviour e.g. babbling idiot) which corresponds to not in traffic contracts faulty behaviour, it will definitely affect the available bandwidth of the other flows and might lead to buffer overflow. Moreover, this faulty behaviour will propagate downstream.

According to the capabilities introduced above, Ethernet technology is not effective enough w.r.t Application-Level Property 3.

7.1.2 ARINC 664

Let us now do the same discussion for ARINC 664.

Application-Level Property 1 - Mixed Traffic Types

Criteria 1 - High Data Rate ✔ ARINC 664 was first implemented at 100Mbit/s. It was later improved to run at 1Gbit/s. Therefore, ARINC 664 is compatible with the first criteria of Property 1.

Criteria 2 - Bounded Latency ✔ Ethernet did not provide any guarantees on latencies by itself. Nevertheless, ARINC 664 extends Ethernet with bounded latency capabilities. In fact, in addition to Static Priority (reduced to only two level and only present in switches), the concept of VL - *Virtual Link* is introduced in ARINC 664. It is indeed a reserved bandwidth for a specific traffic on a static route. This *traffic contract* is characterized by two parameters, the maximum frame size and the so-called BAG - *Bandwidth Allocation Gap* - i.e. the minimum time between two frames' emission in the same VL. By applying Property 2, there can be guarantees on latencies in ARINC 664 since it uses Static Priority and has traffic contracts (i.e. VLs). In fact, latencies can be determined with more or less pessimism through, for instance, network calculus. Typical latencies are in the range of 1 to 10 milliseconds [14]. Therefore, ARINC 664 satisfies the *Bounded Latency* criteria.

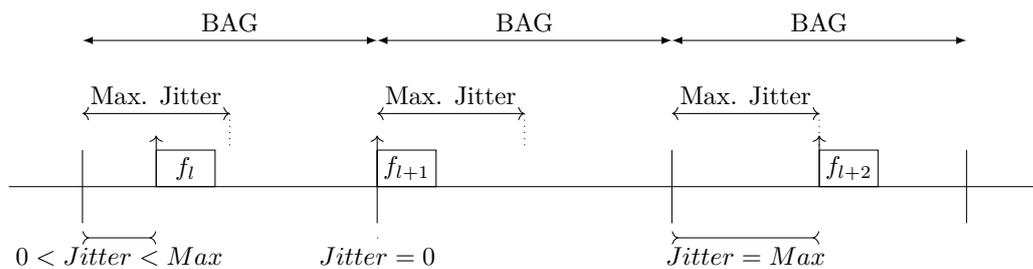


Figure 7.2: BAG concept in AFDX and jitter illustration [3]

Criteria 3 - Very Low Jitter ✘ While Ethernet relied on static priority for medium access in the whole network, ARINC 664 provide a scheduler to arbitrate between VLs competing to access the medium in any emitting end-station. This scheduler can introduce emission jitter, as represented

in Fig. 7.2. This jitter is bounded and its maximum value is known as *maximal admissible jitter* in the standard [3]. This maximum upper bound of the jitter is $500\mu\text{s}$ but the standard indicates that a typical value for this jitter is $40\mu\text{s}$. Virtual links were designed for guaranteed latencies and controlled jitter. However, they were not designed to provide very low jitter ($1\mu\text{s}$ i.e. one or two orders of magnitude lower than the bounds mentioned in the standard). Therefore, ARINC 664 does not satisfy the very low jitter criteria.

Criteria 4 - Mixed Criticality ✘ While the above paragraphs have discussed the handling of platform traffic with deadline and jitter constraints, nothing has been said regarding the capability to also handle high throughput traffic. In fact, in ARINC 664 the payload traffic (e.g. a video traffic coming from the navigation camera) could either be fitted in a specific VL or be standard Ethernet traffic [52]. Nevertheless, fitting it in one or several VL could still affect the nominal behaviour of low jitter platform traffic as explained in the previous criteria, therefore, as is, ARINC 664 is not compatible with the Mixed Criticality criteria.

So far, ARINC 664 does not seem suitable with respect to Application-Level Property 1.

Application-Level Property 2 - Time Management

Regarding synchronization, ARINC 664 is identical to Ethernet and our conclusion is not changed. ARINC 664 is not suitable with Application-Level Property 2.

Criteria 5 - Time Synchronization at MAC level ✘ There is no Time Synchronization mechanism defined in the ARINC 664 standard.

Criteria 6 - Time Management Algorithm Robustness ✘ There is no Time Synchronization mechanism defined in the ARINC 664 standard. However, if such algorithm existed, the synchronization traffic could be given a specific VL. It would hence be provided with a guaranteed reserved bandwidth and ARINC 664 fault tolerant operation mechanisms would prevent impact from synchronization traffic to user traffic latencies and respectively. Nevertheless, ARINC 664 is not suitable with criteria Time Management Algorithm Robustness.

Criteria 7 - Integration with High Layers ✘ There is no Time Synchronization mechanism defined in the ARINC 664 standard therefore we do not discuss this criteria.

Application-Level Property 3 - Fault Tolerant Operations

ARINC 664 offers another major improvement from Ethernet: it enhances Ethernet with fault tolerance and traffic policing capabilities.

Criteria 8 - Error Detection ✘ In addition to CRC, ARINC 664 switches have a traffic policing feature. It allows the switch to verify that every VL respects its traffic contract. If not, flows that exceeds their contract are dropped. This will help preventing the out of traffic contract faulty behaviour, but will also serve for fault containment purposes. In fact, if a device is emitting more than it should or emitting with incorrect addressing (wrong destination address), its traffic will immediately be eliminated at the next hop and will not impact the nominal behaviour of other VLs. Although traffic policing offers a good handling of out of traffic contracts faulty behaviour, it does not detect nor prevent the out of time constraints faulty behaviour. Therefore, ARINC 664 is completely not compatible with this first criteria.

Criteria 9 - Error Reporting ✓ The new mechanisms (i.e. switch traffic policing and redundancy) introduced in ARINC 664 are also represented in the MIB counters. With respect to Error Reporting, ARINC 664 is deemed suitable.

Criteria 10 - Redundancy ✓ ARINC 664 uses an end-to-end redundancy protocol at level 2. This redundancy protocol will help preventing the lost faulty behaviour. It can be seamless, meaning that the application level is not aware that a redundancy protocol was running at level 2 and does not need this information to work nominally. Let us rapidly describe this redundancy protocol and what it offers. Redundancy in ARINC 664 allows the duplication of messages of a flow at the emitting end-point and their reassembly (elimination of the duplicate) at the receiving end-point. Duplicates travel on the same path but on two different channels i.e. two different cables. Duplicates are identified by a two bytes *sequence number* added to the Ethernet frame right before the frame's FCS. The sequence number is used for two functions i.e. *Integrity Checking* and *Redundancy Management*. Integrity Checking is a fault detection and isolation mechanism that helps to detect and eliminate faulty duplicates (incorrect faulty behaviour) as well as lost messages (lost faulty behaviour) whereas Redundancy Management eliminates duplicated messages once the nominal message has been received (lost faulty behaviour). These two functions are illustrated in Fig. 7.3.

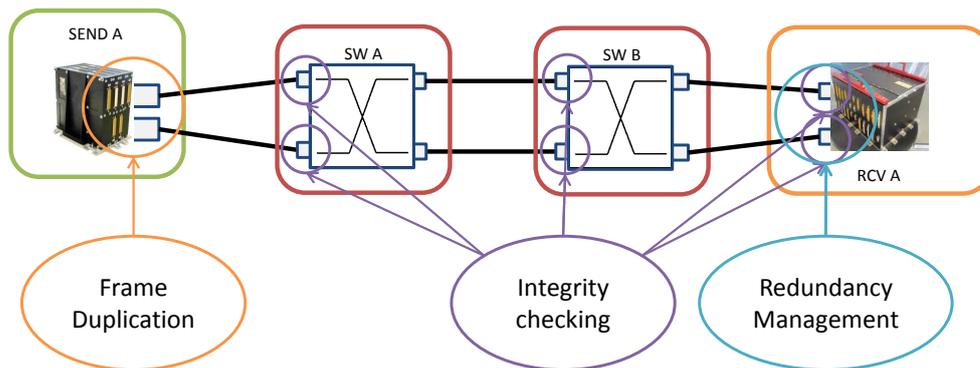


Figure 7.3: AFDX Redundancy Example

Example 10 AFDX Redundancy Example In the emitting device, a frame is associated with a sequence number and then duplicated, one duplicate is sent on the upper path and the other is sent in the lower path. When a frame reaches the input port of a switch, it is checked by the Integrity Checking function which verifies that the integrity of the frame (CRC, length, etc.) and then passed to the switching core. Finally, when frames reach the receiving devices, they are first checked by the Integrity Checking function before being handled by the Redundancy Management Function. When the first duplicate is received, if its sequence number is correct (in the sequence of the previous one received), it is passed to the application. When the second duplicate is received, its sequence number is checked and the frame is dropped since it has already been passed to the application. In the case where one of the duplicate is not received, the other is still passed to the application and the message is not lost. After a time, the missing frame will be marked in the MIB counters and reported to higher layer fault management.

As a conclusion, ARINC 664 is suitable with respect to this criteria.

Criteria 11 - Fault Containment ✘ Regarding fault containment, VL and traffic policing offer a traffic segregation of all the different flows. In addition, Integrity Checking and Redundancy Management functions will also prevent the propagation of faults into the network (e.g. elimination of unwanted duplicated, elimination of messages with wrong sequence number, elimination of faulty frames, etc.). Any fault occurring in one VL will not affect the nominal behaviour of the other VLs.

According to the previous mechanisms, although ARINC 664 improves Ethernet with fault tolerance capabilities, it is still deemed unsuitable w.r.t. to Application-Level Property 3.

7.1.3 TTEthernet

Let us now shift our focus on TTEthernet.

Application-Level Property 1 - Mixed Traffic Types

Criteria 1 - High Data Rate ✔ TTEthernet proposes data rates ranging from 100Mbit/s to 1+Gbit/s, therefore satisfying this first criteria.

Criteria 2 - Bounded Latency ✔ TTEthernet proposes the same types of traffic than ARINC 664 i.e. BE -*Best Effort* (standard Ethernet) and RC-*Rate Constrained* (ARINC 664) traffic, hence it benefits from all the properties that we described for ARINC 664 in particular the bounded latency capability for the RC traffic. Moreover, TTEthernet extends ARINC 664 even further. In addition to BE and RC traffic, TTEthernet introduces a third traffic type entitled TT - *Time Triggered*. Time Triggered traffic is sent in a time triggered manner. Each TTEthernet device (i.e. end-stations and switches) has a transmit schedule per flow. This schedule allows flows to achieve constant communication latency. Therefore, TTEthernet is compliant with the *Bounded Latency* criteria.

Criteria 3 - Very Low Jitter ✔ TT traffic in TTEthernet solves the jitter issue that Ethernet and ARINC 664 faced. In fact, since TT traffic achieves constant communication latency and the schedule (especially in switches) ensures there's no blocking from other frames, equation (7.1) leads to the conclusion that the low jitter on reception problem roots to a low jitter on emission problem (i.e. *AppEmOffset* which is variable). However, thanks to our definition of latency (i.e. defined between a reference date and the reception date), the per-flow schedule solves the problem. It does not matter how variable the deposit date of a frame is as long as it happens before the frame's schedule. We illustrate how time-triggered schedule cancels application emission jitter in Fig. 7.4. In fact, any variability in the deposit date will be compensated by the constant emission date and then constant communication latency will ensure a very low jitter for that frame. The only elements taking part into jitter in this situation are clock precision and TTEthernet constraint stating that in any schedule, there shall be space to fit a PCF - *Protocol Control Frame* - frame. The size of this PCF is 84 bytes (including SFD and IFG), which means that it can lead to a jitter of at most 672 nanoseconds (cf. Fig. 7.1). This jitter is compatible with the 1 microsecond requirement of platform traffic and hence TTEthernet ticks the low jitter criteria box.

Criteria 4 - Mixed Criticality ✔ In order to mix platform and payload traffic in TTEthernet, 2 solutions exist. In both solutions, the platform traffic is put into TT traffic. Then, the payload traffic can either be fitted into VLs traffic or TT traffic. In both cases, the use of the TT schedule

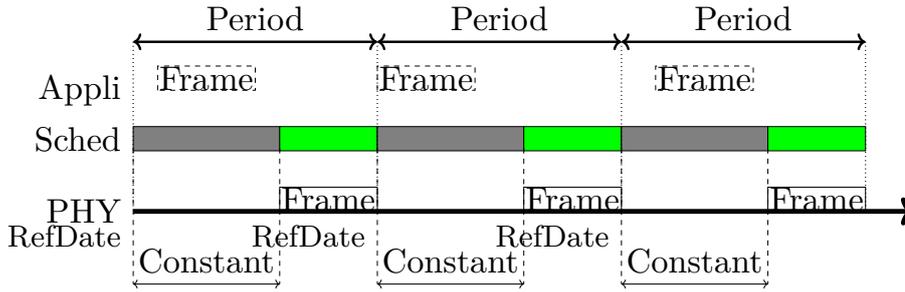


Figure 7.4: TT Schedules cancels application emission jitter

will prevent one traffic interfering with another as well as providing the right level of Quality of Service therefore satisfying the Mixed Criticality criteria.

According to our analysis, TTEthernet appears to be a relevant candidate with respect to Application-Level Property 1.

Application-Level Property 2 - Time Management

Criteria 5 - Time Synchronization at MAC level ✓ TTEthernet proposes a protocol to establish and maintain a global time throughout the network. It is realized by synchronization of local clocks within the devices (i.e. end-point and switches). It works in a similar manner than IEEE 1588 (also known as PTP). In fact, time master devices (entitled *synchronization masters*) distributes time with broadcast messages (entitled "PCF" frames in the TTEthernet standard). Devices in the system gather these PCF frames use their information to correct their local clock. In order for PCF to be correctly received, every slot (for the TT traffic) is configured so that both the frame of this slot and a PCF frame can be emitted. Hence, there is a resource reservation of 84 bytes (frame size + SFD + IFG) per slot for time synchronization. Therefore, TTEthernet satisfies the *Time Synchronization at MAC level* criteria.

Criteria 6 - Time Management Algorithm Robustness ✓ In addition, TTEthernet introduces a new mechanism targeted for fault tolerance i.e. clock redundancy. Clock redundancy is illustrated in Fig. 7.5. In fact, in TTEthernet, there can be several synchronization masters (instead of one in PTP). These *masters* send their time through PCF frames in the network. Specific devices entitled *compression masters* gather PCFs coming from masters and vote for the correct timing information ("Step 1" in Fig. 7.5), this correct information is then distributed through a new PCF towards network devices for synchronization. The PCF is also sent back to synchronization masters for fault detection ("Step 2" in Fig. 7.5). This allows to tolerate the loss or the incorrect behaviour of one or several synchronization masters.

This mechanism is really important since synchronization is critical in TTEthernet. Without synchronization, the proper temporal property obtained for TT flows (in time slots), i.e. very low jitter and constant network latency cannot be guaranteed anymore.

With the clock redundancy mechanism, TTEthernet satisfies this criteria.

Criteria 7 - Interaction with higher layers ✗ There is no interface for interaction with higher layer specified in the TTEthernet standard. Hence, TTEthernet does not match this last criteria regarding Property 2.

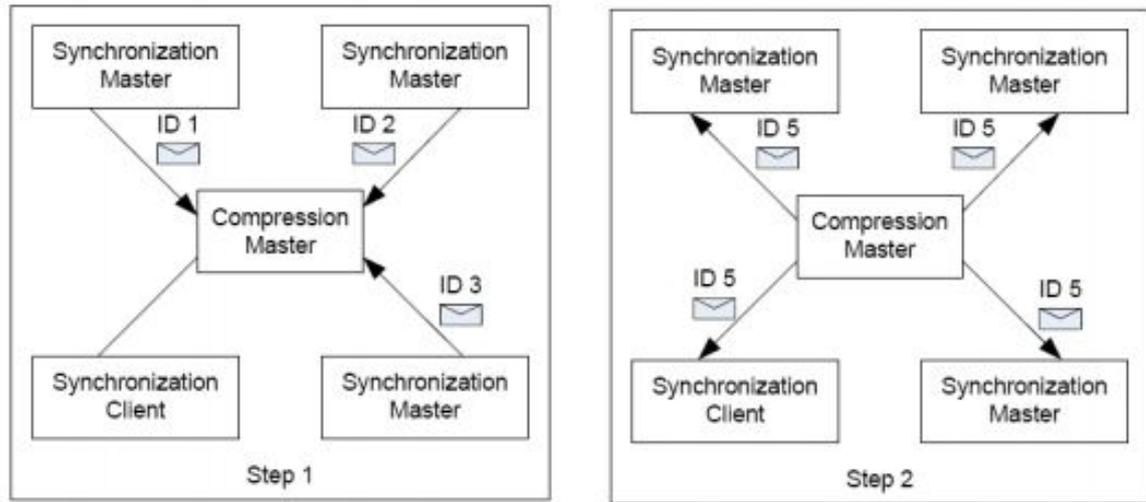


Figure 7.5: TTEthernet two-step synchronization (cf. [119])

To conclude, although there are no interfaces specified in the standard, some interfaces exist in the implementations. Since TTEthernet has a specified synchronization algorithm and that implementations propose such interaction capabilities, we ignore the last criteria for our conclusion. Therefore, we consider TTEthernet relevant w.r.t. Application-Level Property 2.

Application-Level Property 3 - Fault Tolerant Operations

Criteria 8, 9, 10, 11 - Error Detection, Error Reporting, Redundancy and Fault Containment ✓ Regarding fault tolerance, TTEthernet provides both redundancy and policing capabilities inherited from ARINC 664 (cf. Annex D of SAE AS6802 [102]). TTEthernet network takes back the redundancy management and the integrity checking of ARINC 664 for TT and RC traffic. TTEthernet switches can also implement a *central bus guardian* function. It basically provides a policing mechanism to check that traffic contracts (of RC traffic) and schedule (of TT traffic) are respected. This function serves the identification, elimination and containment of the out of traffic contracts and out of time constraints faulty behaviours. In fact, it can for instance prevent the propagation of faults coming from a babbling idiot device in the network. Therefore, TTEthernet is able to detect, report and contain all types of faults identified in the criteria for Application-Level Property 3.

We conclude that TTEthernet is suitable w.r.t. Application-Level Property 3.

7.1.4 Time Sensitive Networking

Let us introduce Time Sensitive Networking capabilities.

Application-Level Property 1 - Mixed Traffic Types

Criteria 1 - High Data Rate ✓ Time Sensitive Networking, still under development by the TSN Task Group, is the successor of AVB. The available implementations of TSN range from

100Mbit/s to 10Gbit/s, and higher data rates will probably be available as TSN gets more mature. Therefore, TSN meets the high data rate criteria.

Criteria 2, 3 - Bounded Latency and Very Low Jitter ✓ Time Sensitive Networking extends Ethernet with traffic shaping capabilities and medium access enhancement. First of all, TSN reuses the static priority mechanism of Ethernet. However, TSN introduces a new mechanism entitled *Frame Preemption*, introduced in [59]. Frame Preemption helps, depending on configuration, to solve the Static Priority jitter introduced by lower priority blocking. In fact 802.1Qbu and 802.3br allow purposely tagged frames (*express*) to suspend the transmission of other frames (*preemptable*) for their own transmission on a point-to-point link, defining a frame fragmentation similar to IP fragmentation. Using Frame Preemption, the lower priority blocking jitter is reduced to the necessary time to transmit, in the worst case, a 143 bytes long frame (see [116]). At 1Gbit/s, this will lead to a $1.144\mu\text{s}$ jitter which is nearly compatible with the very low jitter criteria.

In addition, TSN proposes the Time Aware Shaper mechanism which allows defining time windows in which frames can be emitted, almost in a similar manner than TTEthernet. However, contrary to TTEthernet, the time schedule is not applied to flows (named *Streams* in TSN) but to emission queues. This means that the good temporal properties obtained in TTEthernet are not immediately achievable here. In particular, in the network, there may be more than 8 streams, meaning that several streams would have to share the same queue hence risking additional communication delay from non-exclusive resources.

One solution to do so would be to reproduce TTEthernet with TSN mechanism Time Aware Shaper. Examples of such solutions are presented in [23], [25] and [24]. It helps achieving a per-flow schedule in TSN network. We will propose another solution in our second contribution. In conclusion, Time Sensitive Networking satisfies the bounded latency and very low jitter criteria.

Criteria 4 - Mixed Criticality ✓ After the discussions on bounded latency and very low jitter, let us focus on mixed criticality. As for ARINC 664 and TTEthernet, TSN proposes a way to share bandwidth between flows and prevent any starvation issues related to the use of Static Priority. In fact, TSN inherits the CBS mechanism introduced in AVB. CBS or *Credit Based Shaper* defines a rule to allocate a bandwidth to a queue (ex: 4Mbits/s for queue 1) based on a credit that evolves when frames are enqueued or dequeued. With this credit, a maximum bandwidth can be defined for a set of flows sharing a queue. In addition, the TSN Enhanced Transmission Selection mechanism adds additional opportunities for sharing the available bandwidth between queues of an output port. One algorithm for ETS can be Round Robin or Weighted Round Robin.

Using the Time Aware Shaper for platform traffic and CBS or ETS for payload traffic should be a good way to have both type of traffic coexisting and meeting their quality of service requirements, therefore satisfying this criteria.

We hence declare TSN suitable w.r.t. Application-Level Property 1.

Application-Level Property 2 - Time Management

Criteria 5 - Time Synchronization at MAC level ✓ Inherited from Ethernet, Time Sensitive Networking supports several synchronization protocols. In particular, it supports IEEE 1588 (PTP) but also IEEE 802.1AS [57] also called gPTP. However, PTP and gPTP are not MAC level only protocols. Therefore it does not really entirely comply with this first criteria.

PTP (i.e. IEEE 1588) is quite a simple protocol, one time master device (entitled *grandmaster*) distributes time with broadcast messages. Devices in the system measure propagation delays with their peers or with the time master and use this information to correct the time received from

the master. If several potential grandmasters exist in the network, IEEE 1588 introduces the *Best Master Clock Algorithm - BMCA*, which chooses which device among potential masters is going to be grandmaster. This algorithm allows for faster recovery than PTP since the BMCA is run permanently and triggered if the current grandmaster is not functioning anymore in which case a new master can hence be automatically elected.

gPTP is, for most part, fairly similar or identical to PTP. It contains in fact a profile of 1588 for use in TSN networks. We will not explain how the synchronization is established. Instead, let us focus on the two main events with gPTP : the possibility for a network to have several synchronization masters and the specification of interfaces and primitives for *"time sensitive applications"*.

In gPTP, like in PTP and in TTEthernet, the synchronization traffic travels in the same network than user data. However, it does not requires too stringent resources reservation: [80] shows that the synchronization process based on 802.1AS is not affected by high network load, while not having a dedicated queue or a very high priority.

Criteria 6 - Time Management Algorithm Robustness ✓ Like TTEthernet synchronization, TSN synchronization protocol offers the possibility to have several masters in the network for availability purposes. However, it does not work quite like TTEthernet. In fact, 802.1AS standard does not provide any consolidation strategy for several grandmasters running concurrently (realized with compression masters in TTEthernet), it is left to be developed at application level.

gPTP does not provide any fault tolerance mechanisms apart from BMCA (cf. §7.2.4.3 of [57]) which covers a grandmaster not working. As stated in the standard: *"Techniques for identifying other types of failures, and the appropriate correction necessary, are not specified in this standard. However, if other techniques or standards are used for detection and correction of these (...) failures, this standard provides the means to recover from these errors"*. It will be the responsibility of system engineers designer the higher layers to provide mechanisms for fault detection, isolation and recovery in the synchronization system. Therefore, again, TSN does not entirely comply with this second criteria.

Criteria 7 - Higher Layer Interactions ✓ gPTP also introduces application interfaces (cf. Clause 9 of [57]) for use with time sensitive applications. These interfaces are *"model of behaviour and not application program interfaces"*. Five interfaces are described in the standard but it is stated that others can exist. Hopefully, this will help facilitate the port of *time sensitive applications* on different TSN products from potentially different manufacturers. Hence, TSN is deemed suitable with this third criteria.

To conclude, we consider TSN relevant w.r.t. Application-Level Property 2.

Application-Level Property 3 - Fault Tolerant Operations

Criteria 8, 9 - Error Detection and Error Reporting ✓ Time Sensitive Networking offers a wide range of mechanisms for Fault Tolerant Operations. Since it extends Ethernet, all elements introduced in Ethernet (CRC and MIBs) are still true for TSN. In terms of traffic policing, IEEE 802.1Qci - *Per Stream Filtering and Policing*[54] offers multiple mechanisms. One serves at detecting any temporal error in the reception of frames, this is particularly useful when TSN is configured with time triggered traffic. Another one serves at ensuring, per flow or per queue, the compliance of the traffic with the traffic contracts in place. Indeed, *Per Stream Filtering and Policing* will help detect and prevent the out of contract and out of time constraints faulty behaviours. Therefore TSN complies with this first two criteria.

Criteria 10 - Redundancy ✓ Redundancy in TSN is fairly similar than in ARINC 664. Frames are duplicated and then reassembled using a sequence number. In TSN this sequence number is located in an Ethernet optional field (or tag). Several protocols are available for redundancy such as RTAG, PRP or HSR tags. TSN can deliver frames out of order whereas ARINC 664 provides in order delivery guarantee. However, TSN improves ARINC 664 redundancy by adding the opportunity to have more than two duplicates. It also offers the ability to specify the path of every duplicated flow, meaning that nominal and redundant flows does not have to travel on the same path using two different channels. In addition, in TSN it is possible to do redundancy on only a fragment of the path of the flow instead of only end-to-end. As for ARINC 664 and TTEthernet, redundancy in TSN will help detecting and prevent the incorrect and lost fault behaviour. Therefore TSN complies with the redundancy criteria.

Criteria 11 - Fault Containment ✓ TSN will provide similar levels of fault containment than TTE on the faulty behaviours we have previously identified: faulty frames will be dropped with CRC checking, loss faulty behaviour will be prevented by redundancy, out of traffic contract and out of time constraints traffic will be detected and removed by the Per Stream Filtering and Policing, therefore, in all cases, preventing the propagation of a fault further in the network. However, there is a sort of challenge with fault containment in TSN. In fact, while TSN traffic shaping behaves like Ethernet i.e. the segregation unit is a queue (through Static Priority), the segregation unit for traffic policing is not a queue but a flow since the traffic policing functions are applied per stream (in *Per Stream Filtering and Policing*). This difference of granularity between traffic shaping and traffic policing might increase the complexity of the segregation analysis for fault containment purposes. In any cases, TSN complies with this criteria.

In consideration of the above analysis, we declare TSN suitable w.r.t. Application-Level Property 3.

7.1.5 SpaceFibre

Finally, let us discuss the compatibility of SpaceFibre with respect to the three previously introduced application-level properties.

Application-Level Property 1 - Mixed Traffic Types

Criteria 1 - High Data Rate ✓ Spacefibre probably offers the highest data rate among all our technologies of interest. It can reach more than 40Gbit/s [1]. To do so, the SpaceFibre standard offers the possibility to serialize the communication over up to 16 so-called *lanes* in the same cable (either optical or twisted pairs). This serialization, detailed in [44], is similar to PCI Express serialization over multiple lanes [4].

Criteria 2 - Bounded Latencies ✓ To be able to provide guaranteed latencies, a system must provide both traffic contracts and bandwidth contract (cf. Property 2). In a SpaceFibre network, *SpF Token-Bucket* ensures such a traffic limitation. Nevertheless, to get accurate bounds on latencies, a good worst-case model of the *SpF Scheduler* must exist. Since *SpF Scheduling* combines a per VC FIFO strategy, 16 levels of priority and a credit-based algorithm that looks similar to the CBS and ATS, the analysis methods developed for TSN [82] may certainly be adapted for SpaceFibre.

Criteria 3 - Very Low Jitter ✓ The *SpF Scheduler* is very likely to achieve ultra low jitter ($< 1\mu s$). By configuring the time-slots so that only one VC is allowed to access the medium at any

time, one might expect that no traffic from other VC would interfere and induce unwanted jitter due to non-preemption (see [19] for illustration on non-preemption jitter).

However, this last assertion is not always true. In fact, the SpaceFibre standard does not define any guard band mechanism for scheduler. This means that if a VC starts to emit one frame just before the end of its time-slot, that emission will end during the time-slot of the next VC and therefore delaying the next scheduled emission. The maximum induced jitter in this situation is the transmission duration of a frame (256 bytes). If the SpaceFibre network operates at 1Gbit/s, the induced jitter value is $2\mu s$ per hop, leading the ultra low jitter constraints not being satisfied. Nevertheless, if the data rate increases, it will directly reduce the induced jitter value and the jitter constraint is very likely to be satisfied even without guard bands.

Criteria 4 - Mixed Criticality ✓ SpaceFibre will be able to achieve mixed criticality by assigning different slots to platform and payload traffic. Traffic will coexist and should not have an impact on each other quality of service. Therefore, it satisfies this criteria given the data rate is high enough (for very low jitter criteria compliance).

To conclude, given the data rate is high enough, SpaceFibre is deemed suitable with respect to Application-Level Property 1.

Application-Level Property 2 - Time Management

Criteria 5 - Time Synchronization at MAC level ✓ Regarding synchronization and time distribution, SpaceFibre offers the possibility to rely on SpaceWire time-codes to distribute time information across the network. To do so, the time-code packet is sent to all network devices with the help of broadcast data frames.

Criteria 6 - Time Management Algorithm Robustness ✗ There is no dedicated robustness mechanism specified for this time distribution method. However, the broadcast frames travel in a separate channel (broadcast channel) than user data (virtual channels). As explained in next section, this will provide space and time isolation between broadcast frames (including ones bearing time-codes) and user data. In addition, broadcast messages will also rely on the same fault tolerance mechanisms than data frame (see next section).

Criteria 7 - Higher Layer Interaction ✓ Finally, in order to support a network-application synchronization, SpaceFibre Service Access Points at L2 and L3 can forward a broadcast message indication to the upper OSI layers, which they can use to synchronize themselves with network time.

Although the time distribution mechanism lacks robustness, SpaceFibre is deemed almost suitable with respect to Application-Level Property 2.

Application-Level Property 3 - Fault Tolerant Operations

Criteria 8 - Error Detection ✓ Regarding error detection, the SpaceFibre standard provides three elements. First, a CRC per frame, which allows to detect errors within a frame. Then a sequence number, similar to the one of ARINC 664, allows to detect loss of frames (missing sequence number) or errors in the emitter (several frames with the same sequence number). Finally, the *VC Bandwidth Credit* has a minimum and a maximum value. If the credit reaches any of these bounds, an alert is raised to signify that either the VC uses more bandwidth then reserved or the VC does not get enough bandwidth to meet its traffic contract. The only type of error that SpaceFibre cannot

detect is the out of time error e.g. when a frame belonging to a VC that is not scheduled in the current time-slot is emitted.

Criteria 9 - Error Reporting ✓ All the errors detected by SpaceFibre devices can be reported in both a synchronous and an asynchronous ways. When an error is detected, the data link layer triggers an indication that is passed to the upper layer (in particular the application layer), giving a real-time warning on the error. In addition, when an error occurs, a dedicated MIB - *Management Information Base* - counter is updated and can be monitored later on by another entity of fault management. This MIB is really similar to Ethernet MIBs.

Criteria 10 - Redundancy ✓ SpaceFibre does not offer any redundancy mechanism at Data Link Layer. There is no packet duplication that could travel on disjoint paths whatsoever. However, there is a possibility of having a warm redundancy on the physical medium on a point to point basis. In fact, instead of using the 16 (or less) lanes within a link to increase the link speed, a lane could be in hot standby. In the event of one of the used lane becomes faulty, it would be swapped by the hot standby lane. This would provide a sort of redundancy mechanism at physical level.

Although there is no redundancy, the data link layer in SpaceFibre works in connected mode, meaning that the reception of any frame is acknowledged. In case of erroneous reception, the faulty frame(s) can be retransmitted with the help of a retry mechanism. This behaviour is similar to acknowledgements and retry in TCP protocol [67]. However, this retry mechanism would require further studies as it may affect the performances discussed in Property 1 (e.g. if a frame is retransmitted after the end of the time-slot associated to its VC).

Criteria 11 - Fault Containment ✓ Finally, SpaceFibre proposes solutions to contain the errors detected by a SpaceFibre device. When a frame is received with wrong CRC or wrong sequence number, that frame is deleted therefore preventing it from spreading in the network. That frame can then benefit from the retransmission mechanism. In addition, the token-bucket like mechanism ensures temporal and space isolation between VCs i.e. an out of traffic contract error, as for instance babbling idiot traffic, in one VC will not affect the performances of other VCs. However, to the difference of TSN where out of traffic contract frames are deleted in ingress, out of traffic contract frames in SpaceFibre are not deleted, but reshaped in the next output port so that it fits the resource reservation of the VC. This can therefore induce unwanted increase in the switching fabric workload, and also potentially lead to buffer overflows.

Even with the lack of proper frame replication mechanism, SpaceFibre is deemed suitable with respect to Application-Level Property 3.

7.2 Analysis

Now that the suitability of each technology with respect to Application-Level Properties 1, 2 and 3 has been properly discussed, we summarize the output of the previous sections in the tables below (cf. 7.2, 7.3, 7.3).

7.2.1 Summary of the Comparison

7.2.2 Third-party Arguments for the Selection of an Upgrade Candidate

According to the previous analysis, it seems that three technologies would be good candidates for a future unified satellite on-board networks: SpaceFibre, TTEthernet and Time Sensitive Networking.

Table 7.2: Compliance to Application-Level Property 1 - Mixed QoS

Criteria	Ethernet	ARINC 664	TTEthernet	TSN	SpF
Data Rate	✓	✓	✓	✓	✓
Latency	✗	✓	✓	✓	✓
Jitter	✗	✗	✓	✓	✓
Suitability	✗	✗	✓	✓	✓

Table 7.3: Compliance to Application-Level Property 2 - Time Management

Criteria	Ethernet	ARINC 664	TTEthernet	TSN	SpF
At Layer 2	✗	✗	✓	✓	✓
Robustness	✗	✗	✓	✓	✗
Interaction	✗	✗	✗	✓	✓
Suitability	✗	✗	✓	✓	✓

Table 7.4: Compliance to Application-Level Property 3 - Fault Tolerant Operations

Criteria	Ethernet	ARINC 664	TTEthernet	TSN	SpF
Error Detection	✗	✗	✓	✓	✓
Error Reporting	✓	✓	✓	✓	✓
Redundancy	✗	✓	✓	✓	✓
Fault Containment	✗	✗	✓	✓	✓
Suitability	✗	✗	✓	✓	✓

However, apart from Application Level Properties 1, 2 and 3, several arguments shall also be taken into consideration when choosing between one candidate or another.

First, satellite components have stringent hardware/software constraints, not in certification like in aerospace, but more in radiation, temperature, SEU -*Single Event Upset*- tolerance, etc. This means that the satellite network manufacturer has to either buy end-points and switches designed for space or buy IPs that would be instantiated into space-hardened components. On the one hand, TTEthernet (through TTEch) and SpaceFibre have already been, or are being implemented into several space projects both in Europe and in the USA and are standardized for space use by ESA in an ECSS standard (*European Cooperation for Space Standardization*). It would hence be possible to obtain space-oriented TTEthernet and SpaceFibre components. On the other hand, TSN, for the past years, has gained increasing interest from the automotive industry and automation industry. The TSN devices that would be available on the market would not completely fulfil the space requirements, especially in terms of radiation tolerance. It would however be possible to either buy IPs and instantiate them into space-hardened components or buy the entire COTS and do a radiation tolerance evaluation campaign.

Then, the space community is hoping that the use of COTS components from a widespread technology, shared with other industry verticals would help reducing the overall cost of design, purchase of devices and software development. One drawback of using TTEthernet or SpaceFibre instead of TSN would be that it is only produced and maintained by very limited manufacturers whereas TSN has already dozens of manufacturers working on it. Nevertheless, the products currently advertised by TSN automotive manufacturers might not exactly fit the space needs in terms of performance

or environment tolerance and might require further work before being used in space systems; which in the end might lead to an increase of costs. However, the impact on non-recurring cost might be significant enough to make the use of TSN worth. That is why the definition of a profile (like the TSN Automotive Profile but for space) would be a very good starting point to give space and aerospace an identity towards TSN components manufacturers.

On validation and certification side, there is a certain advantage in using TTEthernet or Time Sensitive Networking instead of SpaceFibre. In fact, both TTEthernet and Time Sensitive Networking are based on Ethernet, where numerous research on validation/certification have been lead during the past 15 years. For as much, since these two technologies receive a lot of attention in multiple industrial sectors, they have also been getting more attention from researchers than SpaceFibre. There are now tools available to simulate, configure and validate TTEthernet networks and Time Sensitive Networking tools are getting more mature every day. SpaceFibre has been modelled in OMNET++ simulator [69] but further research will be required in order to validate the real-time behaviour proposed by its medium access strategy.

Conclusion

To conclude on this first contribution, the qualitative comparison, based on the criteria that we identified in the previous chapter, lead us to select three suitable candidates for a next generation satellite network: TTEthernet, SpaceFibre and Time Sensitive Networking. This comparative study has been published in [19] where only Ethernet technologies are considered and in [20] where all the above technologies are considered. Further studies are necessary to decide which of these technologies (if any) would be the next-generation satellite network technology. As the reader may have glimpsed in Section 7.2.2, each technology has its advantages and drawbacks and the road is still long before the decision is made for future satellites. The most plausible situations that we envision are that either one technology is selected for all satellite missions or rather several technologies are selected, each being adapted to as specific type of mission. For instance, Time Sensitive Networking could be used for satellite constellations in a new space context where the number of components needed for the entire constellation will be high and benefit for the TSN mass market whereas TTEthernet or Spacefibre could be used for more specific mono-satellite missions where space-mature (e.g. rad-hard) components are really needed.

In the following part, since Time Sensitive Networking was totally new in the spacecraft industry, we further analyse its suitability by generating network configurations that satisfy the quality of service requirements of next-generation satellites.

Part III

Contribution: Computation of TSN Networks Configurations for Next-Generation Satellite Networks

Chapter 8

Problem Statement 2

Among the three candidate technologies identified by our first contribution, this PhD focuses on Time Sensitive Networking for the following reason: while the two other candidates have already been studied and even started to be included in satellite network designs at the time of writing this manuscript, Time Sensitive Networking was completely unknown to the spacecraft industry. It had never been addressed before in the scope of a next-generation unified *Platform & Payload* network [18]. Thus, our industrial partner was really willing to get to know the technology and unleash its full potential.

Time Sensitive Networking (cf. Chapter 4) is roughly composed of 20 standards. Gaining expertise on this novel technology and mastering all its mechanisms so as to exploit them correctly takes a lot of time. In addition, all standards (i.e. all mechanisms) might not be necessary to satisfy the satellite requirements. Therefore, the goal of this second problem is to reduce the effort on the network system designer side by identifying a subset of standards able to satisfy the system needs and by automatically computing network configurations based on these standards and requirements.

To do so, we refine the modelling of the system started in the previous chapter by detailing the model of flows travelling across the network. A formal definition of all the quality of service requirements (i.e. performance and fault tolerance requirements) that could be encountered in the use cases to come is proposed. Some definitions and models of this chapter have been retrieved from [17]. Finally, once the model and the requirements have been introduced, the last section of this chapter formulates the problem and showcases our contribution.

8.1 Flow Modelling

8.1.1 Definitions

The applications running on the computing devices communicate with the sensing and actuating devices through the on-board network with messages. These messages are gathered under the concept of flows.

Definition 38 (Flow, f , \mathbb{F}) *A flow is a unidirectional sequence of messages from one sender to one or several receivers. Let us denote f a flow, and \mathbb{F} the set of flows of the system. A flow is characterized by the following tuple:*

$$\forall f \in \mathbb{F} < Src_f, LDests_f, Size_f, r_f > \quad (8.1)$$

Where :

- $Src_f \in \mathcal{D}$ is the device from which the messages are generated and emitted,
- $LDests_f \subseteq \mathcal{D} \wedge Src_f \notin LDests_f$ is the set of receiver end-stations,
- $Size_f$ is the constant size in bytes of one message,
- $f.r_f$ is our way to express the periodicity of the flow (see Eqn. 8.3), $r_f \in \mathbb{Z}$.

Hypothesis 2 In this model, messages are embedded in Ethernet frames and we consider that :

$$\forall f \in \mathbb{F}, Size_f \leq MTU_{Ethernet} \quad (8.2)$$

This means that one applicative message of a flow will lead to one frame in the network. The definitions, properties and constraints in the rest of the document apply this hypothesis. If this hypothesis was to be relaxed, the definitions, properties and constraints of the document would have to be slightly redefined (see Appendix C.1).

Hypothesis 3 In this study, we agreed with the industrial to reduce the problem by only considering unicast flows. Therefore there is only one receiver per flow. Let $Dest_f$ denote this single receiver.

The period P_f of flow f is linked with its ratio r_f with the following equation:

$$\forall f \in \mathbb{F}, \begin{cases} r_f \leq -1 & \implies P_f = |r_f| * P_{MIF} \\ r_f > 1 & \implies r_f \text{ messages per } P_{MIF} \iff P_f \text{ is a period during which } r_f \text{ messages are sent} \\ r_f = 0 & \implies P_f = NA \text{ (} f \text{ is non-periodic)} \end{cases} \quad (8.3)$$

We remind that $k = \frac{P_{MAF}}{P_{MIF}}$ represents the number of MIF cycles during one MAF cycle (cf. Def. 35). Then, figuratively, if $r_f > 1$, flow f sends r_f messages during one P_{MIF} , hence $k * r_f$ messages during P_{MAF} . If $r_f \leq -1$, flow f sends one message during $|r_f| * P_{MIF}$, hence $\frac{k}{|r_f|}$ messages during P_{MAF} . The concept of ratio is illustrated in Fig. 8.1 with, in blue a flow with $r_f = -4$ and in green a flow with $r_f = 2$.

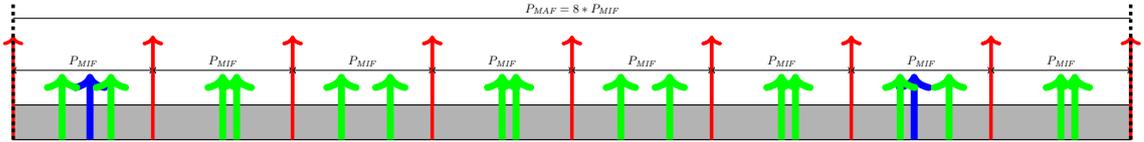


Figure 8.1: Ratio concept with two flows ($r_{f1} = 2$, $r_{f2} = -4$) and $k = 8$

Property 3 (Link between r_f and k)

$$r_f < -1 \implies r_f \mid k \quad (\iff lcm(|r_f|, k) = k) \quad (8.4)$$

Hypothesis 4 In this document, we only consider periodic flows that send at least one message every P_{MAF} and at most one message per P_{MIF} i.e. :

$$r_f \in [-k, -1] \quad (8.5)$$

Definition 39 (Flow - restriction) Therefore in this document a flow $f \in \mathbb{F}$ is characterized by the tuple $\langle Src_f, Dest_f, Size_f, P_f \rangle$ where P_f is the period of the flow.

Definition 40 (f_l, τ_{f_l}) A flow is a sequence of messages (or frames). Let $f_l, l \in \mathbb{N}$, denote the l -th message of f and τ_{f_l} the transmission duration of f_l .

Remark 14 Since the message size per flow is constant, the transmission duration per flow is constant too i.e. $\forall i, j \in \mathbb{N}, \tau_{f_i} = \tau_{f_j}$

8.1.2 Motivating Example: Adding Flows

In our motivating example (cf. 5.2.2), the applications running on the devices communicate with three flows. Starting now, the convention used for naming flows is the following:

Rule 2 (Naming Flows) A flow originating from Src_f and going to $LDests_f$ will be named:

$$f_Src_f_Dest_f_ID \quad (8.6)$$

Where ID is a user defined identifier used to distinguish several flows with the same senders and receivers.

Thus the flows, represented in Fig. 8.2, are the following:

- $f_OBC_HPActuator_1 : \langle OBC, HPActuator, 1500, 1 \rangle$,
- $f_OBC_HPSensor_1 : \langle OBC, HPSensor, 1500, 1 \rangle$,
- $f_HPSensor_OBC_1 : \langle HPSensor, OBC, 1500, 1 \rangle$,

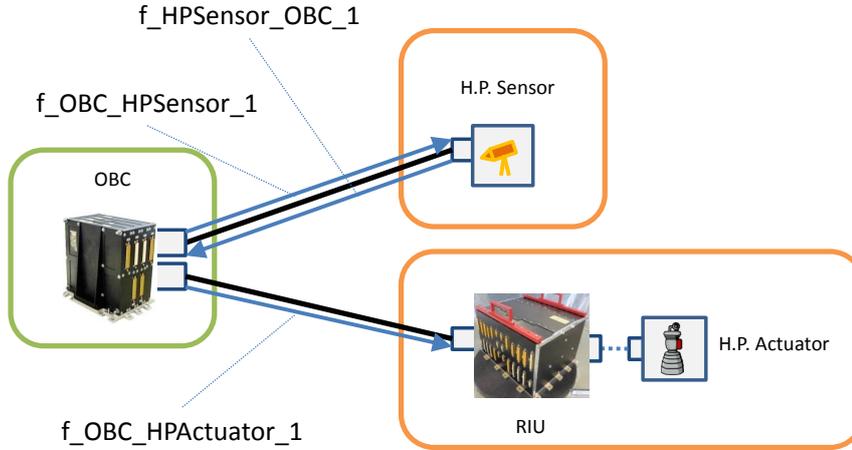


Figure 8.2: Adding three flows to the motivating example

8.1.3 Restriction to Flow Level Requirements

In the original model [17], the specification and the constraints of the system were expressed at application level. Indeed, the system is composed of several applications running on the end-stations and communicating through flows (or *streams* in TSN vocabulary). After detailing how applications

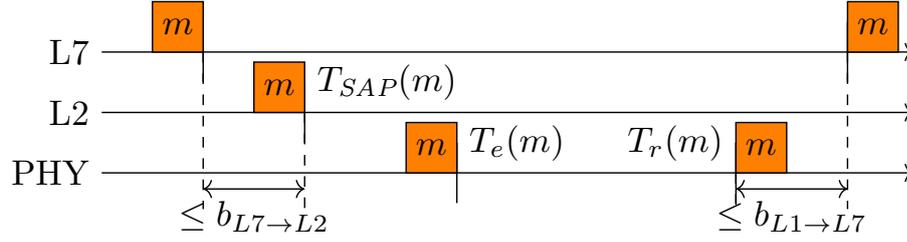


Figure 8.3: Frame behaviour on the network

send messages into the network, we will propose to go from application level requirement to flow level requirements in order to simplify the model.

At emission, the implementation is done as follows: when an application produces a message, it is put into a mailbox (ISO L7) and then placed in the appropriate queue at MAC level (ISO L2).

Definition 41 (Deposit / Emission instants) Let m be a message. We define the deposit instant $T_{SAP}(m)$ as the instant at which m is deposited in the L2 service access point. We define the emission instant $T_e(m)$ as the instant at which the first bit of m is emitted on the medium.

Definition 42 (Reception / Delivery instants) Let m be a message. We define the reception instant $T_r(m)$ as the instant at which the last bit of m is received at receiver end-station physical level. We define the delivery instant $T_d(m)$ as the instant at which m is provided to the receiver application.

Hypothesis 5 (Restricting the problem at flow level) Since we consider real-time equipments, we assume that the time between the production at applicative level and the placement into a queue at MAC level happens in a bounded known time ($b_{L7→L2}$). In the same way, we consider the delay between the reception of a message and its delivery (i.e. availability) at applicative level to be bounded ($b_{L1→L7}$). This is illustrated in Fig. 8.3. As a consequence, the configuration problem can be defined and solved solely at the network level.

8.2 Quality of Service Requirements

In the following sections, we propose to further detail the requirements of the system by deriving them from Application-Level Properties 1, 2 and 3 introduced in Chapter 5.

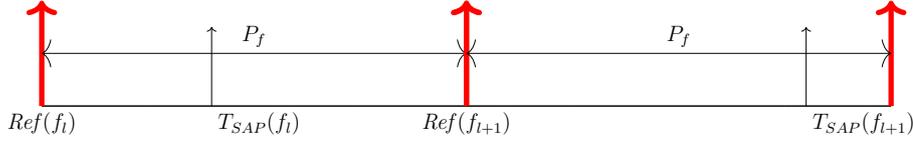
Choice 1 In this document, we focused on the requirements of Application-Level Property 1 and a subset of Application-Level Property 3.

The priority was to assess, via configuration, the network performance quality of service of TSN, before addressing its fault tolerance quality of service requirements.

8.2.1 Reference Instants

Before presenting the requirements, let us define the concept of reference instants.

Definition 43 (Ref(f_i)) We define the reference instant of f_i as $Ref(f_i) = l \times P_f$. For any message f_i of any periodic flow, f_i will be enqueued during the interval $T_{SAP}(f_i) \in [Ref(f_i), Ref(f_{i+1})[$.

Figure 8.4: Reference date of message f_i and f_{i+1} with $r_- = -2$

We illustrate this concept in Fig. 8.4.

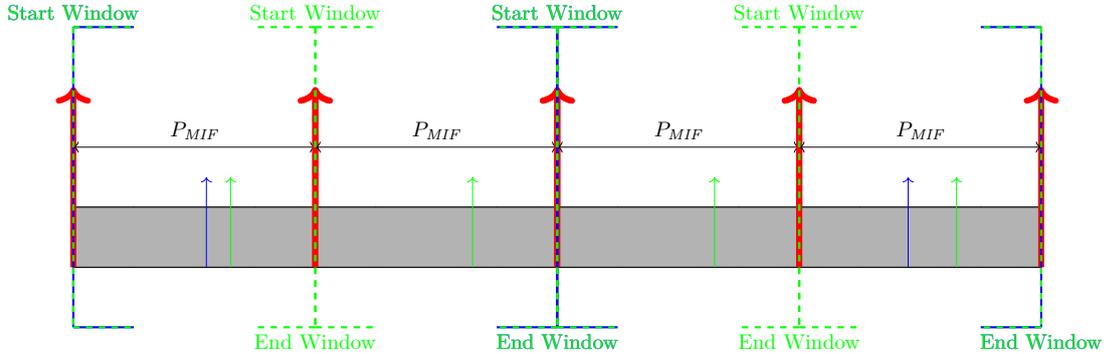
Remark 15 *This definition means that, contrary to a majority of works in the state of the art, flows in this model are not strictly periodic. In fact, the deposit and emission instants of a message f_i can happen at any time in the window $[Ref(f_i), Ref(f_{i+1})]$ instead of happening exactly one period apart from f_{i-1} .*

8.2.2 Mixed Traffic Types Requirements

The following paragraphs will introduce the different performance constraints identified for the flows. For most constraints, an example will illustrate the constraint being satisfied. In these figures, the colors representing the Input/Processing/Output parts have been removed for the sake of readability of the flow constraints.

Performance Requirement 1 (Deadline) *Let a flow $f \in \mathbb{F}$, it comes with a deadline constraint so that $T_r(f_i) \leq f_i.\text{deadline}$ and $f_i.\text{deadline} \leq Ref(f_i) + P_f$.*

This entails that the latest reception of f_i of flow f must be terminated before the beginning of the emission window of f_{i+1} . In the case in which $\forall l \in \mathbb{N}, f_l.\text{deadline} = Ref(f_l) + P_f$, the flow is said to have *implicit deadlines* [47]. Deadlines for several messages of two flows are illustrated in Fig. 8.5.

Figure 8.5: Deadlines for 2 flows ($r_{f1} = 2 * P_{MIF}$, $r_{f2} = 4 * P_{MIF}$) and $k = 8$

Definition 44 (Reception Jitter) *The reception jitter [105] or jitter between two frames f_l and f_m is defined as the variability of their reception dates. It is denoted $Jit_{f_l, m}$ such that $\forall f \in \mathbb{F}, \forall l, m \in \mathbb{N}, Jit_{f_l, m} = |(T_r(f_l) - Ref(f_l)) - (T_r(f_m) - Ref(f_m))|$. The overall jitter of a flow is denoted Jit_f such that $\forall f \in \mathbb{F}, Jit_f = \max_{l, m} Jit_{f_l, m}$.*

Remark 16 The latency of a message f_l can be defined in this model as $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, Lat_{f_l} = T_r(f_l) - Ref(f_l)$.

Performance Requirement 2 (Jitter) A flow f also has a jitter constraint defined as $f.jitter \in \mathbb{N} \cup \{NA\}$ where NA stands for not applicable (thus no jitter constraint) and otherwise $f.jitter$ is the maximum accepted jitter.

Jitter of f_m with respect to f_l is illustrated in Fig. 8.6

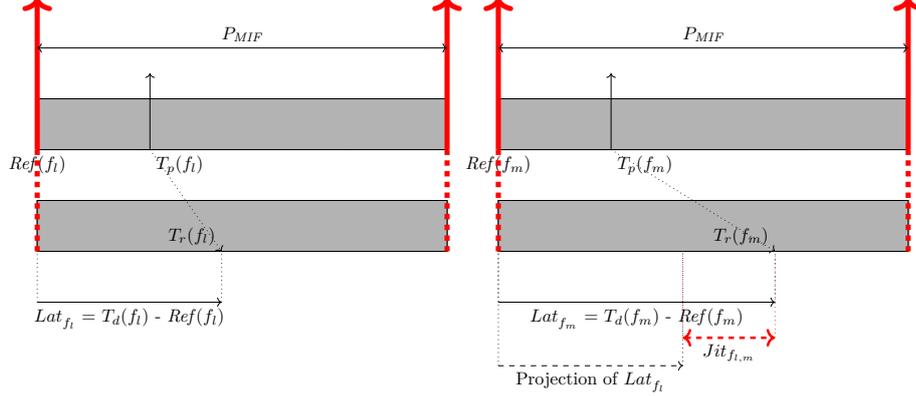


Figure 8.6: Jitter visualisation with two messages f_l and f_m

Definition 45 (Jitter / No Jitter Flows, \mathbb{F}_j) We refer to the flows with jitter constraints as jitter flows and to the others as no jitter flows, and we denote $\mathbb{F}_j = \{f \in \mathbb{F} | f.jitter \neq NA\}$ the set of jitter flows.

8.2.3 Fault Tolerance Requirement

In addition to performance, safety requirements are often required. In particular ARINC 664 [3] or TTEthernet [102] networks offer Fault Isolation mechanism. Among the faults supported by those networks, we restrict ourself to message loss.

Definition 46 (Message loss independence) A system is considered as message loss independent if for all flow f , the loss of messages of f has no negative impact on the performance (deadline/jitter) of the other flows.

Safety Requirement 1 Any configuration of the network should fulfil the message loss independence requirement.

8.2.4 Motivating Example: Adding Flow Constraints

Three flows have been identified in the motivating example (cf. 8.1.2). Let us now assign them flow constraints according to the previously defined constraints. We denote $\Upsilon(f)$ the function that, given a flow, returns its flow constraints. In the motivating example:

$$\Upsilon(f_{OBC_HPActuator_1}) = \begin{cases} Deadlines("implicit") \\ SafetyReq1() \end{cases} \quad (8.7)$$

$$\Upsilon(f_OBC_HPSensor_1) = \begin{cases} \text{Deadlines("implicit")} \\ \text{SafetyReq1()} \\ \text{Jitter}(500\mu\text{s}) \end{cases} \quad (8.8)$$

$$\Upsilon(f_HPSensor_OBC_1) = \begin{cases} \text{Deadlines("implicit")} \\ \text{SafetyReq1()} \end{cases} \quad (8.9)$$

8.3 Industrial Considerations

8.3.1 Production Contract and Release Instant

Applications come with a set of flow contracts, where each flow contract consists of a temporal window for messages production (see Fig. 8.7) so that applications meet their performance, safety and development requirements (see Section 8.2). Such a contract is bargained off-line between applications and platform providers. It is expected that applications always respect their contracts and that the on-board network ensures the quality of service of each application as long as the applications respect their contracts.

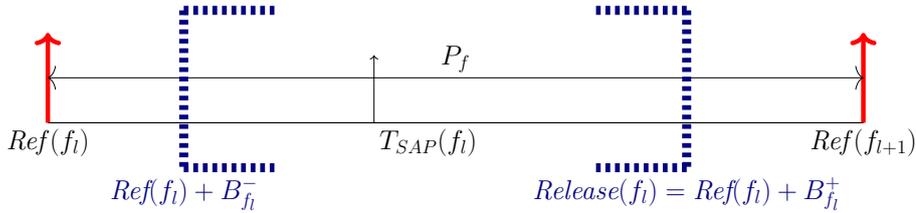


Figure 8.7: $Ref(f_l)$, $T_{SAP}(f_l)$ and $Prod(f_l)$

Definition 47 (Application Flow Contract) Let f_l be the l -th message of f . The production contract associated to f_l is the interval $Prod(f_l) = [Ref(f_l) + B_{f_l}^-, Ref(f_l) + B_{f_l}^+] \subseteq [Ref(f_l), Ref(f_{l+1})]$, where $B_{f_l}^-$ (resp. $B_{f_l}^+$) is the earliest (resp. latest) production offset. The upper bound of $Prod(f_l)$ is called Release instant and denoted $Release(f_l) = Ref(f_l) + B_{f_l}^+$. The production traffic contract for a flow f , denoted $Prod(f)$, is defined by $Prod(f) = \cup_{l \in \mathbb{N}} Prod(f_l)$.

This definition entails that $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, T_{SAP}(f_l) \in Prod(f_l)$ i.e.:

$$Ref(f_l) + B_{f_l}^- \leq T_{SAP}(f_l) \leq Release(f_l). \quad (8.10)$$

8.3.2 Application Emission Scheme

In the current version of the satellite system, the communication on the platform side relies most of the time on 1553 bus. Applications emission is based on a precomputed scheduled accesses to a list of descriptors that compose a frame ahead of time; and not based on a queue like it would be on Ethernet for instance. Therefore, the application can deposit its messages whenever the message is ready. There is no constraint on data production as long as it is deposited before the scheduled register read.

The software running on the satellite relies on this application emission scheme. Changing the emission scheme would therefore have a significant impact on the amount of code that might have

to be redeveloped. Hence, the configured TSN network shall aim at minimizing the amount of redevelopment and more generally the impact on on-board software.

This is translated in the following optimization objective:

Development Effort Requirement 1 *Any configuration should maximize the length of $Prod(f)$ (see Def. 47). In particular, $B_{f_i}^- = 0$ allows the scheduler to start executing any ready task at the beginning of a MIF Period and maximizing $B_{f_i}^+$ (ideally $B_{f_i}^+ = P_f$) gives more time to execute tasks during a MIF Period.*

The idea is to maximize the duration in which the application can emit its messages and mimic the 1553 behaviour. The network will be in charge of correctly delivering the message while respecting the performance and quality of service requirements.

8.3.3 Cost of a Network Upgrade

In addition to software, the cost on the hardware of the next-generation network is also important. In fact, switching from a legacy 1553 + SpaceWire network towards a Ethernet/TSN network might have two impacts. First, changing all the devices to TSN devices might have a considerable cost that the industrial is not willing to endorse. In addition, TSN is composed of more than 20 standards. The more standard implemented, the more complex the fault tolerance analysis will be. Second, in the current architecture, the receiving devices are extremely simple and the On-Board Computer (computing device) is hardly the only complex device in the architecture. Again, increasing the complexity in additional devices might require further analysis on the performance, the fault tolerance, etc. of the system. As a consequence, the configured network shall try to minimize the cost of the hardware by controlling the use of TSN, in number of device or number of implemented standards, but also by keeping the actuating and sensing devices as simple as possible. One other solution would have been to relax this "simplicity of the receivers" paradigm and distribute intelligence across the network but it was out of scope of the study w.r.t. the industrial partner requirements.

8.4 Contribution Overview

In order to demonstrate the suitability of Time Sensitive Networking w.r.t. the requirements of the spacecraft industry, we propose to find network configurations that would satisfy these requirements. Therefore, we wondered :

Definition 48 (Problem 2 - Step 1) *Given the performance and safety quality of service requirements of the spacecraft industry presented in Section 8.2, what is the smallest subset of TSN standards required for a next-generation satellite network ?*

In fact, Time Sensitive Networking (cf. Chapter 4), is not one but a set of roughly 20 standards. Therefore, the design and configuration of such networks is a difficult task. In order to help the network architect design a satellite unified network based on TSN, it would be nice to define a small subset of standards that would be sufficient to fulfil the coming network performance and fault tolerance requirements.

Choice 2 (Choice of TSN standards) *We first presented a glimpse of subset, or profile in the IEEE vocabulary, in [15] and then in [18] and it is still under consolidation in a joint effort between SAE and IEEE (see. IEEE/SAE P802.1DP TSN Profile for Aerospace [65]). This profile contains*

a certain number of standards. In this study, we reduce it to TSN standard 802.1Qbv only so as to cope with the requirements presented in Section 8.2.¹

This choice is motivated by two reasons. First, at the time of writing this document, there were two standards available to handle low jitter traffic among TSN standards: 802.1Qbv and 802.1Qcr - Asynchronous Traffic Shaper (ATS [63]). Most of the papers in the state of the art relied on 802.1Qbv for low jitter traffic. In addition, there was no existing TSN hardware embedding ATS at that time while 802.1Qbv was largely represented.

Remark 17 *In this manuscript, we consider that 802.1Qbv standard includes all transmission selections algorithms, meaning that, for instance, Time Aware Shaper and Credit Based Shaper are included in 802.1bv standard and may be configured. This is not actually really the case: 802.1Qbv and 802.Qav (CBS) are two different standards included in the same 802.1Q standard. However, 802.1Q contains more mechanisms than the ones we have selected for this study. Therefore, it was easier to describe these mechanisms as part of 802.1Qbv.*

Definition 49 (Problem 2 - Step 2) *Given the quality of service requirements, is it possible to compute valid configurations of TSN 802.1Qbv standard?*

Definition 50 (Valid configuration) *A valid network configuration should respect the all the quality of service requirements of all the flows.*

There is certainly a lot a valid network configurations. However, we want to find a valid one that is acceptable in terms of industrial applicability i.e. a reduced cost/impact not only on hardware implementation but also on the software and the integration process.

To confront this second problem, we propose to find a way to automatically generate valid TSN configurations, doing it *by-hand* not being a very scalable and industrial method. Based on existing methods from the state of the art, we generate TSN configurations thanks to constraint programming. This second contribution is organized in three chapters. In the first chapter, we detail a configuration model for 802.1Qbv standard. Then, we introduce in the related work section the state of the art methodology for the computation of TSN network configurations. Finally, we introduce the industrial use cases that we use in Chapter 11. In the second chapter, the novel concept of Egress TT configurations is introduced. Its advantages and drawbacks are discussed, and two strategies (or *implementations* of Egress TT) for the generation of TSN network configurations are proposed, namely Exclusive Queue Allocation and Size Based Isolation. In particular, we detail how these configurations are computed with the previously introduced configuration model and methodology. In the last chapter, the performance of these two implementations of Egress TT for TSN networks is compared with state of the art configuration approach (entitled End-to-End TT) on several use cases (including real-life systems) so as to underline potential improvements on scalability and computation effort.

¹Additional requirements, in particular fault tolerance quality of service requirements derived from Application-Level Properties 2 and 3 (cf. Section 5.3.3 & 5.3.2), were taken into account in the definition of the profile hence leading to the selection of additional TSN standards

Chapter 9

Insights on the Configuration of IEEE 802.1Qbv

Time Sensitive Networking has been selected, in the qualitative study, as one suitable candidate for supporting the on-board network of next-generation satellites along with TTEthernet and Spacefibre. In order to further validate the suitability of Time Sensitive Networking, we wish to consolidate network configurations that fulfil the quality of service requirements (performance and fault tolerance). This configuration challenge is not easy: TSN is composed of several standards, each composed of several mechanisms with several parameters that need to be instantiated for any configuration. In Chapter 8, we have reduced TSN to one standard - IEEE 802.1Qbv - identified in Choice 2 as the necessary standard to support the quality of service requirements of next-generation satellites (cf. Section 8.2). Therefore, in this chapter, we first propose an insight on the configuration of a network relying on IEEE 802.1Qbv by introducing a formal configuration model. Then, we introduce End-to-End TT and its derivatives, a family of configuration designed to support flows with very low jitter requirements. Finally, we discuss the major advantages and drawbacks of these configurations with respect to Problem 2 Step 2 (cf. Def. 49).

9.1 Configuration Model

We detail hereafter the configuration model for a TSN network relying on IEEE 802.1Qbv. This network is composed of several TSN devices and multiple links. These devices (end-station or switch) are composed of a certain number of output ports. A system configuration entails a network configuration i.e. the configuration of every port in every device as well as a configuration of the flows travelling through the network. We now formalize the port configuration.

9.1.1 802.1Qbv Port Configuration

An output port is composed of up to eight internal queues, also known as *traffic classes*. These queues have priorities, and come with several mechanisms to do traffic shaping, bandwidth sharing, etc. These concepts have been extensively described in Section 4.2. We now propose to formalize these concepts so that they can be translated into a model which will be used later for constraint programming.

Definition 51 (Output port) Let \mathbb{P} denote the set of output ports in the network. An output port $p = (q_0, \dots, q_7, TS)$ is composed of eight¹ internal queues q_j and a Transmission Selection (TS). Each queue $q = (TSA_q, TG_q)$ is associated with a Transmission Selection Algorithm (TSA) as well as a Transmission Gate (TG). (cf. Chapter 4).

We summarize the output port model in Fig. 9.1. Both internal queues and TS will rule when frames access the medium.

Transmission Selection Algorithm (TSA). TSA belongs to a list of available algorithms (cf. Section 4.2.2) implemented by the hardware device. Examples of such algorithms are CBS (for *Credit Based Shaper*) and *none* when no restriction on head of queue is added².

Definition 52 (Ready(m, TSA_q)) Let $p = (q_0, \dots, q_7, TS) \in \mathbb{P}$ a port, let m a message stored queue $q_j, j \in [1, 8]$. To allow the transmission of message m from queue q , the transmission selection mechanism of that queue $q = (TSA_q, TG_q)$, if any, shall mark message m as ready. Let $Ready(m, TSA_q)$ denote that message m is ready.

Remark 18 In the case a queue has no configured TSA, a message m will be ready as soon as it becomes head of the queue.

Transmission Gates (TG). This mechanism, also referred to as *Time Aware Shaper* (cf. Section 4.2.3), adds the possibility for internal queues, in both switches and end-stations, to be regulated according to time-driven rules. In effect, there is a gate TG_q associated to any internal queue q which can be opened or closed. The schedule switching from open to closed and back is pre-computed off-line per port, is periodic and is called a *Gate Control List* (GCL).

Definition 53 (Gate Control List) Let p a port, its associated gate control list, denoted $GCL(p)$ ³, is defined by the list $[e_0, \dots, e_u - 1]$ of u events $e_i = \langle s_i, t_i, d_i \rangle$ where

- $s_i = \langle s_{i,0}, \dots, s_{i,7} \rangle$ is the status of the gates $s_{i,j} \in \{o, C\}$ where o stands for open and C stands for closed,
- $t_i \in \mathbb{N}$ is the time offset from the start of e_0 at which event e_i starts,
- $d_i = t_{i+1} - t_i$ is the duration during which the schedule s_i will hold.

In particular, the period of repetition of the pattern is $\sum_{i \in [0, u-1]} d_i$ and $\gcd_{i \in [0, u-1]}(d_i)$ is called gate granularity. These parameters are illustrated in Fig. 9.1.

Remark 19 (Link $TG_q, GCL(p)$) The gate control list describe per port the status over time of all the gates for all the queues in that port. The variable TG_q is denotes the gate of queue q (in port p).

Hypothesis 6 (Gate Control List period) Since the system we consider is periodic, it is sufficient to compute the gate schedules on the hyper-period of all its flows. Therefore, $\sum_{i \in [0, u-1]} d_i = P_{MAF}$.

Remark 20 Ethernet-capable devices do not have transmission gates and this is equivalent to the gates being open all the time, i.e. $\forall p, GCL(p) = [e_0] = [\langle \langle o, \dots, o \rangle, 0, P_{MAF} \rangle]$.

Transmission Selection (TS) A frame is emitted when it is available for transmission (cf. Def. 54) and has the highest priority among frames available for transmission (cf. Def. 55).

¹Without loss of generality, we assumed a fixed number of queues.

²Reminder: Asynchronous Traffic Shaper is considered out of scope of this study

³This is a simplification of IEEE 802.1Q standard where *OperControlList* is the only considered parameter.

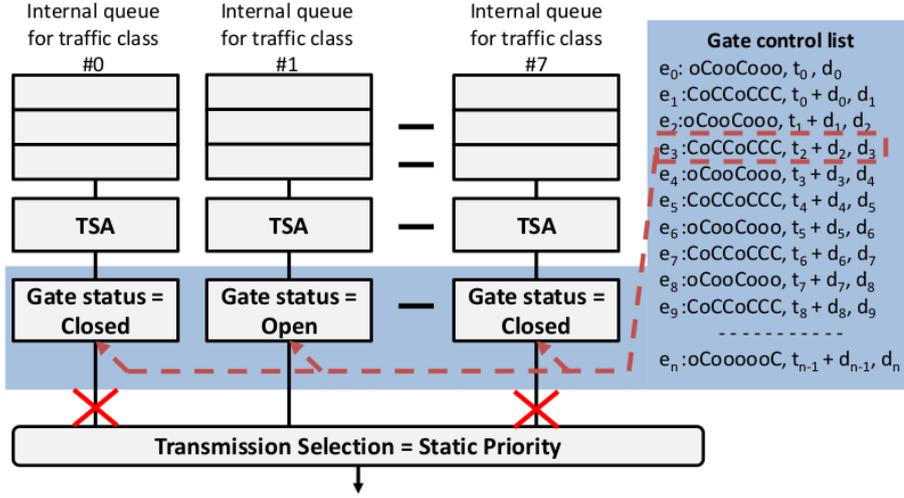


Figure 9.1: Scheduled Traffic Parameters

Definition 54 (Frame available for transmission) A frame (or message) m in queue q of output port p is "Available for transmission" at instant t when:

1. The frame is the head of q ,
2. The transmission selection algorithm of q has marked m as ready
3. TG_q is open at instant t ,
4. TG_q remains open long enough to transmit the frame.

Formally,

$$\begin{cases} \text{head}(q) = m, \text{Ready}(m, TSA_q) \\ \exists m, k \in \mathbb{N} \text{ s.t. } \forall e_i \in GCL(p), i \in [m, m+k], s_{i \% n, q} = o, \\ \exists t \in t_m \leq [t, t_m + \text{sum}_{i=m}^{m+k} d_i] \text{ s.t. } ((t_m + \text{sum}_{i=m}^{m+k} d_i) - t) < \frac{\text{Size}_f}{r} \end{cases}$$

where r denotes link speed.

Remark 21 The sum element in the above equation accounts for potential consecutive gate open events.

Remark 22 (Frame Preemption) We do not consider TSN standard for frame preemption [116] in this configuration model. Such evolution could be done with inspiration from [117] by slightly redefining the equation of Def 54. The standard is presented in Appendix A.1.

Definition 55 (Transmission Selection (TS)) The priority attribution is the following: the higher the traffic class number, the higher the priority. For instance, traffic class #7 has a higher priority than #0.

Definition 56 (Config(p)) The configuration $Config(p)$ of a port $p \in \mathbb{P}$ consists in configuring the Gate Control List of the port, if any, as well as selecting and configuring the Transmission Selection Algorithms for each queue, if any. In summary

$$Config(p) = \begin{cases} GCL(p) \\ \forall q \in \{q_1, \dots, q_8\}, TSA_q \end{cases}$$

Several classic configurations, also known as architectures of interest, exist for a port, depending on the choice of TSA and the gate configuration. Some architectures of interest are described in Section 4.2.5.

9.1.2 System Configuration

Now that the notion of port configuration has been formalized, we formulate what the notion of configuration means at system level. In fact, a system configuration is composed of a configuration of all flows and a network-level configuration.

Definition 57 (Flow configuration) The configuration of a flow f is $Config(f) = [(p_1, FtQM_{p_1}), \dots, (p_l, FtQM_{p_l})]$ where

- $Path_f = p_1, \dots, p_l$ is the path followed by f , that is the sequence of output ports that are crossed;
- $FtQM_{p_j}$ is the associated Flow to Queue Mapping on each port p_j . In particular, since a port is defined by $p = (q_0, \dots, q_7, TS)$, $FtQM_p(f) \in \{q_0, \dots, q_7\}$;

Definition 58 (Config(Net)) A network-level configuration $Config(Net)$ consists in finding a configuration for all the output ports i.e.

$$Config(Net) = \{Config(p), \forall p \in \mathbb{P}\}$$

In summary, computing a system configuration consists in determining:

$$\begin{cases} \forall f \in \mathbb{F}, Path_f \\ \forall f \in \mathbb{F}, \forall p \in Path_f, FtQM_p(f) \\ \forall p \in \mathbb{P}, GCL(p) \\ \forall p = (q_0, \dots, q_7) \in \mathbb{P}, \forall q \in \{q_0, \dots, q_7\}, TSA_q \end{cases} \quad (9.1)$$

9.2 Related Works: Existing System Configurations for Low Jitter Requirements Support

Let us discuss existing system configurations in the state of the art that could be suitable for the constraints identified in Section 8.2 and 8.3. One could distinguish two subsets of configurations, aiming at satisfying the two performance requirements of the on-board network: the first family of configuration is designed for flows with deadline/latency requirements whereas the second family is designed for supporting traffic with low jitter requirements. While the first family has been extensively discussed in the state of the art (e.g. priority assignment [50], AVB/CBS [34, 35, 87], pre-shaping [78, 89], cyclic queueing and forwarding [88, 123], TT-CBS-BE [77, 31], ETS [107, 106]), the computation of configurations from the second family is still a hot topic in the networking community

(considered as a NP problem [23]). That is why this state of the art focuses on configurations designed for low jitter requirements.

For low jitter requirements support, hardly all the papers in the state of the art rely on transmission gates, and transmission gates only (i.e. TSA is set to none for all queues). In addition, they consider that the routing of flows is known and fixed a priori. This means, unless explicitly stated, the system configuration (cf. Equation 9.1) is simplified to:

$$\begin{cases} \forall f \in \mathbb{F}, \forall p \in Path_f, FtQM_p(f) \\ \forall p \in \mathbb{P}, GCL(p) \end{cases} \quad (9.2)$$

After a short introduction on how constraint programming is used to compute these configurations, we detail these GCL-based configurations hereafter.

9.2.1 Methodology: Configuration Generation with Constraint Programming

In our coming contribution, we will adopt the most common methodology used to generate valid configuration for the Time Aware Shaper. This methodology is based on constraint programming ([6, 101]). In fact, the configuration of the TSN Time Aware Shaper mechanism can be translated into a constraint programming problem. It can then be tackled with either constraint programming solvers such as CPLEX [83] or Satisfiability Modulo Theory/Optimization Modulo Theory (SMT/OMT) solvers such as Z3 [32].

Three elements have to be defined for the constraint programming problem: the model, the decision variables and the constraints. The model describes the input on which the constraints will be evaluated (representing the system). The constraints form a system of equations: it is a mathematical formulation of the requirements of the system. The decisions variables are variables in the mathematical formula to which a value must be decided.

In the case of network configuration, the model represents the network topology and its characteristics as well as the flows' definition (cf. Section 8.1). The constraints represent both the quality of service requirements (e.g. Section 8.2) as well as technology/system related constraints (e.g. maximum size of frames, maximum link capability, maximum number of frame on a link at any time, etc.). The decision variables are the parameters of the mechanism that the network designer is aiming at configuring (cf. Section 9.2). In that sense, in Chapter 10, we will reuse part of the model and constraints from the state of the art. In addition, we will define new constraints and decision variables of our novel configuration family.

9.2.2 End-to-End TT configuration and its derivatives

End-to-End TT configurations concept

In the state of the art, the most common configuration supporting low jitter traffic requirements is the so called End-to-End TT configuration (cf. Section 4.2.5).

Definition 59 (End-to-End TT configurations) *End-to-End TT configuration consists of a schedule per frame (or frame offset) in a time triggered fashion for either jitter traffic or all traffic on all ports in their path. This schedule is computed a-priori offline based on the requirements of the system. In this configuration, the transmission instant of any frame at any node is known during the whole life cycle of the system or until a new configuration is computed.*

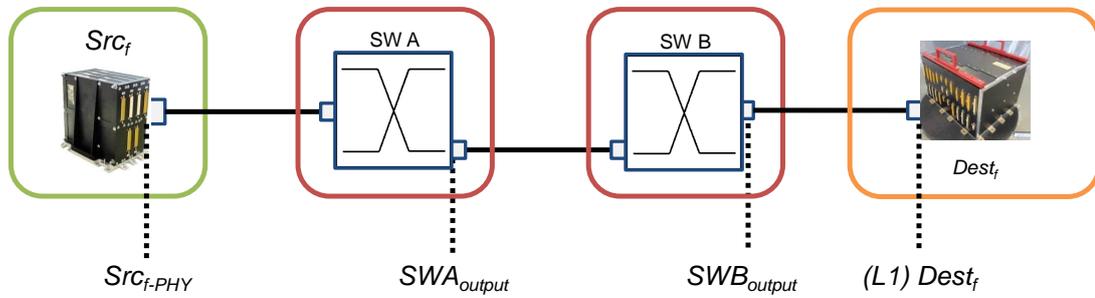


Figure 9.2: Topology illustration

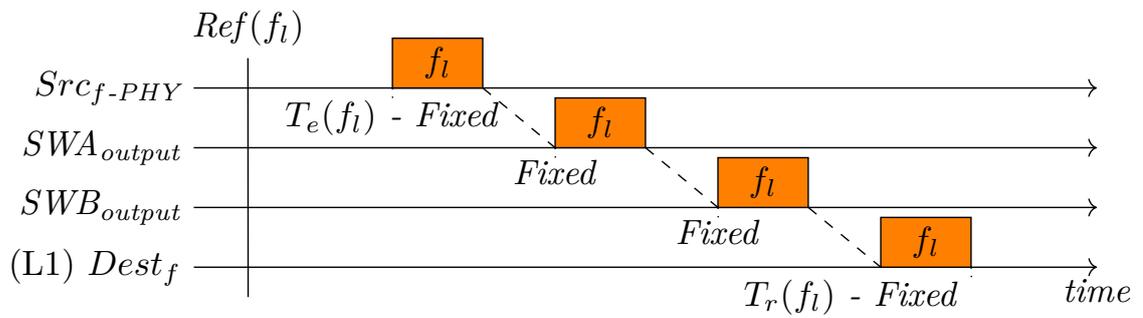


Figure 9.3: End-to-End TT Configuration

Fig. 10.1 illustrates an End-to-End TT configuration with one emitter, one receiver and two switches (SWA and SWB). The principle of these configurations is simple: by fixing the transmission instant of all frames in all hops, the latency and jitter of the flows are controlled and nothing unexpected can occur.

End-to-End TT roots in TTEthernet networks

In Ethernet networks, End-to-End TT configurations root back to TTEthernet networks. The pioneering paper is [110]. The authors introduce a formal TTEthernet network model and an associated set of constraints for SMT schedule generation, some of which we inspired from for the previously presented system model and the coming decision variables and constraints used to compute our novel configurations. Their methodology allowed to compute frame offsets, that were directly applicable into TTEthernet since TTEthernet provides a per frame scheduling capability. The configuration of the network was therefore immediate.

At that stage the authors had already made it clear that the computation of such schedules was expensive (cf. Section 9.3.2) and introduced an incremental strategy for configuration generation to reduce the computation effort.

Exploiting the constraints of the paper [110], [23] (short version) and [24] (long version) propose to create a schedule for both applications running on end-stations and the underlying TTEthernet network.

Link between Frame Schedules and GCL configuration

Then, authors have started to consider network based on Time Sensitive Networking instead of TTEthernet where the scheduling of frame is slightly different. In fact, in order to meet the very low jitter traffic requirements, TSN relies on schedules per queue instantiated through the time aware shaper mechanism. This is genuinely different from TTEthernet which proposes a per frame scheduling. Instead of consisting of an emission schedule of all frames on all hops of the network, TSN schedule consists in serving one or several queues during a configured duration in a scheduled fashion. These queues will have the possibility to emit frames according to the transmission selection rules defined in Section 4.2. Nevertheless, these authors tried to configure TSN in such a way that would be equivalent to a TTE per frame schedule.

There is a slight misalignment between the TSN configuration model and the most common configuration in the state of the art. In fact, on the one hand, as proposed in Equation 9.2, in order to configure the system, the parameters of the gate control list of any port (i.e. $GCL(p)$) have to be computed. On the other hand, the End-to-End TT approach defines frame schedules i.e. instants in which frames are transmitted on all hops. Naturally, one can wonder how to link these two concepts. In fact, in order to transform the per-queue scheduling capability of TSN into a frame scheduling capability, the authors in the state of the art take the problem the other way around. They compute frame offsets and from these offsets deduce gate opening and closing events. For instance, a frame f_l in a port p with a single queue q which offset has been determined to be at instant t will lead to a gate event $e = \langle \langle o \rangle, t, \tau_{f_l} \rangle$. If a frame follows f_l at instant $t + \tau_{f_l}$, then it will lead to a similar opening event, otherwise, a gate closed event will be created until the next frame transmission. In addition, in port with multiple queues, in order to avoid the medium being busy when the transmission of frame f_l must happen, an exclusive gating [28] pattern is proposed (cf. Section 31). Therefore, in addition to opening the gate of queue q at instant t , the gate event will also close the gates of all others queues before or at instant t . We illustrate this transformation in Fig. 9.4. In this figure, we have represented two frames f_l and g_m in port $p = (q_1, q_2)$, which have

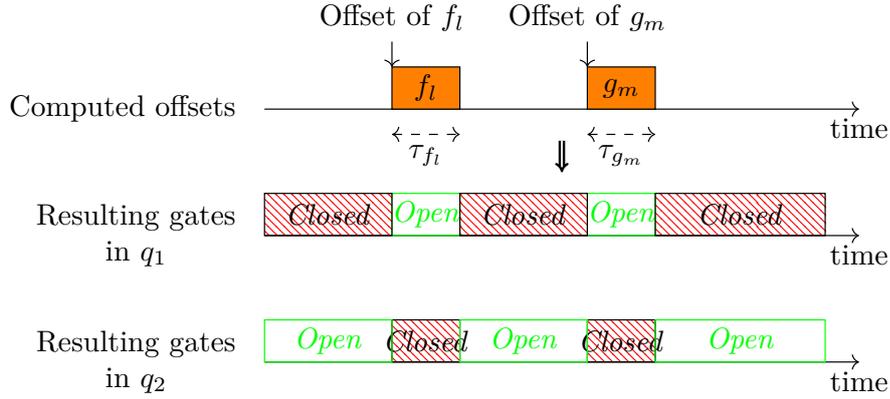


Figure 9.4: Offsets to GCL transformation

been configured to be placed into q_1 (i.e. $FtQM_p(f) = FtQM_p(g) = q_1$). Queue q_1 and q_2 have been configured in exclusive gating.

Need for Flow/Frame Isolation for End-to-End TT configurations of TSN networks

The per-queue schedule may generate non deterministic behaviour at message level in the presence of failure. This concept is illustrated with two figures: in Fig. 9.5a, the nominal expected behaviour (so as to cope with jitter requirements for g_m) is shown. Fig. 9.5b presents a scenario where message f_l is lost, leading g_m to be sent in place of f_l , creating an unwanted jitter.

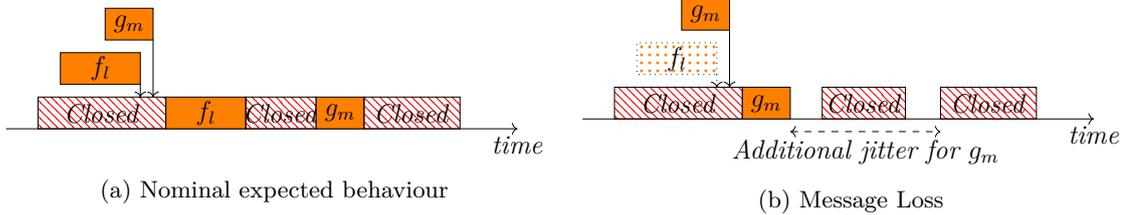


Figure 9.5: Need for Queue/Flow/Frame Isolation

In order to cope with the potential non-determinism induced by the loss of a frame, [25] adapts the constraints of [110] and introduces two new constraints namely *Flow Isolation* and *Frame Isolation*.

- *flow isolation*: a queue is dedicated to a flow from its first to its last message in an hyper-period. Therefore, at each instant, only messages from a single flow can be present in the queue and interleaving of frames from different flows is not allowed;
- *frame isolation*: a queue can be shared by several flows, but at each instant, only messages from a single flow can be present in the queue.

All cases prevent messages from different flows to be in the queue at the same time. Thus, a message loss cannot affect the behaviour of messages of other flows. From any of these two constraints, the flow to queue mapping ($FtQM_p(f)$, see Def. 57) i.e. the choice of a queue, per port, for a flow is immediately computed.

Remark 23 *The non-determinism induced by the loss of a frame is not compatible with safety requirement 1, therefore our configuration strategy will also have to include a constraint to eliminate this issue.*

(Re)ordering End-to-End TT schedules for no jitter traffic support

The previously quoted papers create schedules for jitter traffic (see Def. 45) without any consideration on the remaining traffic in the network. Therefore, in order to improve the performance of no jitter traffic (i.e. deadline requirements) [111], [36] and [53] introduce strategies, a priori or a posteriori, to modify the jitter frame schedule by either spacing the frame offsets or gathering them. [96] also proposes to add space between any two frame offsets, not in a no jitter performance consideration, but rather to leave time for potential retransmission of lost jitter frames.

Since our configuration only focuses on jitter traffic, we will not further detail these approaches. Nevertheless, the (re)ordering philosophy could be integrated into our configurations in a future work.

Group-of-frames scheduling

Most recently, a second family of configuration strategies has appeared. It computes configurations based on a schedule per group of frames instead of a schedule per frame, motivated by TSN Transmission Gates *per queue scheduling* capability. Therefore, instead of computing frame schedules and then converting them into gate control list configurations, the methodology now directly computes gate events.

[26] applies its TTEthernet schedule generation methodology [24] to TSN networks. The authors introduce new sets of constraints adapted for group of frames schedules as well as *Stream Isolation*, a fusion of *Frame isolation and Flow isolation* to again cover the loss of a frame. In [105], the same authors use their new constraints to implement a configuration generator and compare their two approaches (single frame offset v.s. group of frames offsets). It seems that the group of frames scheduling is a great improvement in terms of computation effort. Recently, in [100], the authors propose a group of frames configuration while relaxing the exclusive gating constraint. Based on previous constraints from [105] and new ones, the configurations they computed satisfy temporal constraints for jitter flows and no jitter flows using schedule porosity (i.e. (re)ordering) in an incremental approach based on the constraint programming methodology.

The performance of group of frame configurations, in terms of computation effort, has to be slightly detailed: in fact, when the jitter constraint is not very low (i.e. above $10\mu\text{s}$), the experiments in paper [105] show indeed that the computation time is smaller than End-to-End TT configurations. Nevertheless, once the jitter constraint gets smaller (i.e. around $1\mu\text{s}$ like it is the case for us), the experiments show that computation effort required for configuration generation is high and even greater than End-to-End TT configuration. That is why our coming configuration will not rely on group of frame scheduling.

Adding more variables to the configuration problem

Another group of papers have chosen to take more decision variables into account for configuring TSN networks. In particular, up to now, as all the previously cited paper in this section considered a fixed static routing for flows in their system, several papers (e.g. [46, 45, 77, 92, 93]) have proposed to relax this hypothesis with joint routing and scheduling End-to-End TT configuration generators. This increases the solution space of frame schedules by allowing the route of flows to be modified. To compute these configurations, the authors not only rely on constraint programming methodology

but also on heuristics. [120], one recent addition to the state of the art, claims that with their heuristic based approach, they have solved the scalability issue of very large network configuration generation. They can configure networks with 2000 nodes and 10000 flows. We do not detail further these papers since our contribution is based on constraint programming methodology (that works well on our problem) and fixed route for all flows.

9.3 Limitations of State of the Art Strategies w.r.t. Problem 2

While the existing End-to-End TT configurations and its extensions seem to be capable of handling the low jitter traffic requirements of next-generation satellite networks, the interest of the space industrials towards these configurations is limited for three reasons: their impact on development effort, their consequent computation cost and their upgrade costs.

9.3.1 Application Impact Issues

End-to-End TT configurations can provide guarantees for flows with very low jitter requirements. In order to be compliant with safety objective 1, i.e. be message loss independent, End-to-End TT configurations in the state of the art rely on Frame Isolation or on Flow Isolation. This comes at a cost: the impact on application of such configurations is substantial. Nevertheless, it has never been considered in the state of the art. Let us explain this application impact with the help of the example of Fig. 9.6. In this example, consider three messages f_l , g_m and h_k belonging to the same

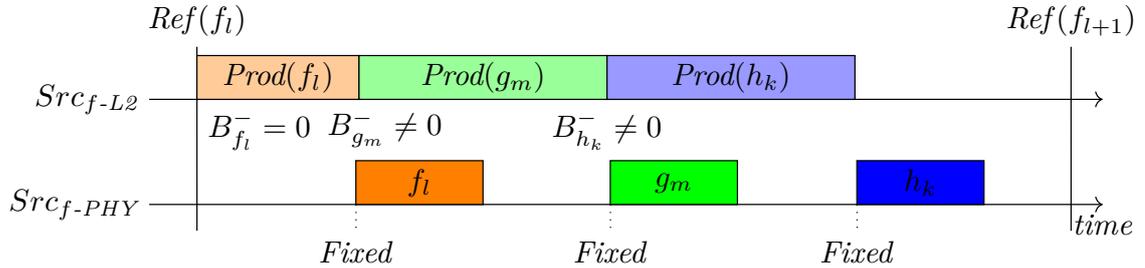


Figure 9.6: End-to-End TT issues with application impact

queue in a last hop port, that are scheduled to be emitted between $Ref(f_l)$ and $Ref(f_{l+1})$, in the order and at the instants represented in the figure. In order to be emitted at their schedule, these messages must be produced before their emission date.

In addition, since they share the same queue, they have to be enqueued in the same order that they will be emitted in. Therefore, the production contract of g_m starts after the end of the production contract of f_l . Identically, the production contract of h_k starts after the end of the production contract of g_m .

Moreover, in systems where message loss independence (see Def. 46) is required, one message must not be emitted in the place of another. In our example, since several messages share the queue, this implies that the production contract of g_m (resp. h_k) must start after the emission of f_l (resp. h_k). We represented the modified production contracts in Fig 9.7. In this situation, it will be impossible for a message to take the slot of its predecessor since it will only arrive in the queue when after the gate closes (following an opening event for the previous message). In the formalism of

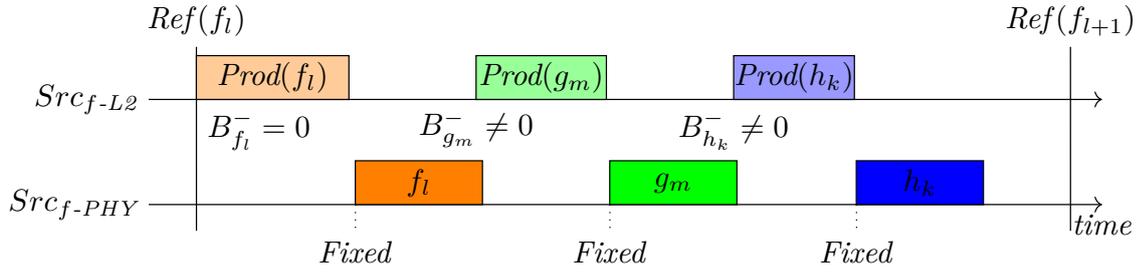


Figure 9.7: End-to-End TT issues with application impact (2)

Section 8.3.2, this entails that, for a majority of messages, $B_{f_l}^- \neq 0$ which, according to Development Requirement 1, has an impact on applications. In fact, tasks have to wait until the beginning of the production contract to produce the message instead of starting at the beginning of the period.

This is a situation that the industrial is aiming at avoiding. Software teams working on applications do not wish to handle constraints induced by the configuration of the underlying network. That is the reason why the existing End-to-End TT configurations do not comply with Development Requirement 1.

9.3.2 Scalability Limitations

As already identified in the state of the art, the computation of End-to-End TT configuration is expensive. In fact, a schedule must be computed for all flows on all hop in their path. In big network configurations with large path and/or large number of flows, scalability is a real issue.

9.3.3 Considerations on Upgrade Costs

In order to work nominally, End-to-End TT configurations require, among other hypotheses, that all devices implement TSN and uses the Time Aware Shaper mechanism. This could hinder the evolution of existing network towards TSN in space. In fact, there is no devices embedding TSN on-board a satellite today, therefore, all the devices would have to be replaced for the upgrade to happen which would represent a considerable cost (design, development, financial, ...).

To conclude, the state of the art provides configurations suitable with the deadline/latency requirements of a next generation satellite network that could be deployed without any concern. The state of the art also proposes configurations compliant with very low jitter requirements, but these configurations come at a cost, that might hinder the introduction of TSN in a satellite context. First, existing End-to-End TT configurations have a significant impact on application development and second, the computation of such configurations is expensive. Finally, End-to-End TT configurations require that all devices in the satellite network implement TSN. This creates a huge gap between network with legacy network devices and a next-generation TSN network.

Therefore, in the following chapters, we will focus on the creation of configurations suitable with the jitter requirements while requiring less computation effort, having a lesser impact on application development and potentially reducing the gap between legacy and next-generation networks.

9.4 Spacecraft Industry Use Cases

Before discussing our contribution with respect to the limitations identified in the previous section, let us introduce two industrial case studies compliant with the system and flow model of Chapter 5 and 8. They will be used in some experiments in this document. The first use case is an Airbus Generic Next-Generation Satellite Use Case that was created during the PhD ([17]) and the second one is ORION Crew Exploration Vehicle (CEV) Use Case retrieved from the state of the art [124].

9.4.1 Airbus Generic Next-Generation Satellite Use Case

This first use case was consolidated during the PhD and formalized in [17]. It describes a generic satellite architecture with a unified network interconnecting both platform and payload devices. The number of devices has been chosen to be representative of several satellite missions (e.g. agile low earth orbit satellite, telecommunication geostationary orbit satellite, scientific mission, etc.).

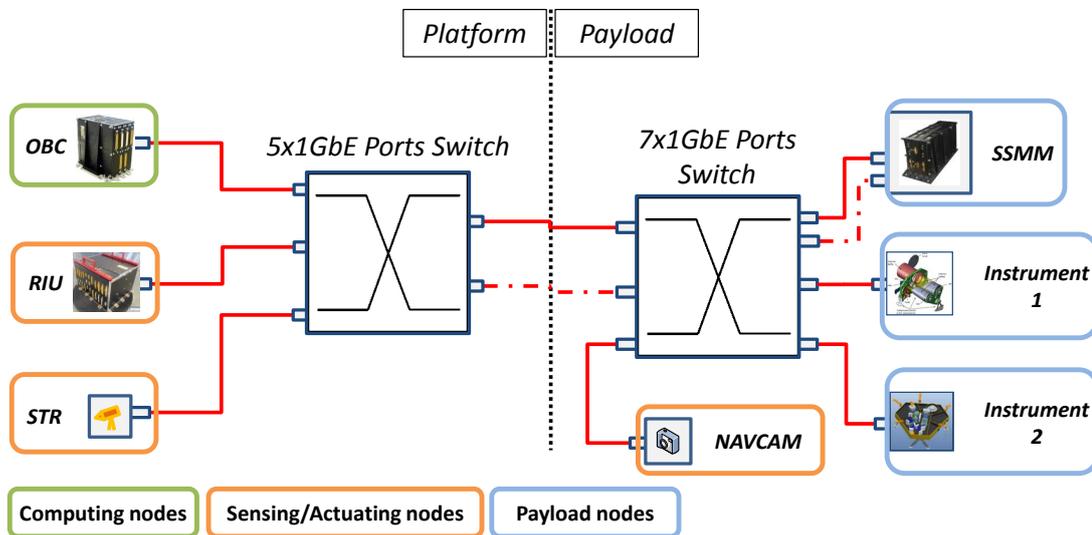


Figure 9.8: Airbus Generic Satellite Use Case Topology

Network topology

The system is composed of:

- One computing device, named *On-Board Computer* (OBC)
- Two sensing devices, named *Star Tracker* (STR) and *Navigation Camera* (NAVCAM)
- One actuating device connected in the network through a *Remote Interface Unit* (RIU)
- Three payload devices, named *Mass Memory* (SSMM), *Instrument 1* (INSTR1) and *Instrument 2* (INSTR2).

These devices are connected together through a set of links and two switches, according to the topology of Fig. 9.8.

Applications

There are three applications:

- *Command & Control (C & C)*
- *Vision Based Navigation (VBN)*
- *Payload*

Command & Control application use case This C & C application is running on the OBC (Platform Computing Device). Flows runs between the OBC, the STR and the RIU. For this application, we choose:

$$k = 8 \text{ hence } P_{MIF} = 125ms \quad (9.3)$$

VBN application use case This VBN application is running on the OBC (Platform Computing Device). Flows runs between the OBC and the NAVCAM. For this application, we choose:

$$k = 30 \text{ hence } P_{MIF} = 33,33ms \quad (9.4)$$

Payload application use case This Payload application is running on the OBC (Platform Computing Device) and on the Payload Devices. Flows runs between the OBC, the INSTR1, the INSTR2 and the SSMM. For this application, we choose:

$$k = 1000 \text{ hence } P_{MIF} = 1ms \quad (9.5)$$

Choice 3 *In the rest of this document, we will only experiment with the C&C application use case.*

The C&C application use case is the only application in which flows have real-time requirements i.e. are critical for the nominal behaviour of the satellite. Therefore, the choice was made to focus first on this application before adding the other applications.

Flow constraints

The flows and their constraints are listed in Appendix B.1 because it is too long and too verbose to be detailed here. There are roughly 120 unicast flows in the C&C use case.

9.4.2 ORION Crew Exploration Vehicle Use Case

This second use case is adapted from the Orion Crew Exploration Vehicle (CEV) use case described in [124]. The network topology is shown in Fig. 9.9.

Network topology

The system we consider is a network composed of 31 devices of undisclosed type (i.e. computing, sensing, actuating or payload) and 15 switches. These devices are connected together through a set of links and switches, according to the topology of Fig. 9.9.

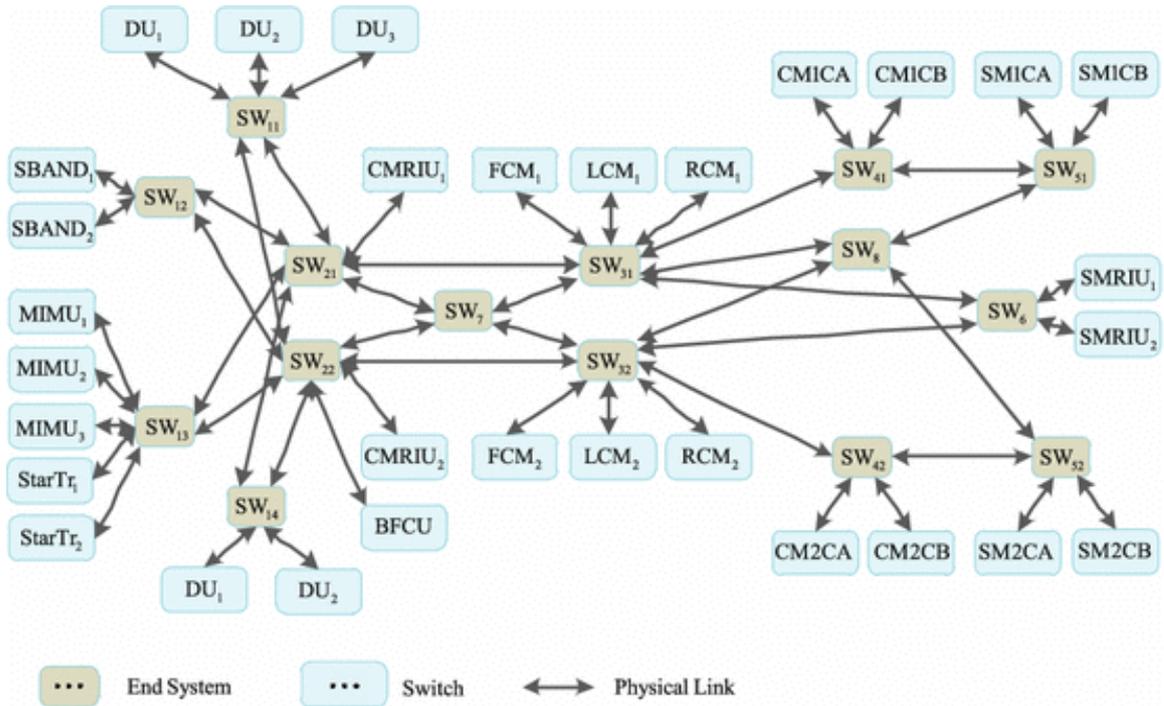


Figure 9.9: Orion Network Topology [124]

Application

We did not dispose of enough information about the applications running over the use case to provide the same level of detail than the Airbus Generic Satellite Use Case. Nevertheless, we have slightly adapted the use case to make it fit into our system model. It is then characterized by $k = 1200$ and $P_{MIF} = 625\mu s$.

Remark 24 In this use case, $P_{MAF} = k * P_{MIF} = 750ms \neq 1ms$.

Flows & Flows constraints

The flows and their constraints for this use case are listed in Appendix B.2. There are 186 flows among which 100 jitter flows (i.e. with jitter and deadline constraints) and 86 no jitter flows (i.e. with deadline constraint only). In our modelling, multicast flows are duplicated into several unicast flows. Thus, the use case is composed of 168 unicast jitter flows and 147 unicast no jitter flows.

Chapter 10

Reduction of the Application Impact and Computation Effort of State of the Art Configuration Generation with Egress TT

In the previous chapter, we have presented End-to-End TT, the most common configuration approach in the state of the art for the configuration of TSN GCL-based networks. These configurations rely on a per-frame schedule on all hops which is translated into a gate control list configuration. In Section 9.3, we have identified the limitations in terms of application impact, scalability and upgrade costs of these configurations with respect to the problem we are dealing with. Therefore, in this chapter, we introduce Egress TT, a new strategy for network configuration inspired from End-to-End TT where a schedule is only computed for the last port in the path of any flow. Egress TT is designed to reduce the limitations of state of the art End-to-End TT configurations i.e. lighter application impact and reduced computation effort for systems with strong jitter requirements but not too stringent latency constraints. In addition, Egress TT is also designed to ease the transition between legacy and next generation networks by requiring less time-triggered-capable devices. We propose two *implementations* of Egress TT configuration generation for TSN GCL-based networks for which we compute configurations with the constraint programming methodology presented in Section 9.2.1. The first one is entitled Exclusive Queue Allocation and the second one, aiming at increasing the schedulability of Exclusive Queue Allocation, is entitled Size Based Isolation. Finally, in the last section, we discuss the limitations of Egress TT and the two proposed implementations.

10.1 Egress TT, a Novel Approach for the Generation of Configurations Suited for Low Jitter Traffic

10.1.1 What is Egress TT?

In End-to-End TT configurations, the support for low jitter traffic is achieved with a network-wide schedule for all frames on all hops i.e. the reception and transmission instants of any frame of any flow in all input/output ports are fixed, and known a priori (see Def. 59). By doing so, the latency

and jitter of all the flows are controlled and nothing unexpected can occur. Fig. 10.1 illustrates an End-to-End TT configuration with one emitter, one receiver and two switches (SWA and SWB).

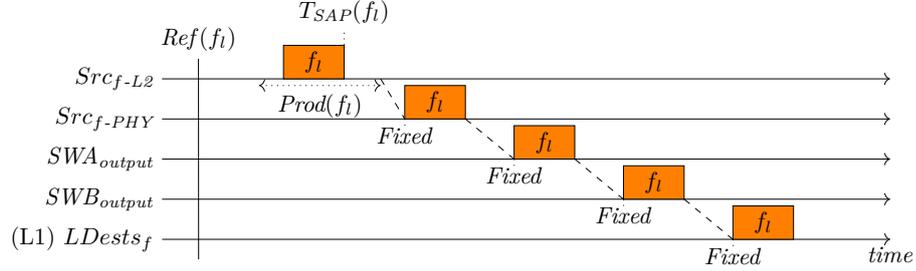


Figure 10.1: End-to-End TT Configuration

One issue with this type of configuration is the high computation effort required to determine the parameters for one valid configuration since a lot of schedules have to be generated for a single configuration. In this context, we introduce Egress TT a new configuration approach. Our idea is the following: in order to satisfy the very low jitter flows requirements, a time-triggered scheduling must be done at some point, but instead of using a schedule on all hops in the network, we propose to schedule frames only on the last hop in their path so as to reduce the computation effort.

Definition 60 (Egress TT Configurations) *In Egress TT configurations, jitter flows are only scheduled in the last hop port in their path. In all other output ports, medium access is not specified. A network designer could choose any medium access strategy as long as a bound on frame latencies can be computed.*

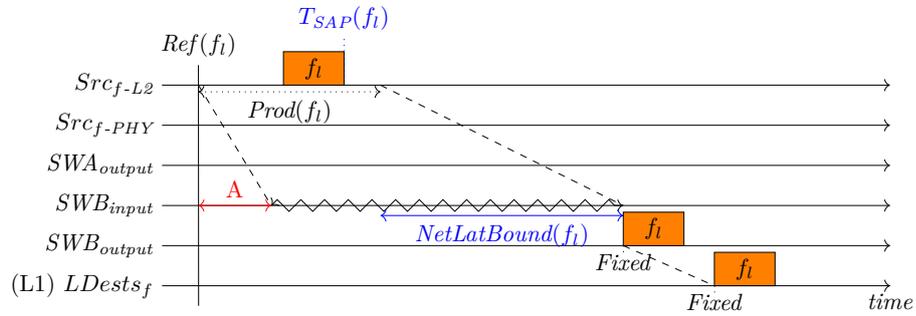


Figure 10.2: Egress TT Configuration

Fig. 10.2 illustrates an Egress TT configuration with one emitter, one receiver and two switches (SWA and SWB), the last hop being the output queue of SWB.

Remark 25 *In the next section, we will apply this approach to the configuration for an Ethernet/TSN network but it could be adapted to any networking technology capable of handling frames in a time triggered way (for the last hop ports).*

Remark 26 *The Egress TT approach (called LETT by [8]) is inspired from the LET – Logical Execution Time - approach [51] where receiving applications can absorb the communication jitter by*

buffering the data up to a nominal fixed release instant. Nevertheless, Egress TT is designed for systems where the network designer do not wish to handle jitter at destination. For instance, in the spacecraft industry, the existing receiving applications do not have the capability to absorb the network jitter. Therefore, in order to avoid redesigning these applications, this capability is relegated to the core of the network which devices are going to be upgraded anyway.

10.1.2 How does it work ?

Let us describe how Egress TT configurations work by studying how a message from a jitter flow f is handled in such configuration.

Per Definition 47, message f_l can be emitted at any time during the interval $Prod(f_l)$. The network traversal delay of f_l can be bounded and let $NetLatBound(f_l)$ be such a bound. We illustrate it in Fig. 10.2 (A represents the best traversal delay). The purpose of these configurations is that whenever f_l is emitted by the application, it will be delivered to the destination end-station at a fixed time.

Definition 61 (*NetLatBound*) *An upper bound on the worst case duration, from deposit (cf. Def. 41) to deposit in the last hop output queue, is denoted $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, NetLatBound(f_l)$.*

Practically, there is no requirements on traffic shaping before the last hop on the path of f as long as a *NetLatBound* bound can be computed. This entails that a message can for instance encounter classical delays due to blocking by other flows (e.g. non-exclusive medium access). The last hop will be in charge of absorbing the upstream network delay variability (i.e. jitter) and delivering the message with low jitter, whatever happened to that message before (be it delayed in the network or waiting in the last hop). To ensure a low jitter reception for f_l , it is sufficient to:

- be received in the correct queue of its last hop port before its schedule,
- be in head of that queue at f_l schedule,
- not be emitted to the destination before its schedule.

In-fine, whenever f_l is emitted by the application, it will arrive at the destination end-station at a fixed time, hence satisfying the very low reception jitter requirements.

10.1.3 What makes Egress TT interesting?

Our novel approach, Egress TT, is interesting for several reasons.

Computation effort

First, it limits the number of schedules to be computed for a single configuration. In Chapter 11, we will assess the computation effort for Egress TT configurations on several use cases. We will observe that this will indeed reduce the computation effort of any configuration.

Application Impact

Second, the concept of Egress TT configuration, where frames are scheduled on the last hop port offers more flexibility for flows' production contracts. In fact, instead of inducing small production windows (like End-to-End TT) due to the strict organisation of messages across the network, Egress TT proposes larger production windows, defined between the frame's reference date and an upper bound (which calculation will be computed later), allowing more flexibility in frames emissions, therefore lighter application impact.

Transition from legacy to next-generation

Finally, Egress TT shall ease the transition from legacy to next generation networks. In fact, only the last hop switch (or device in general) in the path of any flow requires time-triggered scheduling capabilities. Therefore, if the legacy core network is already suitable with the other requirements of the industrial, only the last hop device needs to be upgraded. For instance, Ethernet frames could be sent over an existing Spacewire network and only the last hop device would have to be modified. Therefore, this reduces the gap between legacy and next-generation network compared to End-to-End TT approaches where all devices need to be upgraded.

10.1.4 Suitability of Egress TT

The Egress TT approach is well suited for systems where very low reception jitter is a strong requirement and latency is less of a strong requirement. While Egress TT configurations offers some freedom in the core of the network, latency is traded for jitter. In fact, if a message is emitted at the beginning of its production contract and benefits from a favourable situation in the core network (between the emitter and the last hop device), it will have to wait a "long" time in its last hop port before being delivered to the receiving end-system and application. Therefore, very low reception jitter is achieved at the cost of a greater latency. While this is compatible with the requirements of the spacecraft industry where latency constraints are large, Egress TT configuration might not be suitable for systems where latencies shall be minimal.

In the next section, we will show how the Egress TT approach can be applied to TSN networks with a first implementation of Egress TT configuration generation: Exclusive Queue Allocation.

10.2 A First Implementation of Egress TT in IEEE 802.1Qbv Networks: Exclusive Queue Allocation

Definition 62 (Implementation) *In this manuscript, we use the word Implementation to describe a set of constraints used for the generation of network configurations. Therefore an implementation of Egress TT applied to TSN networks is a set of constraints linked to a TSN configuration model (see Section 9.1), that will generate Egress TT configuration with the constraint programming methodology introduced in Section 9.2.1. Implementation does not refer to the actual device configuration on a mock-up.*

This first implementation of Egress TT is based on the Exclusive Queue Allocation constraint which specifies that no two flows with low jitter requirements can share the same queue. We describe it formally here after.

10.2.1 Adaptation of Egress TT Configurations to TSN Networks

Let us now formalize how to compute valid Egress TT configurations for TSN networks according to the methodology of Section 9.2.1.

Hypothesis 7 (Synchronization) *We assume that emitters and last hop devices are synchronized and the synchronization error is insignificant with respect to the order of magnitude of the requirements presented hereafter.*

Hypothesis 8 (Fixed Path) *As in the state of the art, we assume that the routing of the flows in the network is fixed and static.*

Choice 4 (Ethernet Static Priority) *In the rest of this manuscript, we chose Static Priority, conveniently available in standard Ethernet (cf. Section 3.6.1) as medium access strategy in non-last-hop ports.*

Choice 5 (TSN Time Aware Shaper) *In the rest of this manuscript, we chose TSN Time Aware Shaper as the scheduling mechanism for last hop ports.*

Remark 27 *The above choices entail that no TSA will be used in the configuration process.*

We now formally define the concept of *Last Hop ports*.

Last hop port

We distinguish the output ports which are the last hop of some flows from the others.

Definition 63 (Last Hop Ports) *For a flow f following the path p_1, \dots, p_l , we denote by $LH_f = p_l$ the last hop port. The set of last hop ports is $\mathbb{LH} = \{p \in \mathbb{P} \mid \exists f \in \mathbb{F}, LH_f = p\}$ and the set of last jitter ports is $\mathbb{LH}_j = \{p \in \mathbb{P} \mid \exists f \in \mathbb{F}_j, LH_f = p\}$ ¹.*

Port configuration

The configuration for the ports $\mathbb{P} \setminus \mathbb{LH}_j$ is equivalent to Ethernet-capable port configuration i.e. their gates are always open. Medium access is handled with Static Priority.

$$\forall p \in \mathbb{P} \setminus \mathbb{LH}_j, GCL(p) = \langle \langle o, o, o, o, o, o, o, o \rangle, 0, P_{MAF} \rangle$$

When a port is not a last hop, there is nothing much else to do, hence we must now focus on the configuration of *Last Hop Ports*. In fact, in any last hop, that is in port $p = (q_0, \dots, q_7) \in \mathbb{LH}_j$, gate schedules follow an *exclusive gating pattern* [28]:

- jitter flows and no jitter flows are placed in different queues;
- At any time, either exactly only one jitter associated queue gate is *open* or several no jitter associated queues gates are *open*;
- if q_i is allocated to a jitter flow f : the gate is closed almost all the time. It is opened when a message f_i is scheduled and remains open during the message transmission duration (τ_{f_i});
- if q_i is allocated to no jitter flow(s): the gate remains always open except when one jitter associated queue is open.

Decision variables

The decision variables, i.e. the variables to which we are trying to find a value, should be those of equation 9.1, but thanks to hypothesis 8 (fixed flow routing), remark 27 (no TSA), we simplify equation 9.1 into equation 9.2. Hence the variables are the flow to queue mapping and the gate control list schedule for all output ports. Instead of computing GCL (see. Section 9.1.1) directly, like in the state of the art, we introduce an intermediate decision variable $SchedLH$.

¹As a reminder, \mathbb{F}_j represents the set of jitter flows i.e. with very low jitter constraints

Definition 64 (SchedLH) Let $f_l \in \mathbb{F}_j$ a jitter message, $SchedLH[f_l]$ denotes the instant at which the gate $FtQM_{LH_f}(f)$ shall be opened.

Remark 28 (Link SchedLH[f_l], T_r(f_l), Lat_{f_l}) With this intermediate decision variable, $SchedLH[f_l]$ is linked to $T_r(f_l)$ by the following formula:

$$T_r(f_l) = SchedLH[f_l] + \tau_{f_l}$$

Therefore:

$$Lat_{f_l} = T_r(f_l) - Ref(f_l) = SchedLH[f_l] + \tau_{f_l} - Ref(f_l)$$

From the variables $SchedLH$, it is possible to reconstruct GCL . Indeed, let us consider a jitter flow f and its last hop $LH_f = (q_0, \dots, q_7) \in LH$. f will produce P_{MAF}/P_f events: every time a frame of f is supposed to be transmitted, the gate should be open. More practically, for each f_l , there is an event $e = \langle s, SchedLH[f_l], \tau_{f_l} \rangle$ where $s = \langle s_0, \dots, s_7 \rangle$ with $s_{FtQM_p(f)} = o$ and $s_j = C$ for $j \neq FtQM_p(f)$. Thus GCL is the union of all events associated to all jitter messages f_l where $LH_f = p$. This union is completed with gate opening of queues not allocated to jitter flows on the remaining time (when the jitter associated queue gates are closed). The gate events are generated by a post processing procedure.

Remark 29 Since the system we consider is periodic, it is only necessary to compute $SchedLH[f_l]$ on one period of the system (P_{MAF}).

Finally, the decision variables for the problem become:

$$\begin{cases} \forall f \in \mathbb{F}, \forall p \in Path_f, FtQM_p(f) \\ \forall f \in \mathbb{F}_j, \forall l < \frac{P_{MAF}}{P_f}, SchedLH[f_l] \end{cases} \quad (10.1)$$

10.2.2 Exclusive Queue Allocation Concept

In order to satisfy the safety requirement in Egress TT configurations for TSN networks, instead of reusing existing constraints (i.e. Flow/Frame Isolation) from the state of the art which lead to high application impact configurations, we introduce our own "isolation" constraint: Exclusive Queue Allocation.

Definition 65 (Exclusive Queue Allocation) With Exclusive Queue Allocation, each jitter flow is paired with one dedicated queue in its last hop port. No other flow can use that queue.

Being alone in the queue removes the possible non-determinism induced by TSN Time Aware Shaper mechanism (see Section 9.2.2).

Indeed, if a jitter message is lost in (or before) the last hop port, since the medium access for jitter queue is done via exclusive gating, no messages for other queues will take its place (and suffer from unwanted jitter as illustrated in Fig. 9.5b). Within the same queue, it could still lead to unwanted jitter but this is compatible with safety requirement 1, since in a specific queue, there can only be frames from the same flow.

10.2.3 Constraints Formalization for Exclusive Queue Allocation

Across the network, the Flow to Queue Mapping rule will not always be the same. In all ports except last hop ones, jitter and no jitter flows are allowed to share the same queue. However, Exclusive Queue Allocation is applied in last hop ports for jitter flow. The other flows can share their last hop ports.

Remark 30 (Macrotick) To simplify the formulation of the equations in the rest of the paper, the instants and durations will be written in macroticks (like [25]). For instance, if the macrotick is the necessary duration to transmit a frame of 64 bytes, then $\text{Size}_f = 128 \implies \tau_{f_l} = 2$ (reminder: τ_{f_l} is the transmission duration of f_l , defined in Def. 40).

Constraint 1 (Exclusive Queue Allocation) Each jitter flow is associated with one dedicated queue.

$$\forall f \neq g \in \mathbb{F}_j, LH_f = LH_g \implies FtQM_{LH_f}(f) \neq FtQM_{LH_f}(g)$$

Links are modelled for the solver as two unidirectional links with opposite directions.

Constraint 2 (Link Occupation) A link can only send a message at a time in one direction i.e. $\forall f_l, g_m \in \mathbb{F}_j$ s.t. $LH_f = LH_g$:

$$\text{SchedLH}[f_l] + \tau_{f_l} < \text{SchedLH}[g_m]$$

or

$$\text{SchedLH}[g_m] + \tau_{g_m} < \text{SchedLH}[f_l]$$

Performance Constraints. All jitter flows are subject to deadline and jitter constraints.

Constraint 3 (Ordered Delivery) For any jitter flow, the i -th message shall be delivered before the $(i+k)$ -th message of that flow, i.e. $\forall f \in \mathbb{F}_j \forall l, m \in \mathbb{N}, l < m, T_r(f_l) < T_r(f_m)$. This is translated as $\forall f \in \mathbb{F}_j \forall l, m \in \mathbb{N}, l < m$:

$$\text{SchedLH}[f_l] < \text{SchedLH}[f_m]$$

Constraint 4 (Deadline) The delivery instant of a flow is bounded, indeed $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, \text{Ref}(f_l) \leq T_r(f_l) \leq f_l.\text{deadline}$. This is translated as:

$$\forall f \in \mathbb{F}_j, \forall l \in \mathbb{N}, \text{Ref}(f_l) \leq \text{SchedLH}[f_l] + \tau_{f_l} \leq f_l.\text{deadline}$$

Constraint 5 (Jitter) For any jitter flow, the difference of latency of any two messages is bounded by the flow's jitter constraint i.e. $\forall f \in \mathbb{F}_j, \forall i \neq j \in \mathbb{N}, |\text{Lat}_{f_i} - \text{Lat}_{f_j}| < f.\text{jitter}$. This is translated as $\forall f \in \mathbb{F}_j, \forall i \neq j \in \mathbb{N}$:

$$|\text{SchedLH}[f_i] - \text{Ref}(f_i) - (\text{SchedLH}[f_j] - \text{Ref}(f_j))| < f.\text{jitter}$$

Last Hop associated Constraints. In order to compute the last hop schedule, it is necessary to have an upper bound NetLatBound on the traversal time of flows until their last hop port.

Remark 31 NetLatBound could be estimated with classical worst case traversal time method such as Response Time Analysis [85] or Network Calculus [125], or any other methods as long as it can be integrated in the constraint programming solver. In this manuscript, a bound NetLatBound is estimated with Response Time Analysis [9] because this method was directly implementable as a constraint in the solver. In fact, the solver needs to compute NetLatBound with every new configuration since NetLatBound depends on decision variables. As mentioned earlier, any other methods could be used to estimate that bound as long as it can be either integrated in constraints or coupled with the solver.

Constraint 6 (Traversal Time Constraint) The release instant of any message of a jitter flow shall be within the flow's period. This is expressed as $\forall f \in \mathbb{F}_j, \forall l \in \mathbb{N}$:

$$\text{Ref}(f_l) \leq \text{SchedLH}[f_l] - \text{NetLatBound}(f_l) < \text{Ref}(f_{l+1})$$



Figure 10.3: Release instants for jitter flows

Consider a jitter flow f , and its l -th message f_l . With Egress TT configurations, in order to ensure f_l arrives in $FtQM_{LH_f}(f)$ before its schedule, the message must be sent after its release and before some bound.

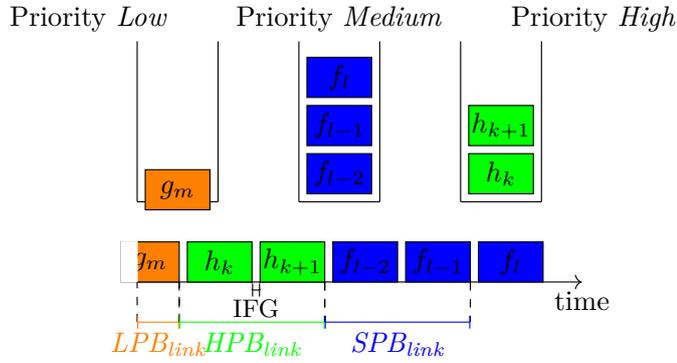
We now explain how $NetLatBound$ is computed.

Definition 66 (Bound on worst case latency $NetLatBound$) A bound on the worst case deposit to last hop emission latency is computed as $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}$:

$$NetLatBound(f_l) = \sum_{p \in Path_f, p \neq LH_f} \Delta(f_l, p) + \tau_{f_l}$$

where $\Delta(f_l, p)$ is a bound on the worst case duration for f_l at output port p .

In any port, f_l can be delayed, in the worst case, by several other messages. First, f_l can be delayed by all messages with same or higher priority than f_l but also one lower priority frame which arrived in the port before f_l . These delays are known as higher priority blocking $HPB(f_l, p)$, same priority blocking $SPB(f_l, p)$ and lower priority blocking $LPB(f_l, p)$. This delay model is inspired from [115, 7, 117]. We illustrate the notion of higher, same and lower priority blocking for a frame f_l in Fig. 10.4².

Figure 10.4: LPB_{link} , SPB_{link} and HPB_{link} example

Definition 67 (Bound on worst case duration $\Delta(f_l, p)$) $\Delta(f_l, p)$ is defined as $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, \forall p \in Path_f, p \neq LH_f$:

$$\Delta(f_l, p) = \frac{HPB(f_l, p) + SPB(f_l, p) + LPB(f_l, p)}{r}$$

²Note that this figure is identical to Fig. 7.1. We put it again in this section for easier readability

It is now necessary to determine which messages will be accounted for in HBP, SPB and LBP. In our system, all messages have a deadline smaller or equal to the end of their period (*implicit deadlines*). Therefore, a finite number of instances (i.e. frames) of each flow may be considered interfering with any message of a defined flow (cf. [111, 10]).

Definition 68 (List of contributing flows [10]) Let $FlowPort(p)$ be the set of all the flows whose path includes p i.e. $\forall p \in \mathbb{P}, FlowPort(p) = \{f \in \mathbb{F} | Path_f \cap p \neq \emptyset\}$.

Property 4 (Number of interfering instances) On any given port p , for any message f_l of $f \in FlowPort(p)$, for every flow $g \in FlowPort(p) \setminus \{f\}$, there is at most $\lceil \frac{P_f}{P_g} \rceil + 1$ instances of flow g taking part in the delay of f_l . This is illustrated in Fig. 10.5.

Proof 1 In fact, since in our model, flows are periodic and have implicit deadlines, we claim that there are at most $\lceil \frac{P_f}{P_g} \rceil + 1$ instances of flow g in the whole network. Therefore, since there cannot more than $\lceil \frac{P_f}{P_g} \rceil + 1$ instances in the whole network, there cannot be more than the same number of instances on any given port.

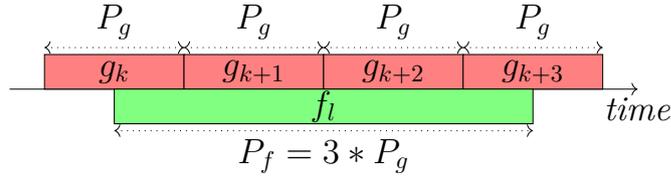


Figure 10.5: Contributing instances of g for the delay of f_l

Remark 32 The considerations of Prop. 4 may seem a little bit too excessive with respect to a decent worst case delay analysis. However, we believe it is sufficient to demonstrate the concept of Egress TT. Improving this bound would of course lead to an improvement of the performances of our approach (in particular for the integration of no jitter traffic).

Remark 33 (Impact of common period start) In our model, there is no offset between periods i.e. any period starts at the beginning of a MIF cycle. Therefore, the number of interfering instances is reduced to $\lceil \frac{P_f}{P_g} \rceil$.

Definition 69 (Blocking durations) We formulate the priority blocking durations as follows: $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, \forall p \in Path_f, p \neq LH_f$:

$$\begin{aligned}
 HBP(f_l, p) &= \sum_{g \in FlowPort(p) | FtQM_p(g) > FtQM_p(f)} \left(\lceil \frac{P_f}{P_g} \rceil \right) * Size_g \\
 SPB(f_l, p) &= \sum_{g \in FlowPort(p) | g \neq f, FtQM_p(g) = FtQM_p(f)} \left(\lceil \frac{P_f}{P_g} \rceil \right) * Size_g \\
 LPB(f_l, p) &= \max_{g \in FlowPort(p), FtQM_p(g) < FtQM_p(f)} Size_g
 \end{aligned} \tag{10.2}$$

10.2.4 Focus on the Computation of Release Instants

Optimization criteria The problem has been encoded as a set of decision variables and a set of constraints to be solved by a constraint solver. Egress TT has been designed to provide a lighter application impact than End-to-End TT. The development effort requirement is ensured with an optimization criteria: maximize the production window of the configurations (i.e. minimize the application impact), we encode this criteria in equation 10.3.

$$\underset{\forall f \in \mathbb{F} \text{ s.t. } f.\text{jitter} \neq NA}{\text{maximize}} \sum (Release(f_i) - Ref(f_i))^2 \quad (10.3)$$

Choice 6 We chose a quadratic cost function to reduce the solution to homogeneous solutions only, where no flow is compensating for another flow (e.g. one flow has a tiny production window and another flow compensate with a huge one) as requested by our industrial use case. This function could be modified depending on specific system requirements.

Therefore, this requires to compute the release instants for both jitter and no jitter flows.

Definition 70 *Release*(f_i) for jitter flows. $SchedLH[f_i]$ occurs exactly later after the worst case duration of f_i compared to $Release(f_i)$. Therefore:

$$\forall f \in \mathbb{F}_j, \forall l \in \mathbb{N}, Release(f_i) = SchedLH[f_i] - NetLatBound(f_i)$$

Definition 71 *Release*(f_i) for no jitter flows. $Release(f_i)$ is computed a posteriori via a post processing. Once the last hop emissions instants for jitter flows have been decided, the scheduling instants of no jitter flows are decided with the remaining port capacity (i.e. when gates for jitter flows are closed).

Remark 34 Because the release instant for no jitter flows is computed a posteriori, it is necessary to check the correctness of that release instant that is $\forall f \in \mathbb{F} \setminus \mathbb{F}_j, \forall l \in \mathbb{N}, Release(f_i) \geq Ref(f_i)$.

The last hop gate of no jitter flows is always open (except when some jitter message is being emitted and the output port is its last hop). Moreover several no jitter messages may be in the same queue at the same time. Thus, the release instant is defined as $\forall f \in \mathbb{F} \setminus \mathbb{F}_j, \forall l \in \mathbb{N}$:

$$Release(f_i) = f_i.\text{deadline} - NetLatBound(f_i) - \Delta_{LH_f}^{WC+closed}(f_i)$$

where $\Delta_{LH_f}^{WC+closed}(f_i)$ denotes the worst case duration needed to transmit, in port LH_f , in queue $FtQM_{LH_f}(f)$, no jitter message f_i , including time for which the gate of $FtQM_{LH_f}(f)$ is closed.

In order to cope with the different size of messages in the queue, a gate will be considered open if and only if it can fit more than the biggest message assigned to that queue.

Definition 72 ($e_n.IsOpen(t)$) Considering an event $e_n = \langle s_n, t_n, d_n \rangle$ from $GCL(p)$ and a date t , $e_n.IsOpen(t)$ returns true if between t and the beginning of e_n , there is enough time to transmit a maximum size frame from the set of frames travelling through p . Mathematically,

$$e_n.IsOpen(t) = ((t - t_n) > \max_{g \in FlowPort(p)} (Size_f))$$

A bound on the worst case duration $\Delta_{LH_f}^{WC+closed}(f_i)$ is determine with an algorithm not disclosed in this paper.

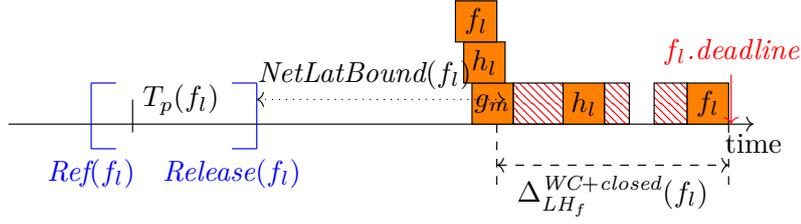


Figure 10.6: Release instant for no jitter flows

Definition 73 (Worst case duration $\Delta_{LH_f}^{WC+closed}(f_i)$) A bound on the worst case duration $\Delta_{LH_f}^{WC+closed}(f_i)$ is defined as

$$\Delta_{LH_f}^{WC+closed}(f_i) = \text{Compute}(\Delta(f_i, LH_f) * r, f_i.\text{deadline}, \\ FtQM_{LH_f}(f), \max_{f \in \text{FlowPort}(LH_f)} (\text{Size}_f))$$

where $\text{Compute}(\text{Remaining}, \text{Start}, \text{Queue}, \text{UnitSize})$ is a recursive function. The first argument is a number of bytes Remaining, the second argument is an instant Start, the third one is a queue Queue and the last one UnitSize is the maximum size of a frame going in that queue. The principle of that function is quite simple: starting at Start, the function goes back in time up to having enough open slots to send $\lfloor \frac{\text{Remaining}}{\text{UnitSize}} \rfloor$ in queue Queue. Since the order of frames received in Queue is unknown a priori, instead of considering the exact size of all the frames, in Compute function, we consider that all frames are UnitSize long.

Wrap-up

The computation of release instants concludes the network configuration with our first Egress TT implementation with Exclusive Queue Allocation. We can generate schedules for frames (i.e. $\text{SchedLH}[f_i]$), that are transformed into GCL events (i.e. $GCL(p)$) as well as an assignment of flows into queues (i.e. $FtQM_{LH_f}(f)$), that were the required elements for the configuration of the network. In addition, we have provided the network designer with information on the available production contract for all flows in the network (i.e. $\text{Release}(f_i)$) which he can use to design the applications running over that network.

Exclusive Queue Allocation comes with one strong limitation: a last hop port can only receive up to 8 different jitter flows (or 7 jitter flows with additional no jitter traffic). While this might be sufficient for small networks with a relatively low number of jitter flows, it is not scalable enough for the configuration of large networks such as Orion CEV (cf. Section 9.4.2). Therefore, in the next section, we introduce Size Based Isolation, a second implementation of Egress TT for 802.1Qbv network aiming at reducing the limitation on number of flows of Exclusive Queue Allocation.

10.3 A Second Implementation of Egress TT with Improvement of the Schedulability of Exclusive Queue Allocation:Size Based Isolation

10.3.1 Size Based Isolation Concept

In the Exclusive Queue Allocation constraint, each jitter flow was paired with one dedicated queue in its last hop port and no other flow could use that queue. Being alone in the queue removed the possible non-determinism induced by TSN Time Aware Shaper mechanism (see Section. 9.2.2). However, respecting Exclusive Queue Allocation came at a cost: an end-station cannot receive more than eight jitter flows (or seven if it also receives no jitter traffic). With Size Based Isolation, we relax that constraint so that several messages from different flows are allowed to exist in the queue at the same time. However, it is necessary to manage the messages behaviour to satisfy the safety objective.

Definition 74 (Size Based Isolation) *All frames sharing the same queue on last hop port shall be enqueued in increasing frame size order. This size may be achieved with the help of padding.*

By ensuring that frames are enqueued in increasing order, if a frame is lost, the following frame will not be emitted in the slot of the lost frame since its size is bigger than the opening of the gate. Instead, the frame will be, as expected, emitted in its allocated slot. This concept is illustrated in Fig. 10.7: we show the nominal situation in 10.7a and the behaviour in case of message loss in 10.7b. Even when f_l is lost, j_k is not sent in place of f_l .

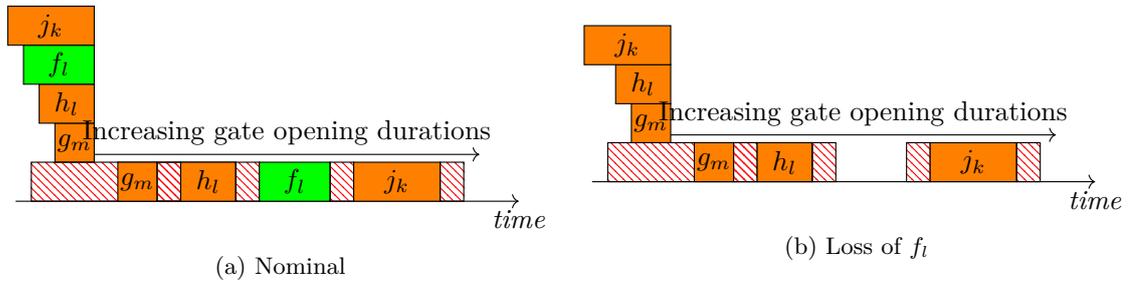


Figure 10.7: Isolation by Message Size

Being unable to impose an order between messages coming from different sources in the last hop port without a negative impact on the application development, we impose that flows sharing a queue in a last hop port shall :

- come from the same emitter,
- and share the same route.

In this situation, the application impact will be slightly increased: in addition to the traffic contract, the emitter will have to ensure an emission order.

10.3.2 Constraints Formalization for Size Based Isolation

Let us formulate the constraints for Egress TT with Size Based Isolation.

Hypothesis 9 *Let us assume that when two flows have the same source and same destination that they share the same route i.e. $\forall f, g \in \mathbb{F}, Src_f = Src_g \text{ and } Dest_f = Dest_g \implies Path(f) = Path(g)$.*

Rationale 1 *This hypothesis is representative of the Airbus generic satellite system presented in Section 9.4.1.*

Remark 35 *If this hypothesis is relaxed, the model could be modified so that the order of deposit in the last hop queue required for Size Based Isolation is maintained, probably with the help of additional constraints.*

Additional Decision Variable

We add an additional decision variable $Padd$ which translates the additional padding used to increase the size of frames.

Definition 75 ($Padd_{f_l}$) *Let f_l a message of f , we define $Padd_{f_l}$ as an additional amount of bytes that is used to increase the size of f_l . In particular, we have:*

$$\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, Size_f \leq Size_f + Padd_{f_l} < MTU_{Ethernet}$$

Remark 36 *The above formulation is generic and allows to take into account systems where periods are not harmonic. In our model, since periods are harmonic, it would be possible to only compute a padding per flow instead of a padding per frame.*

Constraints

We now extend the definition of the transmission duration of f_l i.e. τ_{f_l} ,

$$\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, \tau_{f_l} = \frac{Size_f + Padd_{f_l}}{r}$$

Then, we reuse all the constraints from the first approach (i.e 2, 3, 4, 5 and 6) except Constraint 1. In addition we define two new constraints: *Size Based Isolation* and *QueuePerEmitter*.

Definition 76 ($Queue_p(i)$) *Let $Queue_p(i)$ define the set of flows sharing the same last hop port and the same queue in that port i.e.*

$$\forall p \in \mathbb{P}, \forall i \in [0, 7], Queue_p(i) = \{f \in \mathbb{F} | LH_f = p \text{ and } FtQM_p(f) = i\}$$

Definition 77 ($f_l \# g_m$) *Let $f_l \# g_m$ denote that f_l and g_m can interfere with one another i.e. that they exist in the same queue at the same time. Therefore,*

$$f_l \# g_m \implies \max(Ref(f_l), Ref(g_m)) < \min(f_l.deadline, g_m.deadline)$$

Constraint 7 (Size Based Isolation) *All interfering messages in a last hop port shall be enqueued and transmitted in increasing message size order on last hop i.e.*

$$\forall f, g \in Queue_p(i), \forall f_l, g_m \text{ s.t. } f_l \# g_m, SchedLH[f_l] < SchedLH[g_m] \implies \tau_{f_l} < \tau_{g_m}$$

In order to be able to control the reception order in the last hop port, we define an additional constraint:

Constraint 8 (Queue Per Emitter) *Any two jitter flows having different source and same destination will be placed into the different queues i.e.*

$$\forall f, g \in \mathbb{F}_j \text{ s.t. } LH_f = LH_g, Src_f \neq Src_g \implies FtQM_{LH_f}(f) \neq FtQM_{LH_f}(g)$$

The newly computed configurations allow a greater number of jitter flows per port; but not without a cost. In addition to the release date, the emitting applications must follow an order constraint on emission so that their messages arrive in the correct order in the last hop port.

10.4 Limitations of Exclusive Queue Allocation, Size Based Isolation and Egress TT

First, Egress TT with Exclusive Queue Allocation will always fail to find configurations when a device is supposed to receive more than 8 jitter flows. Indeed, this comes from the exclusive queue allocation since a queue is dedicated to one jitter flow.

Then, Egress TT with Size Based Isolation will always fail when a device is set to receive flows coming from more than eight different sources. Again, this is due to the Size Based Isolation constraint and our objective to keep the application impact relatively low. In addition, the number of low-jitter flows per queue with Size Based Isolation will be limited by the gate granularity i.e. the smallest duration of a gate event. The maximum number of jitter frames that can be held in a queue at the same time $\chi_p(i)$ is computed with the following formula: $\forall p \in \mathbb{LH}_j, \forall i \in [0, 7], \text{ if } \exists f \in \mathbb{F}_j \in \text{Queue}_p(i), \chi_p(i) = \lceil \frac{\text{Max}_{size}}{\text{gcd}(d_j)} \rceil$ where $\text{gcd}_{j \in [0, u-1]}(d_j)$ is the gate granularity (see Def. 53). For instance, with a granularity of $1\mu s$, the smallest open event will be able to transmit 125 bytes (i.e. $\frac{1\mu s}{8} * r$). Therefore considering the maximum frame size is 1518 bytes, this means that a queue can hold up to 13 (i.e. $\lceil \frac{1538}{125} \rceil$) low-jitter frames.

Egress TT will fail when the post processing on no jitter flows fails (i.e. deadlines of no jitter flows cannot be met).

Finally, Egress TT will fail when the computation of $\text{NetLatBound}(f_l)$ becomes too pessimistic: over-reservation of resources (Egress TT) is always less scalable than exact allocation (End-to-End TT).

In the above situations, among others, End-to-End TT will always be a better approach. Nevertheless, we believe that the expected improved scalability, in particular the shorter configuration time, as well as the per design lower application impact will still attract industrials towards Egress TT.

Conclusion

In this chapter, we presented Egress TT, a new scheduling strategy for systems with very low reception jitter requirements and medium latency constraints. Then we proposed Exclusive Queue Allocation and Size Based Isolation two implementations of Egress TT for TSN 802.1Qbv networks. We first detailed the concept of Egress TT, and then applied the methodology introduced in the state of the art by defining a set of constraint for a constraint programming solver. We expect from Egress TT configurations better scalability, lighter application impact and easier transition from legacy to next-generation. In order to assess the performance (compare to what we expected in Section 10.1.3), we evaluate in the next chapter our two implementations of Egress TT, namely Exclusive Queue Allocation and Size Based Isolation as well as an implementation of state of the art End-to-End TT on several use cases.

Chapter 11

Experimentations: Performance Comparison of Egress TT and End-to-End TT Configurations

In this chapter, we propose to assess the performance of the Egress TT configurations introduced previously compared to our own implementation of state of the art End-to-End TT approach. In the first section, we give an overview of the software architecture to compute TSN configurations that was developed during this PhD. Then, in the second section, we introduce two comparison axes i.e. scalability and latencies and evaluate them on small use cases. We compare the obtained metrics between End-to-End TT and Egress TT. Finally, in the last section, we compare the performance of Egress TT and End-to-End TT on the "larger" use cases presented in Section 9.4.1 and 9.4.2.

11.1 Software Architecture

Fig. 11.1 presents the overall architecture of all the software elements that we developed for the configuration process. It is organized in four parts: *Data Conversion*, *Configuration*, *Simulation* and *Validation*. Let us describe briefly the purpose of each part.

11.1.1 Data Conversion

The *Data Conversion* part is in charge of transposing the system that needs to be configured into the model defined in Section 9.2.1. Practically, the network designer provides the *NetworkDescriptor.csv* file, which is our standardized input used to define the system. It includes the description of the devices, of the links and of how they are connecting the devices with one another, the flows and their constraints. As in classical constraint generation softwares, we rely on *.dat* and *.mod* files to describe the system and its associated constraints. In our case, the *NetworkDescriptor.csv* file is converted with the *ConvertToDat* function, encoded in Python, into a *.dat* file usable in the constraint programming solver. A *GenerateNetworkModel* function, encoded via the ILOG-CPLEX scripting language [83], generates, based on the *.dat* file, all the input *.mod* files describing the model and the constraints required for the configuration generation.

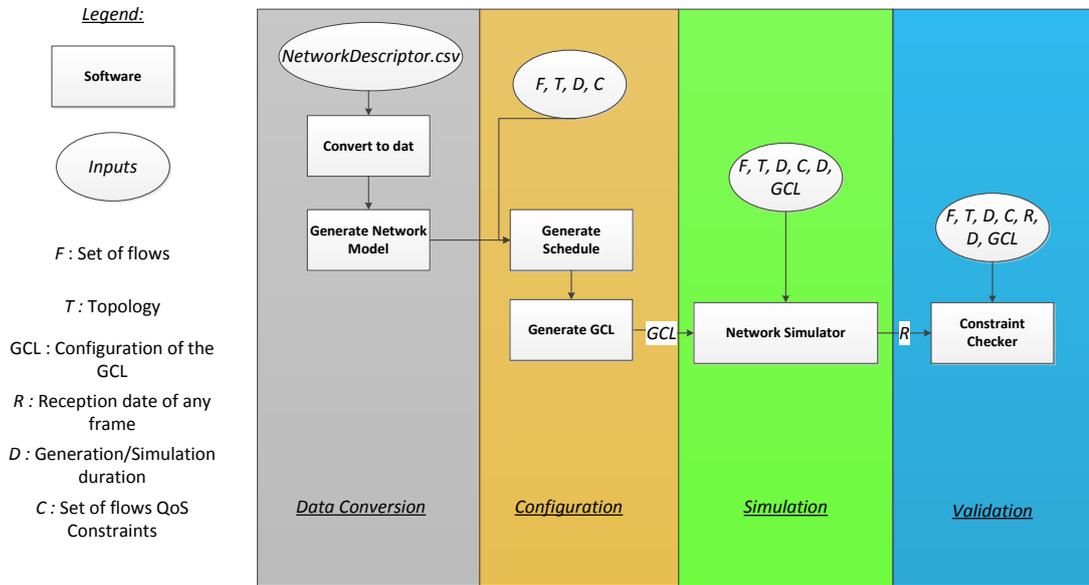


Figure 11.1: Software Architecture

11.1.2 Configuration

Then, the *Configuration* part, is the core function in this software architecture. It is in charge of effectively generating gate schedules via the constraint programming methodology. The *GenerateSchedule* function takes as input the .dat and .mod files describing the model and the constraints. It produces one or several files proposing a valid network configuration. The function was implemented in OPL and executed with the help of IBM CPLEX solver. This network configuration is processed/refined with the *GenerateGCL* function which converts the network configuration files into TSN Gate Control Lists configuration files that will be used in the Simulator.

11.1.3 Simulation

The *Simulation* part is in charge of providing a simulation capability for the configurations generated in the configuration part. The *NetworkSimulator* function is used to verify that the proposed configuration is indeed valid with respect to the quality of service requirements. It is a Python simulator developed during the PhD because it was more easily modifiable compared to existing on-the-shelf tools.

11.1.4 Validation

Once the system has been simulated during one hyper-period (i.e. P_{MAF}), the *ConstraintChecker* function checks that every constraint is satisfied.

Remark 37 This software also has scripts to interface with RTAW Pegase [99] configuration and simulation tool, so that the configurations proposed by the RTAW tool can be simulated and/or verified. In particular, this interface has been used for an internship (master level) during the PhD.

11.2 Latency and Scalability Comparison

The purpose of this section is to evaluate our approach and also to compare it with the state of the art. The comparison will be based on two criteria: *scalability* and *network latency*. The software architecture for configuration generation, simulation and validation is briefly introduced in Appendix. 11.1.

Both approaches were implemented in OPL and the computation was done using CPLEX v12.9.0 running on a Ubuntu computer embedding *Intel Xeon E5-2600 v3 @ 2.6GHz* and 62GiBytes of memory.

Remark 38 For all the experiments of this document, the network devices and links are supposed to work at 1Gbit/s.

11.2.1 Scalability

In this section, we use the sets of constraints without the optimization criteria for Egress TT and our implementation of End-to-End TT with *Frame Isolation*. We compare the results provided by the solver for both approaches. To assess the scalability, we increase the number of switches on the paths and the number of receivers, and we monitor two metrics:

- *Number of constraints* necessary to generate a configuration,
- *Duration* of the computation to find a configuration.

Path size increase

In this first set of experiments, we consider a simple topology with one emitting end-station and one receiving end-station connected with a set of switches from 1 to 10 switches (cf. Fig. 11.2). This allows to quantify the computation cost when adding a switch in the path.

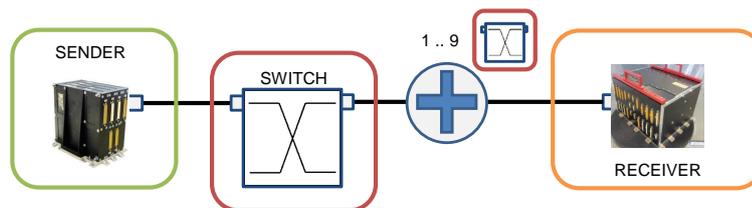


Figure 11.2: Path size increase use case

Table 11.1 showcases the set of flows and their constraints. All flows have deadlines equal to their period. This set of flows is inspired from the *Airbus Generic Next-Generation Satellite Use Case* (see. Section 9.4.1).

Table 11.1: Set of flows \mathbb{F}

Name	Period	$f.jitter$	$Size_f$	Bandwidth
f_1	125ms	NA	64	4Mbit/s
f_2	125ms	NA	512	32Mbit/s
f_3	250ms	NA	64	2Mbit/s
f_4	500ms	NA	1500	24Mbit/s
f_5	125ms	NA	128	8Mbit/s
f_6	125ms	NA	512	32Mbit/s
f_7	250ms	NA	512	16Mbit/s
f_8	125ms	NA	128	4Mbit/s
$f_9 - f_{13}$	125ms	$1\mu s$	64	4Mbit/s
f_{14}	125ms	$500\mu s$	256	16Mbit/s
f_{15}	125ms	$500\mu s$	512	32Mbit/s

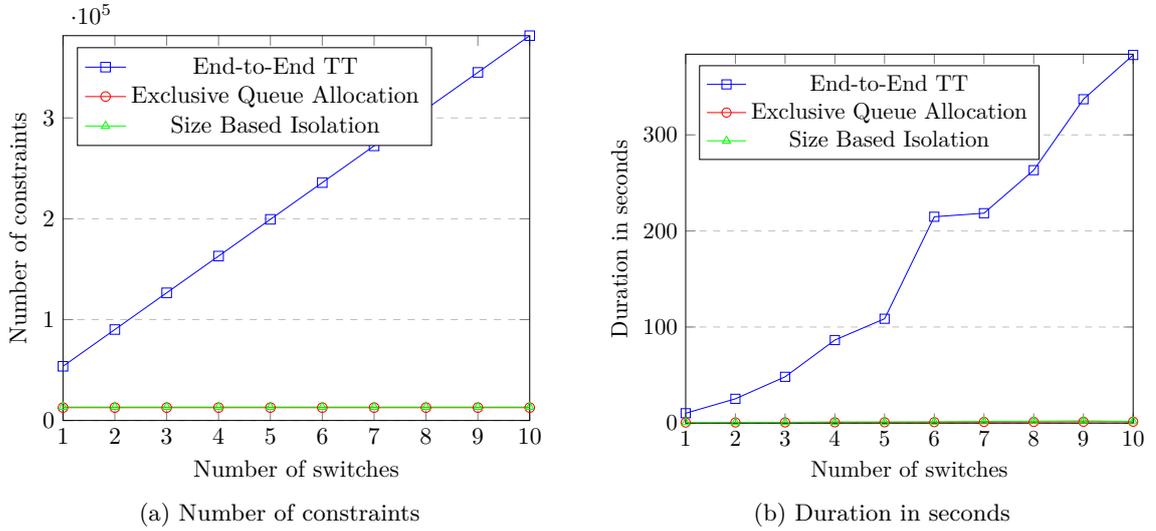


Figure 11.3: Path Size Increase: End-to-End TT vs. Egress TT

Fig. 11.3 presents the number of constraints and duration for the computation of one configuration. The number of constraints for both Egress TT implementations appears constant whereas it increases with End-to-End TT. This result was expected since, in End-to-End TT configurations, an emission instant has to be constructed for all frames in all hops of the network. In Egress TT the size or shape of the path of any flow is taken into account in *NetLatBound*. A change in the path of any flow in Egress TT will only change the value of *NetLatBound* and not add any additional constraint. Thus the resolution time for Egress TT remains almost constant and thus much faster than End-to-End TT. Between Exclusive Queue Allocation and Size Based Isolation, Fig. 11.4 shows that the number of constraints in Size Based Isolation is slightly greater than Exclusive Queue Allocation and therefore the computation time is also slightly greater, while remaining significantly faster than End-to-End TT configurations. This result was also expected: in message size isolation, in addition to gate events, the additional padding for all the flows in the network has to be computed.

Remark 39 *In the experiment with 6 switches, the measured duration for End-to-End TT does not*

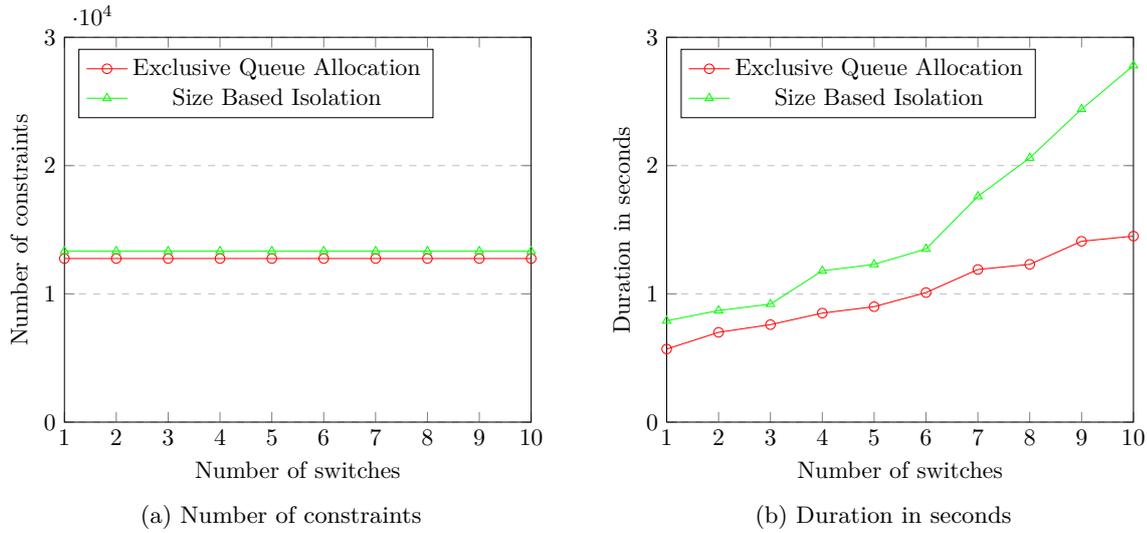


Figure 11.4: Path Size Increase: Exclusive Queue Allocation vs. Size Based Isolation

follow the trend of the other experiments (with different number of switches). We have no justification for this deviation and will investigate it in future works.

Number of receivers increase

In the second set of experiments, we consider the same topology and increase the number of receiving end-stations from 1 to 6 (cf. Fig. 11.5). This helps quantify the computation cost of adding an end-station. Each additional end-station will be receiving 15 flows with characteristics identical to those of Table 11.1, all emitted from "Sender".

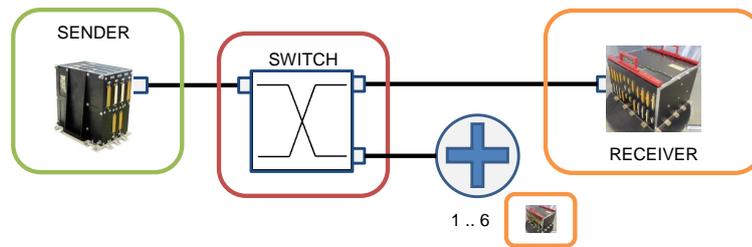


Figure 11.5: Number of receivers increase use case

Fig. 11.6 presents the number of constraints and duration for the computation of one configuration. Again, the computation of a configuration is much quicker with Egress TT than End-to-End TT. While a End-to-End TT configuration of a network with 6 receivers will take roughly 75 minutes with our implementations, the Egress TT configuration of the same network will only take about 9 seconds with Exclusive Queue Allocation and 18 seconds with Size Based Isolation. Increasing the

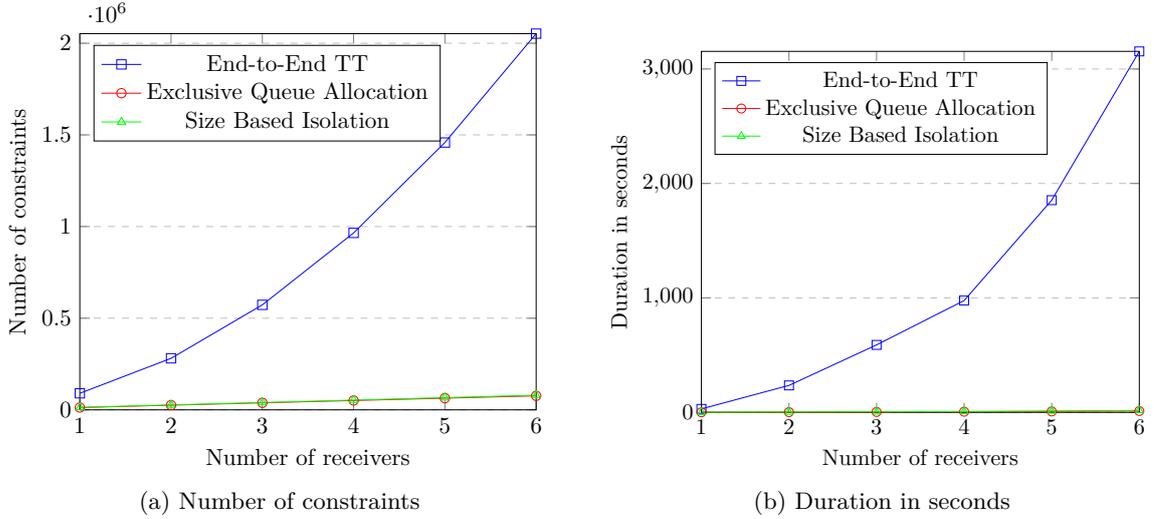


Figure 11.6: Number of Receivers Increase: End-to-End TT vs. Egress TT

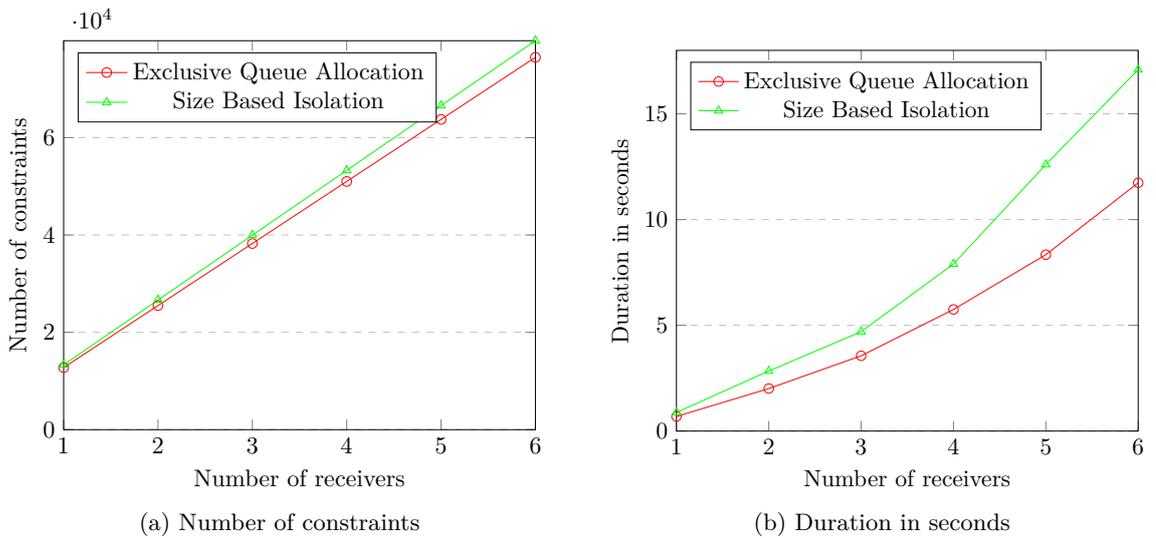


Figure 11.7: Number of Receivers Increase: Exclusive Queue Allocation vs. Size Based Isolation

number of receivers, hence the number of flows, increases the number of constraints per hop for the decision of the emission instants. In Egress TT configurations, only the emission instants of the last hop switch in the path of any flow have to be computed. The impact of flows on each other is taken into account in *NetLatBound* and additional constraints are only added on last hops. Therefore the total number of constraints is lower and the computation time is also shorter. Between Exclusive Queue Allocation and Size Based Isolation, Fig. 11.7 shows the same results than the previous experiment.

11.2.2 Network Latency

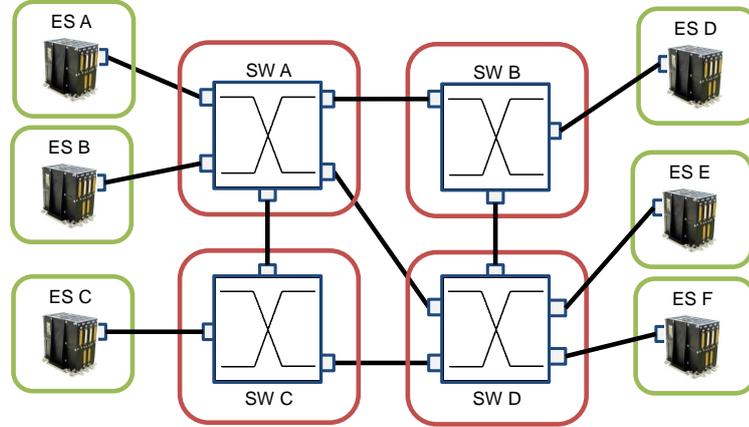


Figure 11.8: Topology for latency evaluation use case

The next experiment consists in running both End-to-End TT and Egress TT algorithms on the same use case (depicted in Fig. 11.8) while using the optimization criteria and compare the resulting network latency for both configurations. However, since our optimization criteria (brought by Development Effort Requirement 1) is not tackled by the literature, we replaced it by an optimization criteria aiming at minimizing network latency for End-to-End TT.

We consider a set of 4 data paths detailed in Table 11.2. On each data path, the source sends a set of 15 flows to the destination. Flows have the same characteristics as the ones of Table 11.1. On the following graphs, we will only depict the average latency from data path ① and ④.

Table 11.2: Set of flows \mathbb{F}

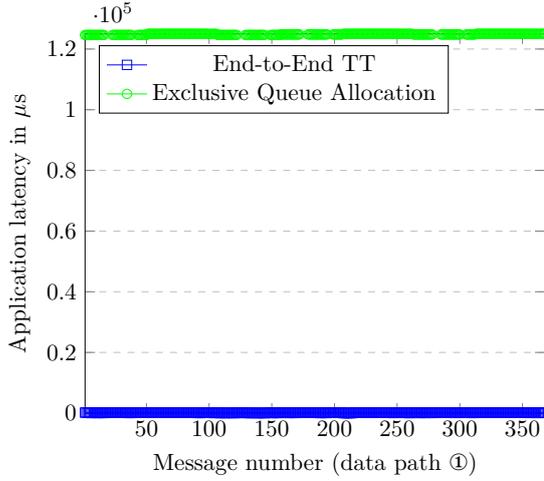
Number	Source	Path	Destination	Flows
①	ES_A	$SW_A-SW_B-SW_D$	ES_E	$f_1 \dots f_{15}$
②	ES_B	SW_A-SW_B	ES_D	$g_1 \dots g_{15}$
③	ES_C	$SW_C-SW_A-SW_B-SW_D$	ES_F	$h_1 \dots h_{15}$
④	ES_D	SW_B-SW_D	ES_E	$i_1 \dots i_{15}$

As a reminder, the latency per frame (see Remark. 16) is:

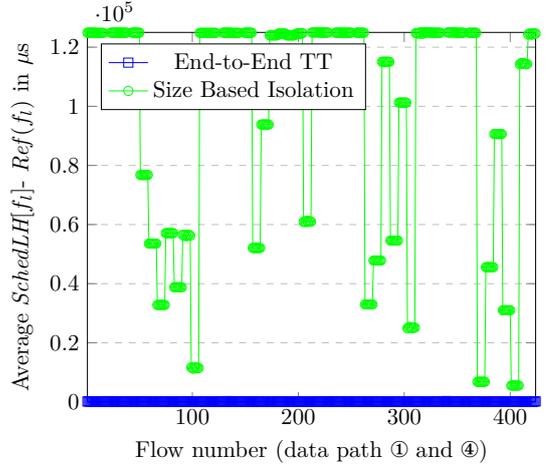
- in End-to-End TT: $Lat_{f_i} = T_r(f_i) - Ref(f_i)$
- in Egress TT: $Lat_{f_i} = SchedLH[f_i] + \tau_{f_i} - Ref(f_i)$

Due to the intrinsic limitation of Exclusive Queue Allocation (maximum of 7/8 jitter flows per last hop port), we have compared Exclusive Queue Allocation with state of the art End-to-End TT only using data path ①, ② and ③. Fig. 11.10a shows the results of this first experiment. It took 5 seconds for Egress TT and 30 minutes for End-to-End TT to generate a valid configuration.

Then, data path ④ is added and Size Based Isolation and state of the art End-to-End TT are then compared. The experiment lasted 12 seconds for Egress TT and 30 minutes for End-to-End TT. The results of this second experiment are shown in Fig. 11.9 and Fig. 11.10.

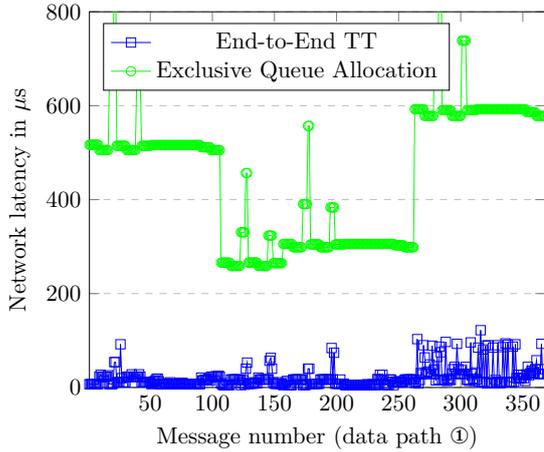


(a) End-to-End TT vs. Exclusive Queue Allocation

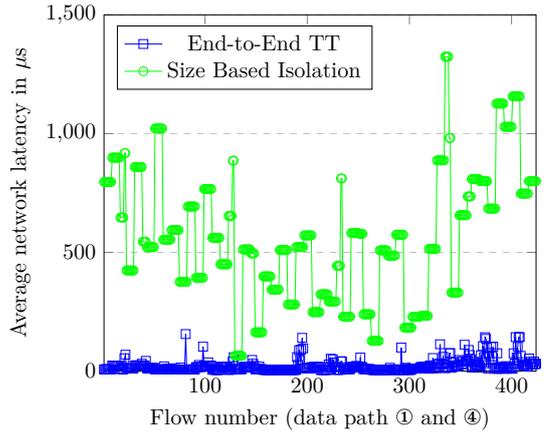


(b) End-to-End TT vs. Size Based Isolation

Figure 11.9: Applicative Latency - End-to-End TT vs. Egress TT



(a) End-to-End TT vs. Exclusive Queue Allocation



(b) End-to-End TT vs. Size Based Isolation

Figure 11.10: Network Latency - End-to-End TT vs. Egress TT

We have chosen to represent two concepts of latencies in these figures. On the one hand, Fig. 11.9 represents the concept of latencies described in our model (cf. Chapter 8). We call these latencies: *Application Latency*. On the other hand, Fig. 11.10 represents the classical concept of latencies in the networking community (hereafter called *Network Latencies*). Unless stated otherwise, we will use the word *latency* to deal with Application latency. As a reminder:

$$\begin{cases} \text{Our model:} & Lat_{f_i} = T_r(f_i) - Ref(f_i) \\ \text{State of the art:} & Lat_{f_i} = T_r(f_i) - T_e(f_i) \end{cases}$$

Experiments show that Egress TT configurations will lead to greater network latencies than End-

to-End TT configurations. This increase of network latency is due to the definition of Egress TT configurations: a message is delayed by a bound on its worst case latency so that it can always be delivered at the same time and meet its jitter requirements.

In addition, Egress TT also leads to greater application latencies. However, one element is worth mentioning when looking at the charts of Fig. 11.9. In fact, by design, Egress TT configurations will induce application latencies (i.e. the definition of our model) as large as possible whereas End-to-End TT will induce network latencies as small as possible. Therefore, when End-to-End TT and Egress TT latencies are close in the graph does not mean that Egress TT performs as good as End-to-End TT. Instead, it just means that for that particular flow, Egress TT did not perform as good as for other messages i.e. that the production contract of that message is tighter than others.

In summary, according to the experimental results above, Egress TT configurations reduce the computation effort (and computation time) compared to End-to-End TT configurations at the cost of a greater network latency while having, by design, a lower impact on applications.

It is important to insist on the meaning of these results: this chapter compares the scalability and latency of two approaches (End-to-End TT and Egress TT) that were not designed for the same purpose. While End-to-End TT aimed at satisfying jitter requirements while minimizing network latency, Egress TT configurations aimed at satisfying jitter requirements while minimizing application impact and reducing the computation effort, based on the assumption that minimizing latency is not always required in implicit deadlines systems and might over-constrain the system. Therefore, one solution is not better than the other. Rather, a network system designer will have the ability to choose, according to his needs, between one approach or the other.

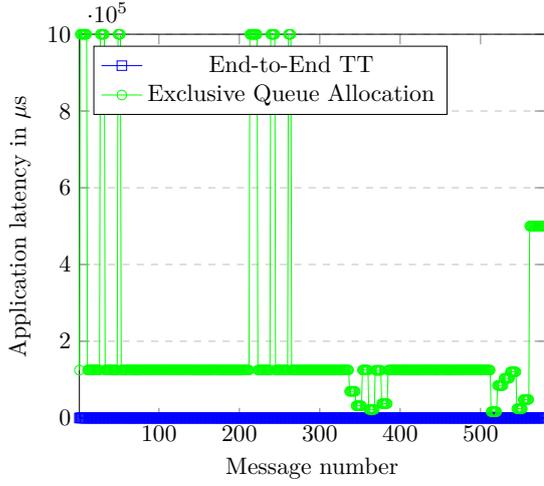
11.3 Experimentation on Larger Use Cases

11.3.1 Airbus Generic Next-Generation Satellite Use Case

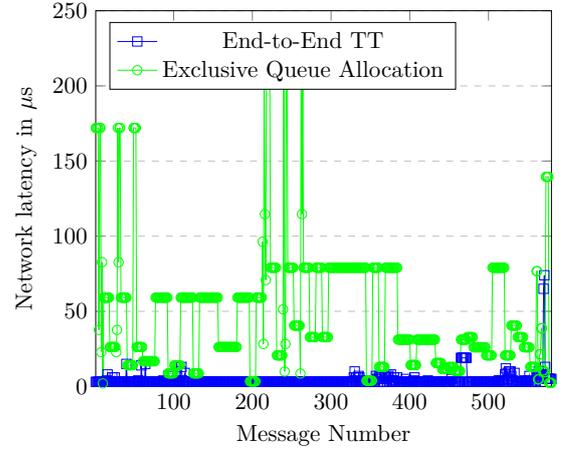
This first larger scale experiment relies on the Airbus Generic Next-Generation Satellite Use Case described in Section 9.4.1. This use case is composed of 120 flows: 18 jitter flows and 102 no jitter flows described in Appendix B.1. We generated configurations for this system with the two implementations from our second contribution i.e. Exclusive Queue Allocation and Size Based Isolation both using the optimization criteria aiming at maximizing the production contract (see Section 10.2.4).

Exclusive Queue Allocation

In Fig. 11.11 (resp. Fig. 11.12), we present the computed application (resp. network) latencies for a configuration of the Airbus Generic Satellite generated with Exclusive Queue Allocation and a configuration generated with our implementation of End-to-End TT. Because of the limitation of Egress TT configurations with Exclusive Queue Allocation (i.e. a last hop port can only receive up to 8 jitter flows without no jitter traffic or 7 jitter flows plus additional no jitter traffic), we only considered 13 jitter flows and 102 no jitter flows in this figure. It appears clearly in Fig 11.11a that Exclusive Queue Allocation generates configuration with greater latencies than End-to-End TT. In average, latencies from the Exclusive Queue Allocation configuration are 8500 times greater than latencies from the End-to-End TT configuration. Nevertheless, the generation with Exclusive Queue Allocation took roughly 9 seconds while the generation with our End-to-End TT implementation lasted about 10 minutes. In addition, we observed that application latencies, for most flows can take up to 99% of their period with Exclusive Queue Allocation. This is an expected result. In fact, let

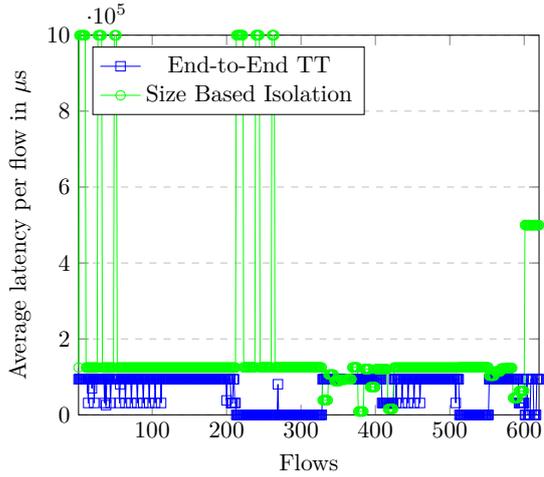


(a) Application Latency

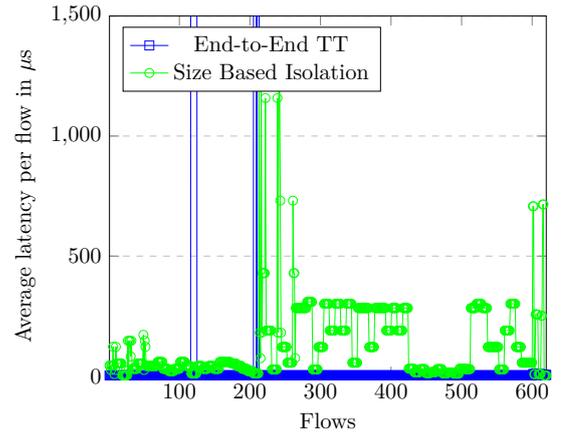


(b) Network Latency

Figure 11.11: Airbus Generic Satellite: End-to-End TT vs. Exclusive Queue Allocation



(a) Application Latency



(b) Network Latency

Figure 11.12: Airbus Generic Satellite: End-to-End TT vs. Size Based Isolation

us remind our optimization criteria:

$$\text{maximize} \quad \sum_{\forall f \in \mathbb{F} \text{ s.t. } f.\text{jitter} \neq NA} (\text{Release}(f_i) - \text{Ref}(f_i))^2$$

Let us write $\text{Release}(f_i)$ as a function of $\text{SchedLH}[f_i]$:

$$\text{maximize} \quad \sum_{\forall f \in \mathbb{F} \text{ s.t. } f.\text{jitter} \neq NA} (\text{SchedLH}[f_i] - \text{NetLatBound}(f_i) - \text{Ref}(f_i))^2$$

Now, we replace $SchedLH[f_i]$ as a function of Lat_{f_i} :

$$emphmaximize \quad \sum_{\forall f \in \mathbb{F} \text{ s.t. } f.jitter \neq NA} (Lat_{f_i} + \tau_{f_i} - NetLatBound(f_i))^2$$

Therefore, since τ_{f_i} is constant, the optimization criteria leads to maximizing latency. Observing great latencies means that we provided large production contracts and therefore a lesser development effort.

Remark 40 *We remind the reader that latency is not the common network latency but the latency of Remark. 16.*

Size Based Isolation

In Fig. 11.12a (resp. Fig. 11.12b), we present the computed application (resp. network) latencies for a configuration generated with Size Based Isolation and a configuration generated with our implementation of End-to-End TT. The configuration is realised on the full set of 120 flows. There are several elements that needs commenting in this second experiment. First of all, with our implementation of Size Based Isolation, we were able to configure the network for all the jitter flows of the generic satellite system. However, we were not able to configure the network for one no jitter flow. This flow is not represented in neither Fig. 11.12a nor Fig. 11.12a. Let us detail the reasons why we believe this flow could not be configured. In fact, this flow is a no jitter flow. Therefore, its release instant is computed a posteriori, once all the jitter flows have been computed (see Section 10.2.4). In order determine the release instant for the messages of that flow, per message, starting from the message's deadline, we go back in time until we find an *open* gate event large enough to fit that message. According to the algorithm provided in Section 10.2.4, a gate event is considered large enough if and only if it is larger than the largest frame transmission duration going through that port. Therefore, even if this message is really small, if some large no jitter frames go through the same port, a small gate event that could have been large enough for the small message is considered closed. Therefore, we keep going back in time until we find an open event large enough for the message. For the particular flow that we could not configure, our algorithm proposed, for several of its messages, a gate event before the reference date of these messages. This would imply that the message would have to be produced before the beginning of its period which is impossible. Nevertheless, this does not mean that the Size Based Isolation implementation is not correct, but instead that it can be improved. We think of two potential solutions: adding additional hypothesis (order between no jitter message for instance) that would help use the small gate events that we discarded; or rely existing in the state of the art such as incremental approaches [110] for the configuration of no jitter flows.

Besides, there is a strange result than we are not able to explain: computed latencies in Size Based Isolation are sometimes, for no apparent reasons, smaller that latencies from the End-to-End TT implementation. In addition, we are surprised by the difference in latencies between End-to-End TT on the reduced and the full set of flows: by adding only 5 low jitter flows, latencies as completely modified. One possible explanation might be that we did not execute the configuration generator long enough for our optimization criteria to do its job. We will investigate these reasons in future works.

11.3.2 ORION CEV Use Case

Finally, in one last experiment, we evaluate Egress TT on a use case adapted from the Orion Crew Exploration Vehicle (CEV) use case (described in Section 9.4.2). It is composed of 100 jitter

flows and 86 no jitter flows. Although the model described in Section 2 supports multicast flows, our implementations of End-to-End TT and Egress TT that we have used for this paper do not. Therefore, we have duplicated multicast flows into several unicast flows. Thus, the use case is composed of 168 unicast jitter flows and 147 unicast no jitter flows.

Remark 41 (*Multicast Support*) *There is an adverse effect in doing so: the number of messages in the network is unnecessarily increased, which might reduce the possible configurations. However, it is sufficient to demonstrate the concept of Egress TT configurations. Improving the implementation for multicast support is relatively simple and will be proposed in future works.*

Size Based Isolation offers the possibility to put in the same queue several flows with same source and same destination. However, in this use case, there is no two flows with same source and same destination. Therefore, we will only experiment with Exclusive Queue Allocation.

In addition, because of the limitation of Egress TT configurations with Exclusive Queue Allocation, we can only consider 157 unicast jitter flows and 147 unicast no jitter flows. We compare this reduced use case with End-to-End TT.

Unfortunately, our implementation of End-to-End TT (state of the art), maybe too naive, did not allow us to compute a configuration on the full set of flows of the Orion CEV use case like [125] did in their paper. Therefore, we were only able to obtain an End-to-End TT configuration (without optimization criteria) on a reduced set of 60 flows. The generation of the configuration lasted about 4 hours in End-to-End TT and 18s with Exclusive Queue Allocation. The Egress TT configuration generation for the full size use case was successful and lasted 4 minutes.

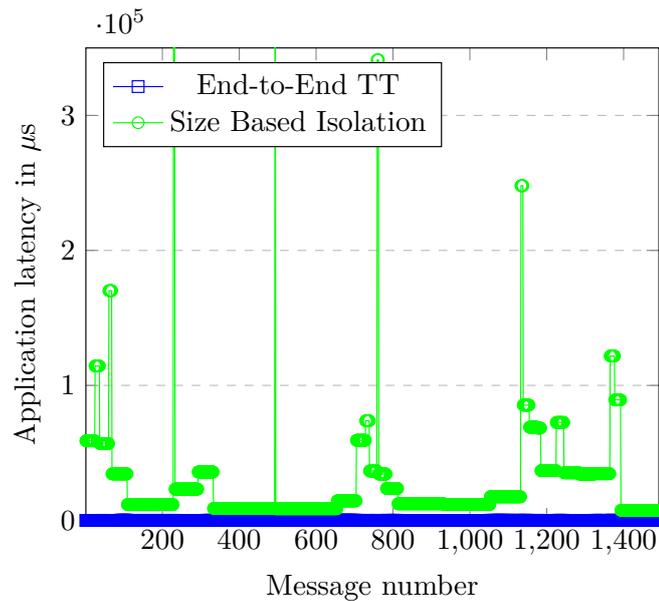


Figure 11.13: Orion CEV: End-to-End TT vs. Exclusive Queue Allocation (Application Latency)

This experiment confirms our previous observation on scalability and latency. On the reduced use case, observed network latencies are, in average, 30 times greater in Egress TT than in End-to-End TT configurations.

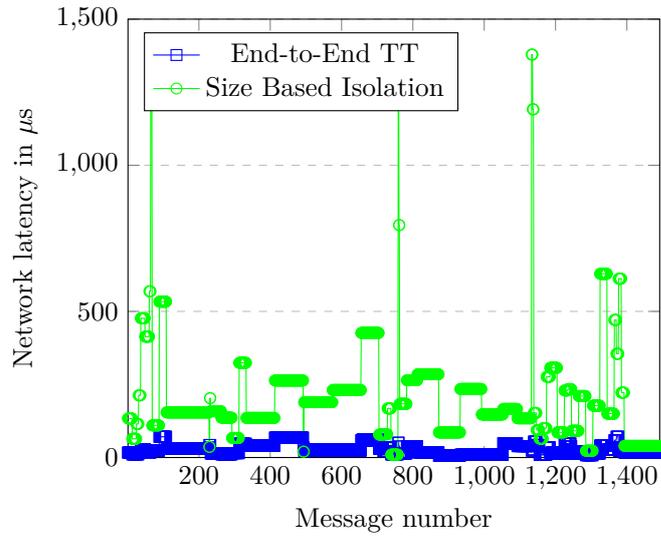


Figure 11.14: Orion CEV: End-to-End TT vs. Exclusive Queue Allocation (Network Latency)

Remark 42 *If network latency is an important requirement for the system, the optimization criteria could be modified to take it into account when generating a configuration. We did not do it in these experiments since the optimization criteria that we use was provided along the Airbus Generic Satellite use case.*

Part IV

Conclusions & Perspectives

Chapter 12

Conclusions & Perspectives

Conclusion

In light of the raging increase in performance demand in the satellite industry, the goal of this PhD was to discuss the suitability of an emerging set of standards, namely Time Sensitive Networking (TSN for short), with respect to the foreseen requirements of next generation satellite on-board networks. This study was included in the TANIA-DP roadmap (*Technological Assessment for New Instruments and Avionics - Data Processing*) at Airbus Defence and Space, which, among others, aims at providing new on-board communication standards for satellites *Platform & Payload* networks which would benefit future missions, be it on performance, on cost, on compatibility with ground segment standards or on system operation sides.

The TSN standards

Time Sensitive Networking is the state of the art IEEE Ethernet technology. Initiated after the previous activities on IEEE AVB (Audio Video Bridging), the IEEE TSN Working Group has introduced more than twenty standards aimed at providing real-time and high-throughput capabilities, fault tolerance capabilities, time-management capabilities, and configuration management capabilities to an Ethernet network. In 2017/2018, the popularity of Time Sensitive Networking began to expand with growing interest coming from three industrial sectors i.e. the factory automation industry, the automotive industry and the 5G industry. Soon, it aroused the interest of other industries including the aerospace industry. The set of standards is still growing today but first TSN components have already made it to the market, implementing only a subset of TSN standards. Components will be getting more mature in the years to come and products embedding TSN will definitely appear in the next five years. As the set of standards in TSN is getting bigger and bigger, industrial working groups have started to propose TSN *Profiles* defining a reduced set of standards and/or parameters for the use of TSN in a specific context (e.g. Automotive Profile [64]). In that sense, we were the first to promote, in the TSN\A conference [15] and in ERTS Congress [18], the interest of the spacecraft industry towards TSN for next-generation satellite networks as well as a glimpse on an aerospace profile. This aerospace profile is now being refined in a joint effort of IEEE and SAE ([65]).

Contributions

In order to discuss the suitability of Time Sensitive Networking for the spacecraft industry, we described in this manuscript the two step approach that we put in place during this 3 year PhD program. First, we assessed in a qualitative approach that TSN was indeed a suitable candidate for a next-generation on-board network. Second, we assessed, in a quantitative way, that TSN was capable of satisfying the performance quality of service requirements of future satellite missions. Let us summarize the process and results we obtained in both steps.

Selection of candidate technologies compliant with the quality of service requirements of a next-generation satellite network

In order to validate that TSN was a suitable candidate for a next-generation on-board network, we propose to realise a qualitative comparison of several technologies, including TSN, inspired from existing comparisons in the state of the art. The list of technologies (i.e. *Ethernet*, *ARINC 664*, *TTEthernet*, *Spacefibre* and *Time Sensitive Networking*) were pre-selected by our industrial partner at the beginning of this PhD. To effectively compare the technologies we defined a set of three properties (*Mixed Traffic Type*, *Time Management* and *Fault Tolerant Operations*) that described a high level view on foreseen quality of service requirements of future missions. We then refined these properties into criteria that served as a basis for the comparison. The output of the comparison was a set tables, available in Tab. 7.2, 7.3 and 7.4, which represent, per criteria and per technology, our evaluation of the compatibility of the technology with the corresponding criteria.

We identified, through this qualitative comparison, three suitable candidates for a next generation satellite network: TTEthernet, Spacefibre and TSN. Ethernet and ARINC 664 were discarded since they did not provide, among others, a very low reception jitter capability which was critical for current and future satellite missions. We then confronted the three remaining technologies on third party arguments i.e. not quality of service related so as to reduce even more the list of candidates. Nevertheless, this discussion did not reduce the list of candidates. On the one hand, while TTEthernet and Spacefibre have components that are designed to work in a space environment, the existing TSN components currently on the market do not have this kind of capability. Therefore, the industrial partner would have to make its own hardware and software so as to embedded TSN in Space. On the other hand, TSN is (or will be) a widespread technology, used in diverse industrial sectors with many use cases, therefore they will definitely be numerous COTS IP Cores (Components Of The Shelves Intellectual Property Cores) from different manufacturers that the industrial could easily instantiate in its own space-hardened hardware. This is actually already considered in the roadmap of several component providers which try to create synergies between their space and their automotive/industrial automation business units working on the same TSN topics. By relying on COTS components, the spacecraft industry would benefit from a TSN mass market to have attractive cost for the use of TSN. This might not be the case with TTEthernet (resp. Spacefibre) which is a mono-manufacturer proprietary (resp. space-specific) technology. The comparison and the analysis of Part II have been published in [19] and [20].

TTEthernet had already been studied in previous studies and Spacefibre will be studied in future activities, therefore, we proceeded with a more precise evaluation of the suitability of TSN in the second step of our approach.

Computation of TSN network configurations for next-generation satellite networks

In order to further assess the suitability of Time Sensitive Networking with next-generation satellite networks, we proposed in this second step to highlight its compatibility by computing valid network

configurations i.e. configurations that would satisfy the performance and fault tolerance quality of service requirements of next-generation on-board networks. This is not a simple task since TSN is composed of multiple standards, themselves composed of several mechanisms which are configured via numerous parameters. In order to simplify the problem that we had at hands, we proposed to reduce our scope to only one TSN standard i.e. IEEE 802.1Qbv-Enhancement for Scheduled Traffic and discuss its compatibility with performance quality of service requirements.

The configuration challenge of TSN was and still is a hot topic in the networking community and configuration strategies are already available in the state of the art. Nevertheless, the existing methodology in the state of the art for the configuration of 802.1Qbv had multiple drawbacks. First there were computationally expensive. In fact, these configurations relied on a frame schedule on all hops for all frames in the network that had to be computed for any configuration. As a consequence, the computation of a single configuration took time and resources and the effort drastically increased as the networks to configure got bigger and bigger. Then, these configurations induced a certain effort on application development. In fact, in these configurations, applications running over the network must emit their messages in somewhat tight emission windows that the application designer had to take into account during system design. Finally, these configurations required that all devices in the network embedded TSN features (i.e. TSN standard 802.1Qbv). This created a huge gap between legacy networks (relying mostly on MIL-STD-1553 and SpaceWire) and next-generation TSN networks as all network devices and links had to be replaced. As a consequence, state of the art configurations strategies received moderate interest from our industrial partner.

In order to reduce the limitation from existing configurations in the state of the art, we proposed Egress TT, a novel configuration approach where a frame scheduled is only computed in the very last device in the path of any flow. The travel time of a message across the network is bounded and any variability in this delay is compensated by a fixed emission date in the last hop port. In other words, latency is traded off for jitter i.e. Egress TT provides very low reception jitter but induces greater latency than existing state of the art configurations. While Egress TT might not be suitable for all types of systems, it was of great interest in the satellite environment where latency constraints are large. In fact, the Egress TT approach seemed to reduce the computation effort since the number of schedule needed to be computed was considerably lower than state of the art configurations. It also seemed to reduce the legacy-to-next-generation gap since only the very last device in the network requires frame scheduling capabilities. In this manuscript, we presented two versions of our approach for Ethernet/Time Sensitive Networking networks entitled Exclusive Queue Allocation and Size Based Isolation.

In order to implement our Egress TT approach, we first created a formal model of the network and of its foreseen requirements. Then, we reused an existing configuration computation method from the state of the art: define the configuration problem as a constraint programming problem that can be solved by Constraint Programming solver or by SMT/OMT solvers. We adapted the existing 802.1Qbv configuration model so that it took the specific satellite requirements that we had previously formalized.

The first implementation, Exclusive Queue Allocation, relied on an isolation constraint that we designed so that the existing message-loss independence property of satellite networks i.e. a loss a frame in a flow does not affect the nominal behaviour of other flows, remained satisfied. However, this came at a cost: an output port, in a last hop device (i.e. a device at the last hop in the path of a flow), could no receive more that 8 different flows (i.e. space isolation of different flows). While this could have been good enough for small networks, bigger satellite use cases could not be computed with this method. Therefore, in an attempt at increasing the schedulability of Exclusive Queue Allocation while keeping the good expected properties of Egress TT, we implemented Size Based Isolation. This second implementation relied on a modified version of the previous isolation

constraints where messages from different flows are not separated in space anymore (i.e. they share the same resources) but instead, are isolated with a clever use of message sizes.

We evaluated, in the last chapter, the performance of our two versions of Egress TT on several space use cases and compared them to our own implementation of state of the art configuration strategy and confirmed the reduction on computation effort and application impact.

Perspectives on the Egress TT approach

There are several perspectives that we envision as future work on the Egress TT approach.

New features in the model

New features could be added to the model without any considerable effort and therefore be part of the constraint programming model.

First of all, multicast could be integrated so as to support a wider variety of systems. As explained in Chapter 11, we had to adapt the Orion use case to our model since it had multicast flows and therefore the configurations that we ended-up with were not applicable to the original use case.

Another improvement would be to allow greater messages sizes at application level. This means that one applicative message would be divided into several Ethernet frames. We discuss the modifications of the model for this specific improvement in Appendix C.1.

In addition, one simple modification of the model would be to include frame preemption for no jitter traffic. This would lead to a modification of the delay computation formula that we provided. TSN frame preemption standards are detailed in Appendix A.1.

Talking about delay computation, the delay computation formula that we proposed in this manuscript is somewhat naive and could be improved. Nevertheless it was sufficient for the considered use cases. One way of improving the delay computation would be to couple the constraint programming solver with a network calculus tool that could compute better bounds on the delay at each iteration of the solver. This being said, we raise the reader awareness on the fact that this is currently not possible with the constraint programming solver that we used and might not be time efficient at all.

Finally, one could propose to integrate the Credit Based Shaper for no jitter traffic or even introduce a third traffic type in the model which medium access is handled via CBS. To do so, several elements would have to be refined. First, new decisions variables corresponding to the *idle_slope* and *send_slope* parameters of the used CBS would have to be introduced so that the CBS configuration can be computed. Then, the delay computation formula would have to be modified to take into account the CBS queues. Finally, the *exclusive gating* philosophy currently applied in Egress TT configurations would have to be challenged to decide whether CBS queues are open when no jitter (static priority) queues are open or not. In any case, this will definitely affect the computation of Release instants for no jitter flows.

New standards

Apart from new features related to the 802.1Qbv standard (and frame preemption), other standards could be included in the model for better support of real life systems. First, TSN standard IEEE 802.1CB Frame Replication and Elimination for Reliability could be added to our model. This would lead to a traffic increase in the network and therefore might have an impact on the delay analysis. If for instance only a subset of flows benefit from redundancy or if redundancy is only applied in a

subset of the network, the delay of flows sharing the same resources than those that are duplicated will be affected.

Another standard that could be integrated into the model is 802.1AS-2020. This standard is dedicated to network synchronization and based on a specific PTP profile. Therefore, there are messages exchanged in the network, that share the same resources as user data. These messages shall be taken into account in the configuration and resource reservation (for instance flow to queue mapping) since they will have any impact on the delay. In addition, depending on the expected synchronization quality, the number of message might not be the same and therefore the configurations might evolve depending on the number of synchronization messages.

Future works on Industrial Use of TSN in Space

Beside the new features and new standards additions to the Egress TT approach, there are still challenges remaining on industrial side towards the selection of the next-generation satellite on-board network technology. This not only deals with our assessment of TSN but also with the assessment of the other technology candidates (e.g. TTEthernet and Spacefibre) that were identified throughout the qualitative comparison. In fact, in the second contribution of this PhD, we only addressed performance quality of service requirements. Therefore, in order to further assess the suitability of TSN (and also of the other technology candidates), it is necessary to work on the definition of a clear FDIR model for these next-generation networks. For instance, will the switches be used in a hot redundancy or cold redundancy scheme? Will the links be duplicated, or triplicated? will redundancy be applied to the whole network? What are the Reliability and Availability values that are expected for future missions ? How can it be evaluated on a TSN/TTE/Spacefibre network ?

Another aspect that has not been discussed at all during this PhD is the physical medium aspects as we focused on ISO L2 features only. Therefore, we did not analyse the availability of physical mediums capable of handling the high-data rate traffic of next-generation satellites nor their compatibility with the space environment. Shall it be a fibre optic cable, a twisted-pair cable, a custom cable ? Shall we try to adapt a Spacefibre physical layer for a TSN MAC layer device ? The preliminary activities in which the industry is currently involved tend to suggest that the medium for the next-generation network would be from the fibre optic family but more studies are needed to fully design this physical layer.

Part V
Appendices

Appendix A

Introduction to Time Sensitive Networking

A.1 802.1Qbu & 802.3br -*Frame Preemption*

In order to reduce the interaction between frames of different traffic classes, the Time Sensitive Networking Task Group has introduced one feature dedicated to preemption namely *IEEE 802.1Qbu - Frame Preemption*. It is associated with IEEE 802.3br - *Interspersing express traffic* for lower ISO layer support. In fact 802.1Qbu and 802.3br allow purposely tagged frames (*express*) to suspend the transmission of other frames (*preemptable*) for their own transmission on a point-to-point link.

The part of the TSN output port in switches or end-systems affected by this protocol is highlighted in Fig.A.1

A.1.1 Introduction

Definition 78 (Express/Preemptable Tag) *IEEE 802.1Qbu offers the possibility to tag each packet, according to its priority, per port, as express or preemptable via a configuration table. This hence define two types of traffic:*

- **Express:** *This traffic, tagged express, is considered to have a higher priority than preemptable traffic. As a consequence, it can preempt any preemptable traffic.*
- **Preemptable:** *This traffic, tagged preemptable, is considered less important (i.e. lower priority) than express traffic. As a consequence, it can be preempted by any express traffic and cannot preempt any traffic.*

A.1.2 Frame format for Frame Preemption Support

802.3br defines the frames format in order to support frame preemption. It uses the Start Frame Delimiter (SFD) of the Ethernet frame to indicate whether a frame is *express (SMD-E)* or *preemptable (SMD-Sx, SMD-Cx)* as shown in Fig. A.2. The figure shows a standard Ethernet frame and how it is modified to become an express or a non-fragmented preemptable frame. The number at the right of the frame formats corresponds to the length, in bytes, of the associated field. The SMD hexadecimal values are not important for understanding this TSN feature. However, they can be found in Table

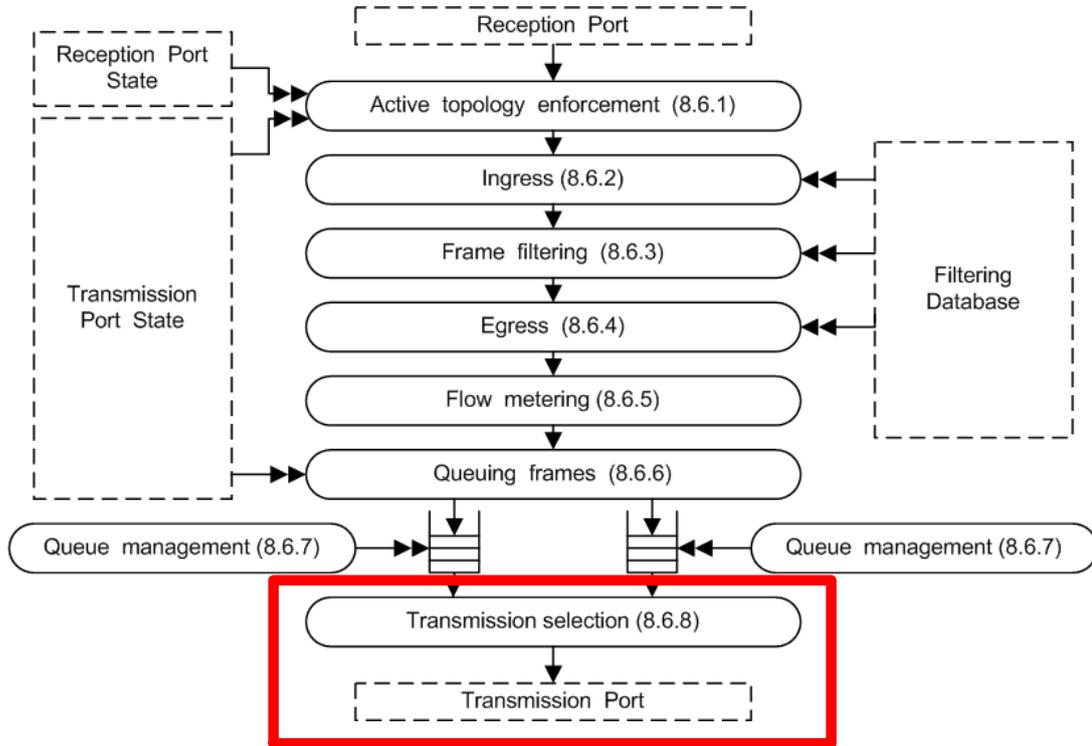


Figure A.1: 802.1Qbu & 802.3br perimeter in a 802.1Q switch

99-1, page 41 of *IEEE Std 802.3br-2016-Amendment 5: Specification and Management Parameters for Interspersing Express Traffic* (cf. [61]).

A.1.3 Frame Preemption How To

In fact, frame preemption occurs when an express frame becomes *available for transmission* while a preemptable frame is being emitted. The emission of the preemptable frame shall be stopped so that the express frame can be emitted. In order to be able to do so, at least $64 * (1 + addFragSize) - 4$ bytes of the preemptable frame must have already been sent. The variable *addFragSize* allows the user to configure the minimum fragment size. Once this number of bytes has been reached, a FCS (short for Frame Check Sequence), corresponding to the 4 least significant bytes of the CRC (Cyclic Redundancy Check) of the already transmitted bytes is computed and appended to the frame and sent on the medium. It is then called *mCRC* and this first frame is called *First Fragment*. The rest of the preempted frame is on hold until the express frame has been emitted.

When the medium is free, i.e. all express traffic that was available for transmission has been sent, the emission of the preemptable frame can start again. There cannot be any transmission of preemptable traffic except from the frame that was preempted because this would mean that a preemptable frame is preempting another preemptable frame, which is, per definition, impossible. When the emission of the frame starts again, the MAC Ethernet headers are not repeated, instead,

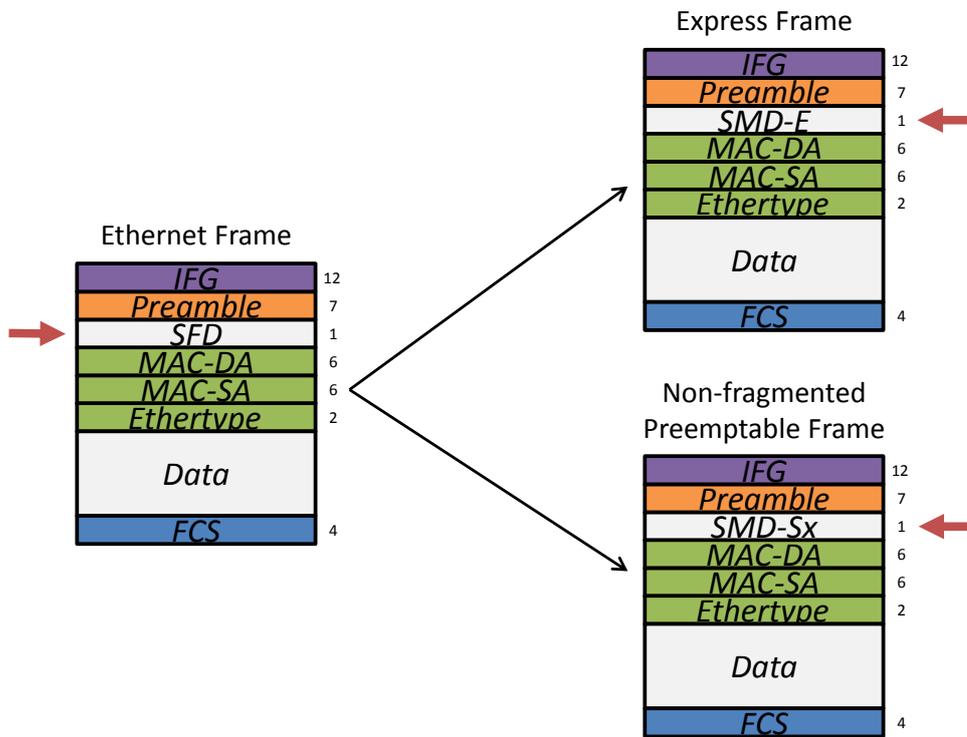


Figure A.2: SFD for preemptable/express traffic identification

a short frame’s header is used. It bears, within the Start Frame Delimiter, the information that this new frame is either an *Intermediate Fragment* or a *Last Fragment* as well as a modulo-4 counter that counts the number of fragments.

If the transmission of the frame is ended (i.e. the First Fragment was sent and the rest of the transmission happened in one frame), the second fragment is called Last Fragment. However, if during the transmission of the fragment, an express frame becomes available for transmission, this new fragment, called Intermediate Fragment, is preempted (under the same conditions that the one listed above), an mCRC is generated and the process starts again (i.e. waiting for the medium to be free and trying to send the rest of the frame).

For every Intermediate Fragment, the FCS used is the mCRC i.e. as a reminder, the four least significant bytes of the CRC of the current fragment. For the Last Fragment, the FCS used is the 32bit-CRC of the whole preemptable frame. This will allow to detect that the fragment received is the Last Fragment (i.e. $FCS \neq mCRC$), as well as to do integrity checking on the entire preemptable frame.

The format of the First Fragment, Intermediate Fragment(s) and Last Fragment are summarized in Fig. A.3.

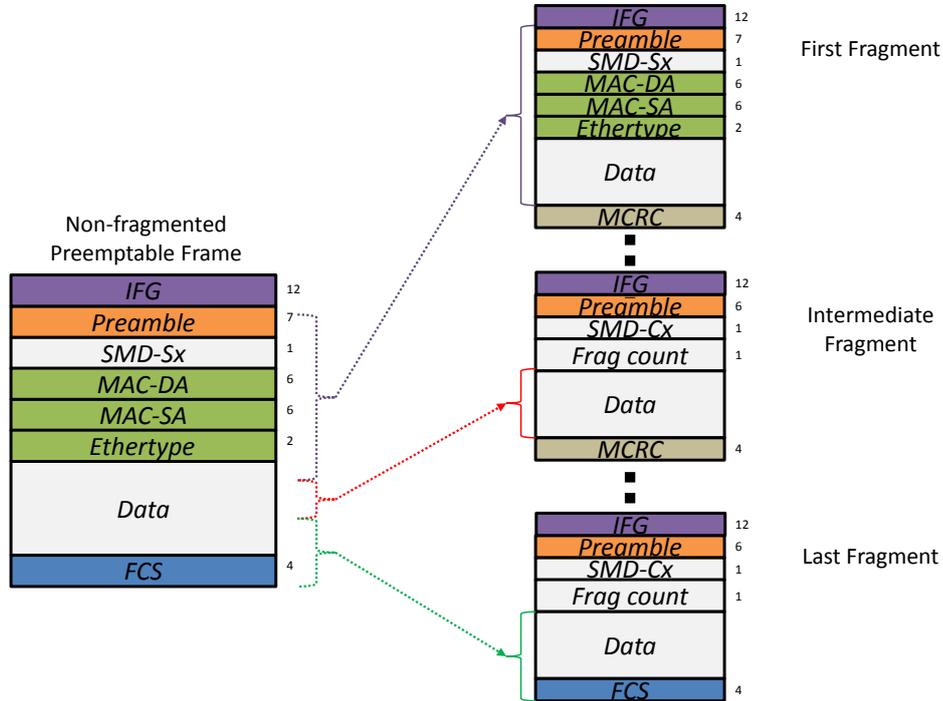


Figure A.3: Preemptible frame fragments formats

Thanks to Frame Preemption, the express traffic has not been affected by preemptible traffic.

A.1.4 Preemption Limitation

Frame preemption is not allowed to add padding to fragments. As a consequence, not all frames can be preempted! When a frame's length is inferior to 143 bytes, then it is considered not preemptible. In fact if the preemptible frame was 143 bytes long (including IFG, Preamble and SFD), then splitting it in two would violate the Ethernet frame minimum length of 84 bytes (including IFG, Preamble and SFD). This minimum length for preemptible frames is demonstrated in [118].

A.1.5 Configuring Frame Preemption

TSN feature IEEE 802.1Qbr (& IEEE 802.3br) Frame Preemption has only one mechanism: itself. There are only two possible configurations: *configured* or *not configured*. When configured, one parameter, namely *addFragSize*, allows to tune the size of the fragments. When Frame Preemption is configured, it is also assumed that traffic classes have to be classified as *express* or *preemptible* before-hand.

A.1.6 Behaviour of Frame Preemption when Combined with Transmission Gates in Exclusive Gating

Combining TSN features Frame Preemption and Enhancement for Scheduled Traffic (cf. Chapter 4.2) configured in exclusive gating may have a certain interest when trying to maximize the bandwidth of preemptable traffic. In fact, let us imagine a TSN output port TDMA configuration (see 4.2.5) with $M = 8$ and #7 in exclusive gating with the other queues. All frames from #7 will be tagged *express* and frames from the other queues are tagged *preemptable*.

As #7 and the other queues are in exclusive gating, this means that when the gate of #7 is open, the gates of all the other queues are closed and respectively when #7 is closed, the gates of all the other queues are open. In this situation, frames for traffic classes than #7 cannot interfere with #7 frames.

However, when the gate of #7 is about to open and all the other gates are about to close, depending on the configuration of the Frame Preemption feature the behaviour of the port changes. If Frame Preemption is not configured, the behaviour will be qualified of Preventive and if Frame Preemption is configured, the behaviour will be qualified of Preemptive. In the preventive scenario,

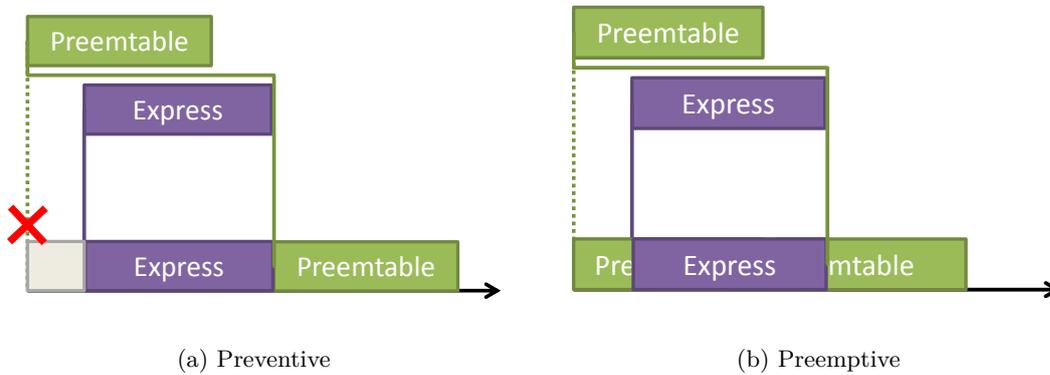


Figure A.4: Integration methods for Time Aware Shaper

as well as in the preemptive method, the schedule of closing and opening sequences of all traffic classes of the port is known. As a consequence, it is possible for the port to know the remaining time before the opening of the gate bearing express traffic (#7 in our example), hence if a preemptable frame is available for transmission, before being emitted, the port will check that there is enough time to transmit the whole frame. If it is the case, the frame is transmitted, if not, the frame is on hold, it waits for all express traffic available for transmission when their gate opens to be sent and then the preemptable frame is sent. This prevents any interference between preemptable and express traffic, i.e. between #7 and all other traffic classes however, the time during which the preemptable frame is on hold, because it cannot be emitted, is unused i.e. loss of bandwidth.

In the preemptive scenario i.e. when Frame Preemption is configured, the preemptable frame currently being emitted is preempted (if possible), when an express frame becomes available for transmission. If the preemptable frame was non-preemptable, it will be on hold until all express traffic available for transmission is sent. Configuring Frame Preemption seems interesting, in terms of bandwidth available for preemptable traffic because, instead of having the whole preemptable frame on hold, some of it can already been sent. In an exclusive gating configuration, using Frame

Preemption allows to increase the bandwidth of preemptable traffic at the cost of a small overhead of 12 bytes per-fragment (excluding Inter-Frame Gap), which is a rather small overhead.

A.1.7 Summary

TSN protocol Frame Preemption proposes a way to reduce the interference between express and preemptable traffic while maximizing the useful bandwidth for the preemptable traffic when combined with 802.1Qbv. Fig. A.5 summarizes the different configuration of TSN feature 802.1Qbu - Frame Preemption.

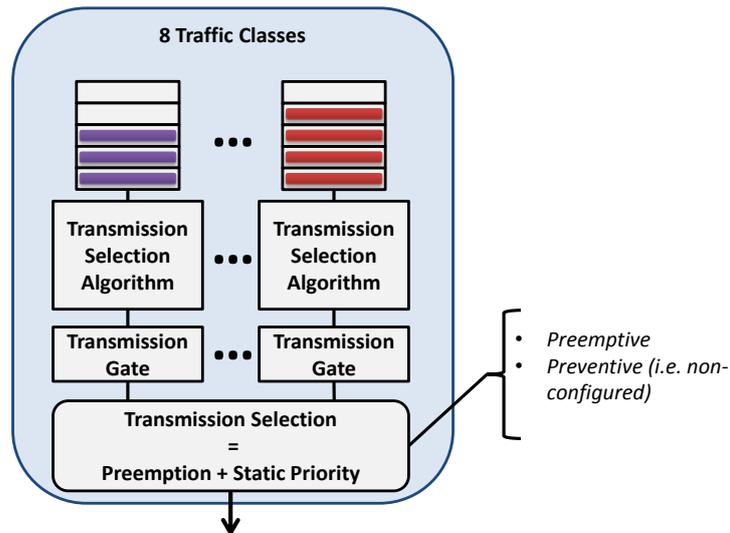


Figure A.5: Summary of configurable mechanisms in 802.1Qbu (and 802.3br)

A.2 802.1Qci-Per Stream Policing and Filtering

A.2.1 Introduction

Using TSN protocol 802.1Qbv, the bandwidth is share between traffic classed. In order to ensure that this share is respected and in order to add security features to the switch, the TSN Task Group introduced IEEE 802.1Qci - *Per Stream Policing and Filtering*. The overall goal of this TSN protocol is to protect the downfall switches and network devices both device failures and security attacks through impersonation or deny of service. It works by filtering and policing incoming traffic in the ingress port of a switch.

A.2.2 Concept of Streams

With Qci, a new concept must be introduced: streams. This stream concept is a new level of abstraction above frames. All the policing or filtering functions introduced with 802.1Qci will be applied to streams and not to frames.

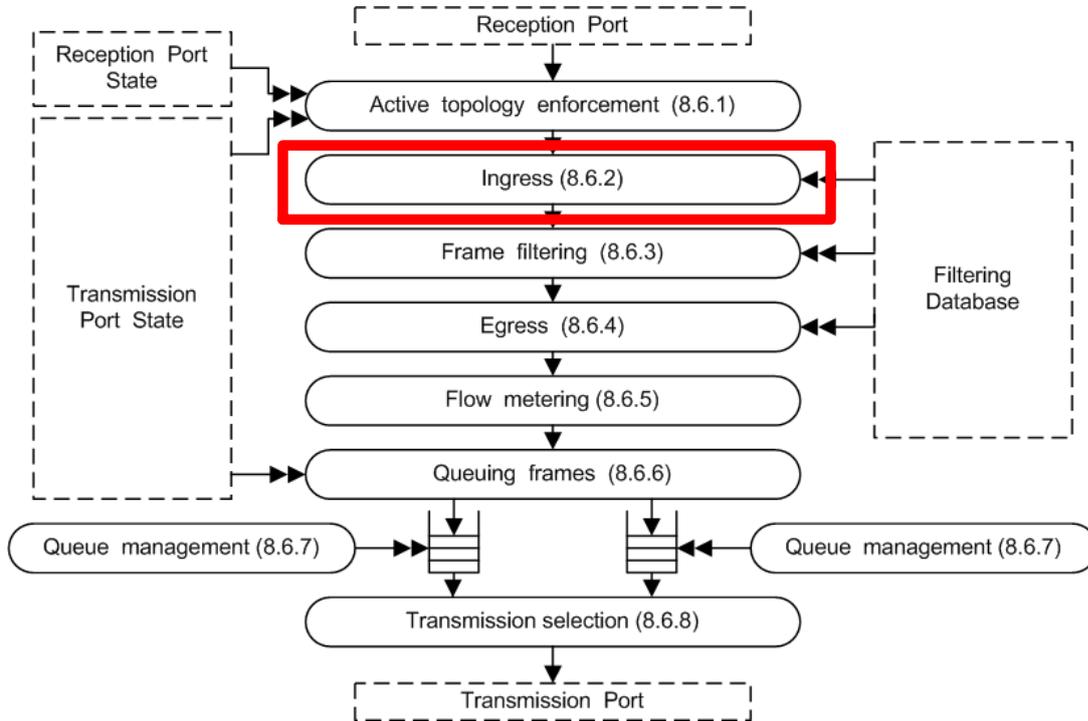


Figure A.6: 802.1Qci perimeter in switches

A stream is a multicast unidirectional flow of data going from a Talker End-Station to one or several Listener End-Stations. A stream is composed of frames going from an emitting application to one or several receiving applications. A stream, at system level, has an identifier, that allows to differentiate one stream from another. However this identifier is not a field of the Ethernet MAC header and is not written anywhere else in the frame. The identifier only exists within network devices (switches and End-Systems) and is called *stream_handle*.

As this *stream_handle* is not part of the frame, a method must be created and added to all the network devices in order to be able to distinguish between frames from different streams. This function, in TSN, is called the *Stream Identification Function*. We will describe this function in the next subsection.

A.2.3 Stream Identification Function

The *Stream Identification Function* is the function that allows network elements to recognise to which stream a frame belongs. As the *stream_handle* is actually part of the frame, a stream is identified through a combination of actual fields of the frame that is being analysed. The table of Fig. A.7 lists all available combination of fields that could be used to identify a stream. As this list of combination is not so big, an amendment is being written right now in order to offer an extension of the stream identification function (cf. IEEE 802.1CBdb). For the record, the *Stream Identification*

Stream Identification Function	Active/passive	Examines	Overwrites
Null Stream Identification	Passive	MAC <i>destination_address</i> + <i>vlan_id</i>	None
Source MAC and VLAN Stream Identification	Passive	MAC <i>source_address</i> + <i>vlan_id</i>	None
Active Destination MAC and VLAN Stream Identification	Active	MAC <i>destination_address</i> + <i>vlan_id</i>	<i>destination_address</i> + <i>vlan_id</i> + priority (PCP)
IP Stream Identification	Passive	MAC <i>destination_address</i> + <i>vlan_id</i> + IP source and destination addresses + DSCP + IP next protocol + <i>source_port</i> + <i>destination_port</i>	None

Figure A.7: Stream Identification Functions Summary

Function is not part of 802.1Qci. It was introduced in 802.1CB-*Frame Replication and Elimination for Reliability* (see Chapter A.3), but for the purpose of understanding 802.1Qci, we believe it was necessary to introduce it here.

Basically, stream identification functions analyses part of the MAC header (destination/source MAC addresses and VLAN identifier) and sometimes higher level elements (IP and above). One of the function is qualified active because it can, when the *stream_handle* is computed, change the value of the MAC destination address, VLAN identifier and priority of the frame.

A.2.4 QCi ingress port organization

With 802.1Qci, 4 mechanisms have been added to the switch ingress port:

1. *Stream Identification Function*,
2. *Stream Filters*,
3. *Stream Gates*,
4. *Stream Meters*.

Let us describe mechanisms 2, 3 and 4. 1 has already been introduced in Section A.2.3.

Stream Filters

Once the *stream_handle* has been recovered thanks to the Stream Identification Function in the ingress port of a switch, streams are passed to stream filters according to their *stream_handle*. In fact there are several stream filters available in the same ingress port. One stream filter instance can be associated with one stream or one subset of the stream in the case the *stream_handle* is combined with *frame_priority* for stream filter instance attribution. We envision that there will be

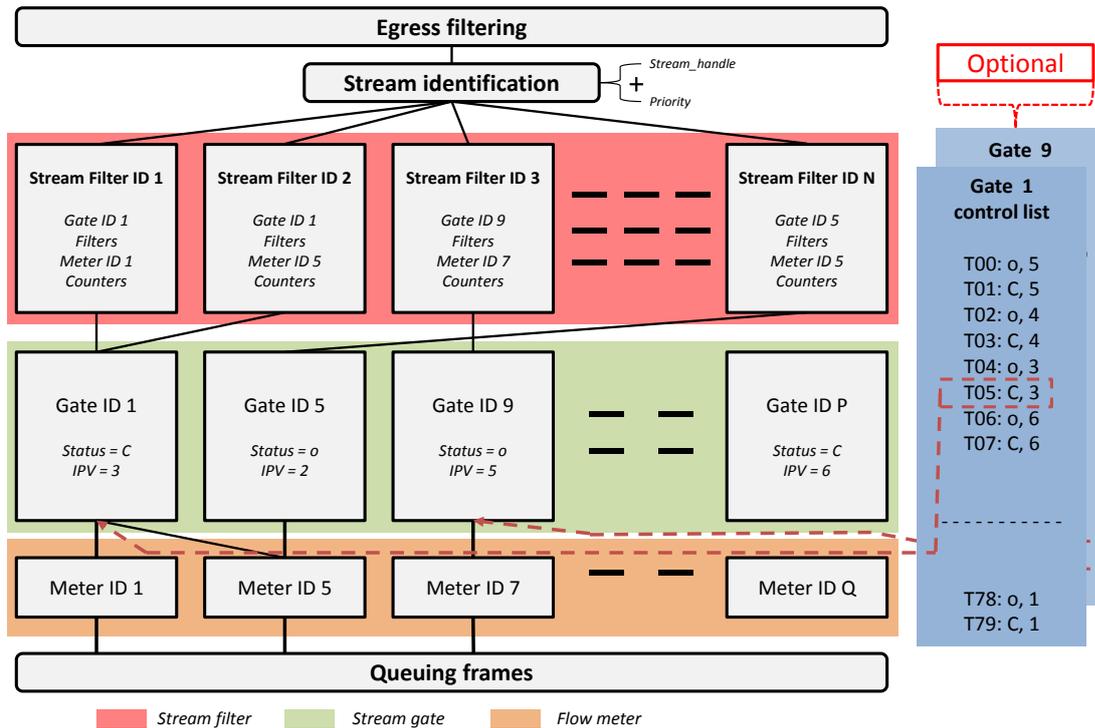


Figure A.8: 802.1Qci switch ingress port organization

more streams than stream filters due to the memory size limitation in switches, that is why, there is generally one of the stream filters that has a wild-card, signifying that all streams that were not specifically associated with a stream filter will be handled with a "default" filter. All the frames of a stream that pass through the same stream filter will go through the same filters, the same stream gate and the same stream meters.

A stream filter has several parameters:

- *Stream_filter_id*, that identifies the stream filter,
- *Stream_handle*, identifies the associated stream,
- *Priority*, for further associated stream identification (optional),
- Zero or more Filter specifications, that defines a set of filters (ex: Max MDU size, Flow meter identifier, etc.). If the frame of a stream does not match the filter conditions, it may be discarded,
- Frame counters, (total number of frames, number of dropped frames due to SDU filter, number of dropped frames due to Flow metering, etc.) for MIBs counters and statistics.
- a *StreamBlockedDueToOversizeFrameEnable* function that can be enabled or not. If enabled, the parameter *StreamBlockedDueToOversizeFrame* can be true or false. If it gets true, this means that a stream that went through this stream filter had an oversize frame. In that case, all frames of the stream passing through this stream filter are blocked and dropped. This function is optional.

- Stream_gate_id, used to identify to which stream gate the streams will be passed,
- Stream_meter_id, used to identify to which stream meter the streams will be passed.

With this first mechanism, TSN protocol 802.1Qci offers a way to filter streams, and discard their frames if they do not match any previously specified format requirements. In particular, the size of the frame. This looks very appealing, for security purposes as well as safety purposes. In particular thanks to the ability to block and drop certain streams that would not match some previously specified system/network requirements is very interesting compared to standard Ethernet where this type of filtering would be left to Quality of Service Mechanisms at higher ISO level.

Stream Gates

Among the parameters of the stream filter, one parameter identifies the stream gate instance to which the stream filter sends its streams. Contrarily to the one to one (stream;stream_filter_id) couple, a stream gate can be associated with several stream filters. This means that in a single stream gate, frames from different streams may coexist (in space or time). Again, a stream gate has several parameters:

- Stream_Gate_instance_ID, identifies the stream gate instance,
- Gate status, can be "Open" or "Closed",
- Internal Priority Value, is used to specify the 802.1Qbv traffic class to streams.
- GateClosedDueToInvalidRx function and parameter, used to detect and drop frames of streams that would arrive at a time instant when it was not supposed to arrived. This will come handy in time triggered situation in order to ensure that frames only arrive in the correct time windows.
- GateClosedDueToOctetExceeded function and parameter, used to check that during a time window, a certain amount of bandwidth has not been overpassed. Again this will come handy in order to ensure a correct share of the bandwidth between streams.
- Stream Gate Control List, same definition than the 802.1Qbv Gate Control List, but instead of controlling the behaviour of 802.1Qbv gates, it controls the 802.1Qci stream gates state.

The Internal Priority Value, or IPV, is rather interesting because this field will define in which 802.1Qbv traffic class (i.e. queue) a stream will go. This means that the association between streams and traffic classes is not constant, it may evolve during the journey of the stream's frame in the network. This opportunity of having the traffic class changed during the journey of the stream's frame is exploited by another TSN addendum (only an informative addendum and not a normative standard) referenced IEEE 802.1Qch - *Cyclic Queueing and Forwarding*, that will not be detailed in this document.

The Stream Gate Control List (SGCL), is, to us, a good way to protect the network from a schedule failure of one of the device. If the SGCL is identical to the 802.1Qbv GCL modulo a transmission time, then the switch could be aware of the schedule of the end-station or switch used before. If the schedule is known, then time windows, where traffic of such or such stream is not supposed to arrive, can be define. Using these time windows, it would be possible first to detect that there are schedule problems and notify the user but also to protect these streams' frames from potentially messing up with the schedule later on in the network.

Again, the policing functions provided by the streams gates is interesting for security and safety purposes. However, one thing there is more tricky: as several stream may be associated to the same gate, if the gate is closed because one of the streams did not respect its bandwidth constraints or its behaviour over time (schedule), the "bad" stream will be blocked and its frame will be dropped but the other streams using this gate will be also blocked and suffer the same punishment. This will

have to be taken into account when designing the system, especially on Time and Space partitioning aspects.

Stream Meters

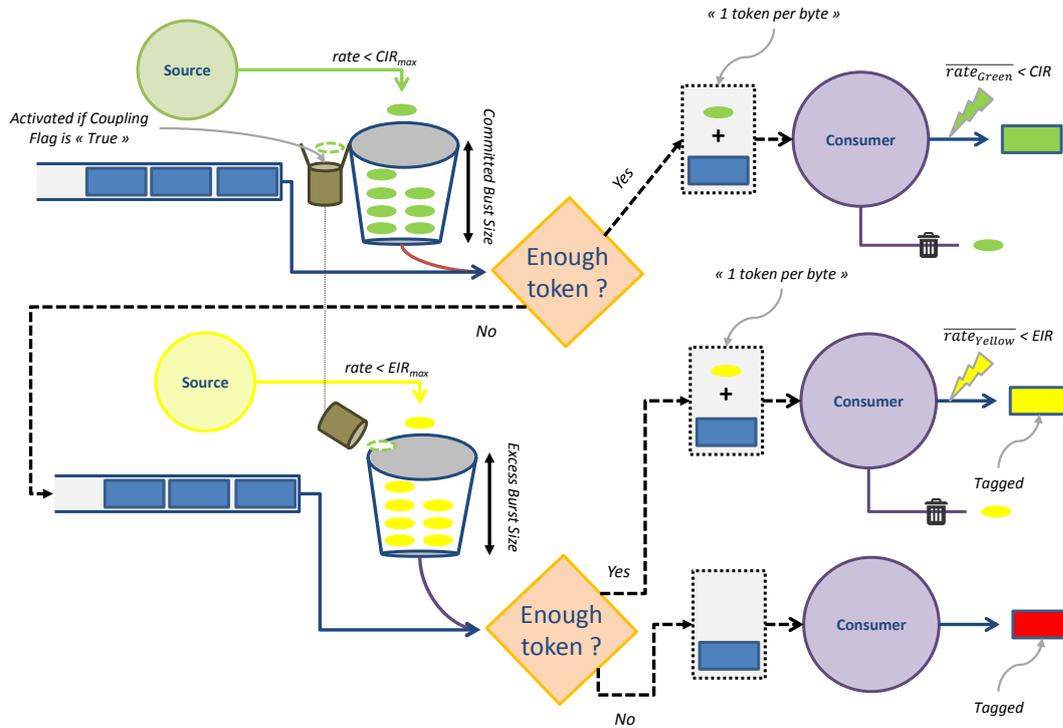


Figure A.9: Token bucket in 802.1Qci ingress switch ports

Last but not least, the last mechanism provided by 802.1Qci is the stream meter mechanism. It is the equivalent of a token bucket that does some filtering and policing decision to ensure that the streams respects their bandwidth specifications. Fig. A.9 illustrate how the Stream Meter mechanism works.

In fact, it really resemble to two "linked together" token buckets. The first token bucket is used to check that the regular bandwidth reserved or attributed for a stream or a group of stream is respected. If the stream(s) overtake this bandwidth limit, then there frames go to a second token bucket, that can be filled with token in order to admit some burst. In that case, the frames are marked yellow and passed to the switch. If there is not enough token available, the frame is marked red and dropped. This colour mapping could come handy if the yellow frames are automatically dropped in case of high network overload. A link between the two token bucket has been also provided but is mandatory. If used, if the nominal traffic token bucket overflows (because of absence of traffic) then the overflowing tokens goes to the burst bucket instead of being lost.

The behaviour of this token bucket is linked to counters that can be retrieved by the user for fault detection and/or monitoring purposes.

This meter mechanism is again very handy: it protects the network from security and safety aspects. Even if the device produces more data than what it has been specified and allowed to, the rest of the network will not be impacted and the user will be notified. Again, the only drawback is that a meter can be shared between several streams. The time and space partitioning between streams really needs to be assessed.

A.2.5 Summary

TSN protocol 802.1Qci provides a good way to handle streams in a TSN network and apply ingress policing and filtering functions to sent for security and safety purposes. The only drawback of it being that the network will have to design its system knowing that Quality of Service is applied at stream or group of stream level instead of frame level. Some time and space partitioning usually acquired when working with Virtual Links (TTEthernet, AFDX) will have to be re-discussed in TSN if Qci is used.

A.3 Fault Tolerance with 802.1CB-*Frame Replication and Elimination for Reliability*

A.3.1 Introduction

Standard Ethernet, contrarily to TTEthernet or ARINC 664, does not provide any support for redundancy (see Sections 3.6.1, 3.6.2, 3.6.3). This means that if any redundancy strategy has to be defined in an Ethernet network, up to now, it would have to be handled at application layer (ISO Layer 7). With IEEE 802.1CB - *Frame Replication and Elimination for Reliability*, the TSN working group has introduced a way to handle redundancy, for reliability purposes, at ISO Layer 2. This standard is very often called *FRER* instead of 802.1CB. In our document, the reader is very likely to find both names.

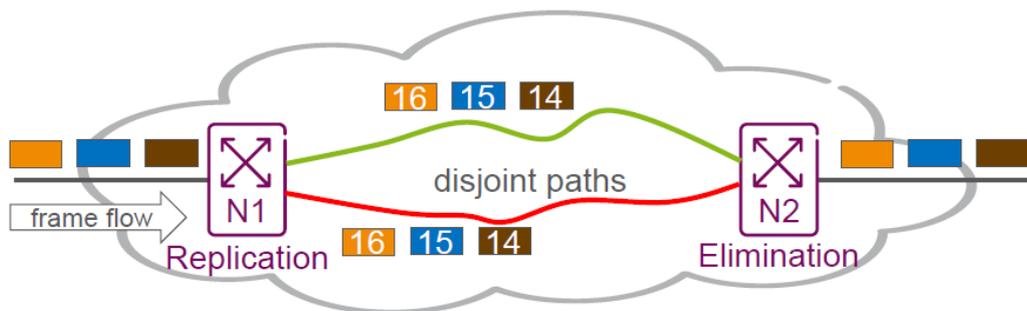


Figure A.10: FRER illustration

The overall goal of 802.1CB is to provide a protocol to avoid frame loss due to equipment failure. It is realized by duplicating frames of a stream and sending them in the network along as disjoint

as possible paths to their destination where duplicates are eliminated, in a seamless redundancy (transparent to the application).

A.3.2 Frame Format for Redundancy Support

In order to support redundancy, it is necessary to define a way to identify duplicated frames and eliminate said duplicates. In ARINC 664, a sequence number is added to any duplicated frames. Two frames from the same flow sharing the same sequences number are therefore identified as duplicates.

In 802.1CB, a similar mechanism is proposed. In fact, a field is added to any frame, in order to identify IT as duplicate. This field is the sequence number. There are three sequence number implementations proposed in the standard:

- HRS, High Seamless Redundancy,
- PRP, Parallel Redundancy Protocol,
- RTAG, Redundancy TAG.

While HRS and PRP are designed for ring topology networks, RTAG is designed for switched networks. Let us briefly describe RTAG sequence number. The RTAG field is pretty basic. It is a 6 bytes optional tag of the MAC header with the EtherType equal to *0xF1C1*. The sequence number is encoded on two bytes. The format of the RTAG and how it is included in an Ethernet frame is shown in figure A.11.

The RTAG field is the closest implementation to the existing sequence number implementation in TTEthernet and ARINC 664.

A.3.3 Difference with ARINC 664 Redundancy

There are two major differences between ARINC 664 Redundancy and 802.1CB + RTAG redundancy. First, in TSN, it is possible to have more than two duplicates whereas ARINC 664 only supports two. Second, in ARINC 664 redundancy is realised end-to-end, meaning that frame duplication is done in the source end-system and Redundancy Management & Integrity Checking is done in the receiving end-system. In TSN, it is possible to realise redundancy in a specific part of the network. For instance, one can imagine that frame are only duplicated in the core network and that reassembly is realised in the last switch in the path of any flow. Therefore, the end station only emit and receive one frame (and no duplicates). Another use case for this capability would be if a part of the network is located in an environment leading to a lot of perturbation on the physical medium, then locally, the number of duplicates could be increased to compensate these perturbations.

A.3.4 Summary

With 802.1CB, TSN proposes a way to have seamless redundancy at the cost of 6 bytes in the Ethernet frame. It allows the TSN stack to duplicate (or n-plicate) frames and send them on disjoint path to avoid single point of failures.

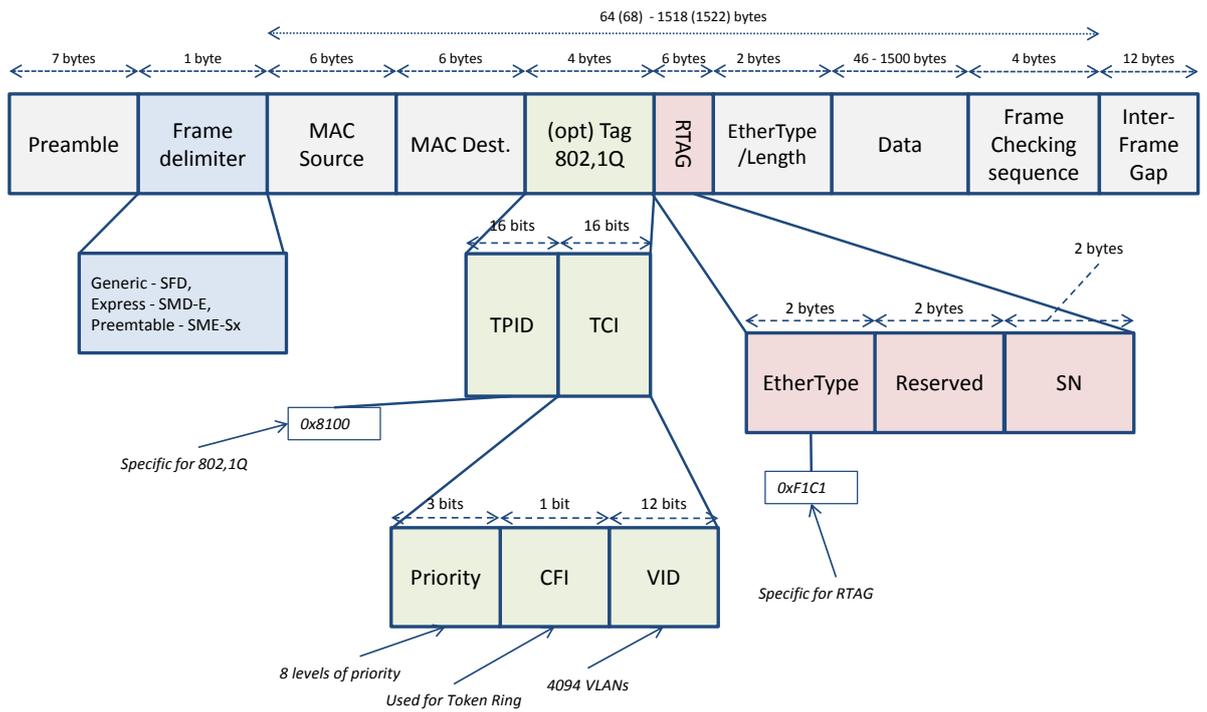


Figure A.11: Ethernet and RTAG

Appendix B

Use Cases: Flows & Flows Constraints Definitions

B.1 Airbus Generic Satellite Use Case

B.1.1 Flows

In this satellite use case, we consider that the applications running on the different devices send their messages through several flows. Let us now define these flows and their constraints with the formalism of Section 8.1. For easier readability, the flow's definition are gathered in Tables B.1 to B.5.

C & C application use case : adding flows

Name	Source	Dest	Max. Data Size	P_f
f_RIU_OBC_HK_1	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_2	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_3	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_4	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_5	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_6	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_7	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_8	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_9	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_10	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_HK_11	RIU	OBC	64	P_{MIF}
f_RIU_OBC_HK_12	RIU	OBC	64	P_{MIF}
f_RIU_OBC_FDIR_1	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_2	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_3	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_4	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_5	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_6	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_7	RIU	OBC	64	P_{MIF}
f_RIU_OBC_FDIR_8	RIU	OBC	64	P_{MIF}
f_RIU_OBC_FDIR_9	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_10	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_11	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_FDIR_12	RIU	OBC	64	8^*P_{MIF}
f_RIU_OBC_MEO_1	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_2	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_3	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_4	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_5	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_6	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_7	RIU	OBC	64	P_{MIF}
f_RIU_OBC_MEO_8	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_1	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_2	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_3	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_4	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_5	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_6	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_7	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_8	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_9	RIU	OBC	64	P_{MIF}
f_RIU_OBC_ACK_10	RIU	OBC	64	P_{MIF}
f_RIU_OBC_DATA_1	RIU	OBC	64	P_{MIF}
f_RIU_OBC_DATA_2	RIU	OBC	64	P_{MIF}

Table B.1: Flow definition for the satellite C&C use case, device RIU

Name	Source	Dest	Max. Data Size	r_f
f_OBC_RIU_HK_1	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_2	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_3	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_4	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_5	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_6	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_7	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_8	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_9	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_10	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_HK_11	OBC	RIU	64	P_{MIF}
f_OBC_RIU_HK_12	OBC	RIU	64	P_{MIF}
f_OBC_RIU_FDIR_1	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_2	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_3	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_4	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_5	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_6	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_7	OBC	RIU	64	P_{MIF}
f_OBC_RIU_FDIR_8	OBC	RIU	64	P_{MIF}
f_OBC_RIU_FDIR_9	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_10	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_11	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_FDIR_12	OBC	RIU	64	8^*P_{MIF}
f_OBC_RIU_MEO_1	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_2	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_3	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_4	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_5	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_6	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_7	OBC	RIU	64	P_{MIF}
f_OBC_RIU_MEO_8	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_1	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_2	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_3	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_4	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_5	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_6	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_7	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_8	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_9	OBC	RIU	64	P_{MIF}
f_OBC_RIU_CMD_10	OBC	RIU	64	P_{MIF}
f_OBC_RIU_ACQUL1	OBC	RIU	64	P_{MIF}
f_OBC_RIU_ACQUL2	OBC	RIU	64	P_{MIF}

Table B.2: Flow definition for the satellite C&C use case, device OBC-Part 1

Name	Source	Dest	Max. Data Size	r_f
f_STR_OBC_HK_1	STR	OBC	64	P_{MIF}
f_STR_OBC_HK_2	STR	OBC	64	P_{MIF}
f_STR_OBC_FDIR	STR	OBC	64	P_{MIF}
f_STR_OBC_MEO_1	STR	OBC	64	P_{MIF}
f_STR_OBC_MEO_2	STR	OBC	64	P_{MIF}
f_STR_OBC_ACK_1	STR	OBC	64	P_{MIF}
f_STR_OBC_ACK_2	STR	OBC	64	P_{MIF}
f_STR_OBC_ACK_3	STR	OBC	64	P_{MIF}
f_STR_OBC_ACK_4	STR	OBC	64	P_{MIF}
f_STR_OBC_ACK_5	STR	OBC	64	P_{MIF}
f_STR_OBC_DATA	STR	OBC	1088	P_{MIF}

Table B.3: Flow definition for the satellite C&C use case, device STR

Name	Source	Dest	Max. Data Size	r_f
f_OBC_STR_HK_1	OBC	STR	64	P_{MIF}
f_OBC_STR_HK_2	OBC	STR	64	P_{MIF}
f_OBC_STR_FDIR	OBC	STR	64	P_{MIF}
f_OBC_STR_MEO_1	OBC	STR	64	P_{MIF}
f_OBC_STR_MEO_2	OBC	STR	64	P_{MIF}
f_OBC_STR_CMD_1	OBC	STR	64	P_{MIF}
f_OBC_STR_CMD_2	OBC	STR	64	P_{MIF}
f_OBC_STR_CMD_3	OBC	STR	64	P_{MIF}
f_OBC_STR_CMD_4	OBC	STR	64	P_{MIF}
f_OBC_STR_CMD_5	OBC	STR	64	P_{MIF}
f_OBC_STR_ACQUI	OBC	STR	64	P_{MIF}
f_OBC_INSTR1_HK	OBC	INSTR1	128	4^*P_{MIF}
f_OBC_INSTR2_HK	OBC	INSTR2	128	4^*P_{MIF}
f_OBC_NAVCAM_HK	OBC	NAVCAM	128	4^*P_{MIF}

Table B.4: Flow definition for the satellite C&C use case, device OBC-Part 2

Name	Source	Dest	Max. Data Size	r_f
f_INSTR1_OBC_HK	OBC	INSTR1	128	4^*P_{MIF}
f_INSTR2_OBC_HK	OBC	INSTR2	128	4^*P_{MIF}
f_NAVCAM_OBC_HK	OBC	NAVCAM	128	4^*P_{MIF}

Table B.5: Flow definition for the satellite C&C use case, device INSTR1-2 and NAVCAM

B.1.2 Flow Constraints

Now that the flows of our system have been identified, let us assign them some flow constraints. Again, for easier readability, the flow constraints are gathered in Tables B.6 and B.7.

Name (Suffix)	$\Upsilon(f)$
f_* (all flows)	Deadlines("implicit")
f_* (all flows)	SafetyReq1()

Table B.6: Flow constraints definition for the satellite use case

C & C application use case : adding flows constraints

f.OBC_STR_CMD_*	$\{Jitter(1\mu s)\}$
f.OBC_STR_ACQUI_*	$\{Jitter(500\mu s)\}$
f.OBC_RIU_CMD	$\{Jitter(1\mu s)\}$
f.OBC_RIU_ACQUI	$\{Jitter(500\mu s)\}$

Table B.7: Flow constraints definition for the satellite C&C use case

B.2 Orion CEV Use Case**B.2.1 Flows**

It is composed of 100 jitter flows and 86 no jitter flows. In our modelling, multicast flows are duplicated into several unicast flows. Thus, the use case is composed of 168 unicast jitter flows and 147 unicast no jitter flows.

Name	Source	Dest	Max. Data Size	P_f
f_vl11937_STARTRB	SMBCA	STARTRB	145	$100 * P_{MIF}$
f_vl11937_FCMB	SMBCA	FCMB	145	$100 * P_{MIF}$
f_vl1506_SMRIUA	CMBCA	SMRIUA	202	$200 * P_{MIF}$
f_vl1506_CMACB	CMBCA	CMACB	202	$200 * P_{MIF}$
f_vl4331_CMBCB	DUC	CMBCB	508	$100 * P_{MIF}$
f_vl4331_FCMB	DUC	FCMB	508	$100 * P_{MIF}$
f_vl10794_CMACB	SMACA	CMACB	550	$300 * P_{MIF}$
f_vl10794_LCMA	SMACA	LCMA	550	$300 * P_{MIF}$
f_vl10964_SMACB	SMACA	SMACB	1268	$60 * P_{MIF}$
f_vl10964_DUC	SMACA	DUC	1268	$60 * P_{MIF}$
f_vl4583_MIMUC	DUC	MIMUC	742	$20 * P_{MIF}$
f_vl4583_STARTRA	DUC	STARTRA	742	$20 * P_{MIF}$
f_vl5383_SMACA	FCMB	SMACA	888	$600 * P_{MIF}$
f_vl5383_DUC	FCMB	DUC	888	$600 * P_{MIF}$
f_vl12723_DUB	SMRIUA	DUB	170	$40 * P_{MIF}$
f_vl12723_CMRIUA	SMRIUA	CMRIUA	170	$40 * P_{MIF}$
f_vl3504_DUC	DUA	DUC	773	$60 * P_{MIF}$
f_vl3504_SMACB	DUA	SMACB	773	$60 * P_{MIF}$
f_vl1266_LCMA	CMACB	LCMA	1500	$15 * P_{MIF}$
f_vl1266_SBANDA	CMACB	SBANDA	1500	$15 * P_{MIF}$
f_vl304_SBANDB	BFCU	SBANDB	1234	$600 * P_{MIF}$
f_vl3511_FCMA	DUA	FCMA	578	$15 * P_{MIF}$
f_vl3511_FCMB	DUA	FCMB	578	$15 * P_{MIF}$
f_vl10713_MIMUB	SMACA	MIMUB	1071	$24 * P_{MIF}$
f_vl7101_MIMUA	LCMB	MIMUA	748	$100 * P_{MIF}$
f_vl7101_CMBCA	LCMB	CMBCA	748	$100 * P_{MIF}$
f_vl13800_LCMB	STARTRA	LCMB	908	$120 * P_{MIF}$
f_vl2809_RCMA	CMRIUB	RCMA	186	$60 * P_{MIF}$
f_vl3764_LCMB	DUB	LCMB	1420	$600 * P_{MIF}$
f_vl3764_BFCU	DUB	BFCU	1420	$600 * P_{MIF}$
f_vl10890_CMRIUB	SMACA	CMRIUB	197	$60 * P_{MIF}$
f_vl4205_RCMB	DUC	RCMB	1097	$40 * P_{MIF}$
f_vl3812_CMACB	DUB	CMACB	246	$20 * P_{MIF}$
f_vl3812_DUA	DUB	DUA	246	$20 * P_{MIF}$
f_vl639_LCMB	CMACA	LCMB	103	$20 * P_{MIF}$
f_vl639_CMRIUB	CMACA	CMRIUB	103	$20 * P_{MIF}$
f_vl8982_CMBCB	RCMA	CMBCB	1053	$30 * P_{MIF}$

Table B.8: Flow Definition for Orion CEV use case - Part 1

Name	Source	Dest	Max. Data Size	P_f
f_vl8982_FCMB	RCMA	FCMB	1053	30^*P_{MIF}
f_vl3832_SMRIUB	DUB	SMRIUB	355	400^*P_{MIF}
f_vl3832_CMACB	DUB	CMACB	355	400^*P_{MIF}
f_vl2092_DUB	CMBCB	DUB	1173	150^*P_{MIF}
f_vl2092_SMRIUB	CMBCB	SMRIUB	1173	150^*P_{MIF}
f_vl11710_CMACA	SMBCA	CMACA	87	120^*P_{MIF}
f_vl11710_SMRIUB	SMBCA	SMRIUB	87	120^*P_{MIF}
f_vl10101_RCMB	SBANDA	RCMB	913	120^*P_{MIF}
f_vl8883_MIMUC	RCMA	MIMUC	233	60^*P_{MIF}
f_vl8883_BFCU	RCMA	BFCU	233	60^*P_{MIF}
f_vl14362_RCMA	STARTRB	RCMA	14362	120^*P_{MIF}
f_vl14362_SBANDA	STARTRB	SBANDA	14362	120^*P_{MIF}
f_vl3758_DUC	DUB	DUC	654	60^*P_{MIF}
f_vl3758_BFCU	DUB	BFCU	654	60^*P_{MIF}
f_vl3185_MIMUC	CMRIUB	MIMUC	131	60^*P_{MIF}
f_vl3185_SMBCB	CMRIUB	SMBCB	131	60^*P_{MIF}
f_vl11127_SBANDA	SMACA	SBANDA	682	60^*P_{MIF}
f_vl11127_SMBCA	SMACA	SMBCA	682	60^*P_{MIF}
f_vl8358_SMRIUB	MIMUB	SMRIUB	1259	200^*P_{MIF}
f_vl8358_SMBCA	MIMUB	SMBCA	1259	200^*P_{MIF}
f_vl9266_SBANDB	RCMA	SBANDB	498	150^*P_{MIF}
f_vl9266_SMACB	RCMA	SMACB	498	150^*P_{MIF}
f_vl9977_CMRIUB	SBANDA	CMRIUB	595	12^*P_{MIF}
f_vl9977_STARTRB	SBANDA	STARTRB	595	12^*P_{MIF}
f_vl7398_SBANDA	LCMB	SBANDA	186	120^*P_{MIF}
f_vl7398_SMACB	LCMB	SMACB	186	120^*P_{MIF}
f_vl11654_STARTRA	SMBCA	STARTRA	744	60^*P_{MIF}
f_vl8306_RCMA	MIMUB	RCMA	295	120^*P_{MIF}
f_vl8306_SBANDA	MIMUB	SBANDA	295	120^*P_{MIF}
f_vl3834_CMACB	DUB	CMACB	520	30^*P_{MIF}
f_vl3834_STARTRB	DUB	STARTRB	520	30^*P_{MIF}
f_vl11997_LCMB	SMBCA	LCMB	974	12^*P_{MIF}
f_vl11997_SMRIUB	SMBCA	SMRIUB	974	12^*P_{MIF}
f_vl6964_STARTRA	LCMA	STARTRA	526	120^*P_{MIF}
f_vl6964_SMBCA	LCMA	SMBCA	526	120^*P_{MIF}
f_vl8753_MIMUA	MIMUC	MIMUA	354	150^*P_{MIF}
f_vl8753_SMBCB	MIMUC	SMBCB	354	150^*P_{MIF}
f_vl11888_DUC	SMBCA	DUC	438	24^*P_{MIF}

Table B.9: Flow Definition for Orion CEV use case - Part 2

Name	Source	Dest	Max. Data Size	P_f
f_vl11888_MIMUB	SMBCA	MIMUB	438	24^*P_{MIF}
f_vl1522_CMBCB	CMBCA	CMBCB	560	150^*P_{MIF}
f_vl1522_MIMUB	CMBCA	MIMUB	560	150^*P_{MIF}
f_vl11808_CMRIUA	SMBCA	CMRIUA	310	200^*P_{MIF}
f_vl11808_SMACA	SMBCA	SMACA	310	200^*P_{MIF}
f_vl4115_MIMUC	DUB	MIMUC	1260	30^*P_{MIF}
f_vl4115_SMBCB	DUB	SMBCB	1260	30^*P_{MIF}
f_vl13760_DUC	STARTRA	DUC	502	400^*P_{MIF}
f_vl13760_STARTRB	STARTRA	STARTRB	502	400^*P_{MIF}
f_vl13493_DUA	STARTRA	DUA	291	15^*P_{MIF}
f_vl4419_DUA	DUC	DUA	1468	12^*P_{MIF}
f_vl4419_STARTRB	DUC	STARTRB	1468	12^*P_{MIF}
f_vl780_MIMUA	CMACA	MIMUA	411	60^*P_{MIF}
f_vl2666_SMBCB	CMRIUA	SMBCB	498	150^*P_{MIF}
f_vl2666_LCMA	CMRIUA	LCMA	498	150^*P_{MIF}
f_vl11727_MIMUB	SMBCA	MIMUB	447	60^*P_{MIF}
f_vl11727_CMACB	SMBCA	CMACB	447	60^*P_{MIF}
f_vl6641_SBANDA	LCMA	SBANDA	305	12^*P_{MIF}
f_vl6641_CMBCA	LCMA	CMBCA	305	12^*P_{MIF}
f_vl8229_SMBCA	MIMUB	SMBCA	732	240^*P_{MIF}
f_vl9247_SBANDB	RCMA	SBANDB	810	120^*P_{MIF}
f_vl9247_RCMB	RCMA	RCMB	810	120^*P_{MIF}
f_vl10050_RCMB	SBANDA	RCMB	850	12^*P_{MIF}
f_vl10050_FCMA	SBANDA	FCMA	850	12^*P_{MIF}
f_vl13172_CMBCB	SMRIUB	CMBCB	738	50^*P_{MIF}
f_vl13172_MIMUA	SMRIUB	MIMUA	738	50^*P_{MIF}
f_vl2733_SMRIUB	CMRIUA	SMRIUB	138	200^*P_{MIF}
f_vl2733_RCMA	CMRIUA	RCMA	138	200^*P_{MIF}
f_vl3305_BFCU	DUA	BFCU	135	100^*P_{MIF}
f_vl3305_SBANDA	DUA	SBANDA	135	100^*P_{MIF}
f_vl1476_CMACA	CMBCA	CMACA	1124	300^*P_{MIF}
f_vl1476_SMACB	CMBCA	SMACB	1124	300^*P_{MIF}
f_vl9008_CMRIUA	RCMA	CMRIUA	455	40^*P_{MIF}
f_vl4444_FCMA	DUC	FCMA	764	60^*P_{MIF}
f_vl4444_LCMA	DUC	LCMA	764	60^*P_{MIF}
f_vl9239_MIMUC	RCMA	MIMUC	1132	300^*P_{MIF}
f_vl9239_SMACB	RCMA	SMACB	1132	300^*P_{MIF}
f_vl5556_SMACA	FCMB	SMACA	1003	300^*P_{MIF}

Table B.10: Flow Definition for Orion CEV use case - Part 3

Name	Source	Dest	Max. Data Size	P_f
f_vl5556_SMRIUA	FCMB	SMRIUA	1003	$300 * P_{MIF}$
f_vl14198_SBANDB	STARTRB	SBANDB	1249	$20 * P_{MIF}$
f_vl14198_DUB	STARTRB	DUB	1249	$20 * P_{MIF}$
f_vl10922_DUA	SMACA	DUA	646	$400 * P_{MIF}$
f_vl10922_SBANDB	SMACA	SBANDB	646	$400 * P_{MIF}$
f_vl11054_LCMA	SMACA	LCMA	1131	$30 * P_{MIF}$
f_vl11054_STARTRA	SMACA	STARTRA	1131	$30 * P_{MIF}$
f_vl4241_BFCU	DUC	BFCU	643	$40 * P_{MIF}$
f_vl4241_SMRIUA	DUC	SMRIUA	643	$40 * P_{MIF}$
f_vl8342_SBANDB	MIMUB	SBANDB	912	$200 * P_{MIF}$
f_vl8342_STARTRB	MIMUB	STARTRB	912	$200 * P_{MIF}$
f_vl5179_CMACA	FCMB	CMACA	1352	$40 * P_{MIF}$
f_vl5179_CMRIUB	FCMB	CMRIUB	1352	$40 * P_{MIF}$
f_vl10267_BFCU	SBANDB	BFCU	1374	$24 * P_{MIF}$
f_vl10267_DUA	SBANDB	DUA	1374	$24 * P_{MIF}$
f_vl2493_CMRIUB	CMRIUA	CMRIUB	740	$12 * P_{MIF}$
f_vl2493_DUC	CMRIUA	DUC	740	$12 * P_{MIF}$
f_vl2114_STARTRA	CMBCB	STARTRA	1389	$40 * P_{MIF}$
f_vl2114_DUC	CMBCB	DUC	1389	$40 * P_{MIF}$
f_vl3703_SMACB	DUA	SMACB	1336	$240 * P_{MIF}$
f_vl3703_SMRIUB	DUA	SMRIUB	1336	$240 * P_{MIF}$
f_vl7343_MIMUA	LCMB	MIMUA	1288	$400 * P_{MIF}$
f_vl7343_SMACB	LCMB	SMACB	1288	$400 * P_{MIF}$
f_vl14211_DUC	STARTRB	DUC	888	$240 * P_{MIF}$
f_vl14211_LCMB	STARTRB	LCMB	888	$240 * P_{MIF}$
f_vl3760_BFCU	DUB	BFCU	736	$12 * P_{MIF}$
f_vl3760_FCMB	DUB	FCMB	736	$12 * P_{MIF}$
f_vl8837_CMACA	RCMA	CMACA	1492	$200 * P_{MIF}$
f_vl4727_CMACA	FCMA	CMACA	502	$60 * P_{MIF}$
f_vl4727_RCMB	FCMA	RCMB	502	$60 * P_{MIF}$
f_vl4656_CMRIUA	FCMA	CMRIUA	920	$100 * P_{MIF}$
f_vl7770_LCMA	MIMUA	LCMA	431	$400 * P_{MIF}$
f_vl493_SMRIUB	CMACA	SMRIUB	824	$600 * P_{MIF}$
f_vl13336_MIMUA	SMRIUB	MIMUA	397	$15 * P_{MIF}$
f_vl14149_CMRIUB	STARTRB	CMRIUB	757	$600 * P_{MIF}$
f_vl14149_MIMUA	STARTRB	MIMUA	757	$600 * P_{MIF}$
f_vl13503_MIMUB	STARTRA	MIMUB	654	$120 * P_{MIF}$
f_vl404_RCMA	BFCU	RCMA	814	$40 * P_{MIF}$

Table B.11: Flow Definition for Orion CEV use case - Part 4

Name	Source	Dest	Max. Data Size	P_f
f_vl404_SMACB	BFCU	SMACB	814	$40 * P_{MIF}$
f_vl14085_SMACB	STARTRB	SMACB	616	$600 * P_{MIF}$
f_vl14085_CMBCA	STARTRB	CMBCA	616	$600 * P_{MIF}$
f_vl3262_CMRIUB	DUA	CMRIUB	387	$20 * P_{MIF}$
f_vl13418_MIMUB	SMRIUB	MIMUB	1239	$120 * P_{MIF}$
f_vl13418_STARTRA	SMRIUB	STARTRA	1239	$120 * P_{MIF}$
f_vl9151_MIMUA	RCMA	MIMUA	460	$100 * P_{MIF}$
f_vl1353_SBANDA	CMACB	SBANDA	526	$150 * P_{MIF}$
f_vl1353_SMACB	CMACB	SMACB	526	$150 * P_{MIF}$
f_vl11304_CMBCB	SMACB	CMBCB	608	$200 * P_{MIF}$
f_vl11304_DUB	SMACB	DUB	608	$200 * P_{MIF}$
f_vl8174_DUC	MIMUB	DUC	212	$30 * P_{MIF}$
f_vl8174_SMACB	MIMUB	SMACB	212	$30 * P_{MIF}$
f_vl13722_FCMB	STARTRA	FCMB	599	$120 * P_{MIF}$
f_vl13722_DUB	STARTRA	DUB	599	$120 * P_{MIF}$
f_vl161_SMBCB	BFCU	SMBCB	1017	$120 * P_{MIF}$
f_vl161_CMRIUA	BFCU	CMRIUA	1017	$120 * P_{MIF}$
f_vl2066_DUA	CMBCB	DUA	1290	$200 * P_{MIF}$
f_vl2066_SMACB	CMBCB	SMACB	1290	$200 * P_{MIF}$
f_vl12229_STARTRA	SMBCB	STARTRA	1442	$200 * P_{MIF}$
f_vl12229_CMBCA	SMBCB	CMBCA	1442	$200 * P_{MIF}$
f_vl2622_LCMB	CMRIUA	LCMB	526	$200 * P_{MIF}$
f_vl10932_DUB	SMACA	DUB	193	$200 * P_{MIF}$
f_vl10932_FCMB	SMACA	FCMB	193	$200 * P_{MIF}$
f_vl12148_BFCU	SMBCB	BFCU	1426	$200 * P_{MIF}$
f_vl12148_STARTRA	SMBCB	STARTRA	1426	$200 * P_{MIF}$
f_vl282_FCMB	BFCU	FCMB	778	$200 * P_{MIF}$
f_vl282_MIMUC	BFCU	MIMUC	778	$200 * P_{MIF}$
f_vl5300_SMBCB	FCMB	SMBCB	1336	$200 * P_{MIF}$
f_vl5300_CMRIUA	FCMB	CMRIUA	1336	$200 * P_{MIF}$
f_vl2781_SMBCB	CMRIUA	SMBCB	959	$200 * P_{MIF}$
f_vl2781_SMRIUA	CMRIUA	SMRIUA	959	$200 * P_{MIF}$
f_vl7161_CMRIUA	LCMB	CMRIUA	930	$200 * P_{MIF}$
f_vl7161_SMRIUA	LCMB	SMRIUA	930	$200 * P_{MIF}$
f_vl2140_MIMUA	CMBCB	MIMUA	1263	$200 * P_{MIF}$
f_vl2140_FCMB	CMBCB	FCMB	1263	$200 * P_{MIF}$
f_vl5524_RCMA	FCMB	RCMA	1079	$200 * P_{MIF}$
f_vl5524_STARTRA	FCMB	STARTRA	1079	$200 * P_{MIF}$

Table B.12: Flow Definition for Orion CEV use case - Part 5

Name	Source	Dest	Max. Data Size	P_f
f_vl12545_STARTRB	SMBCB	STARTRB	1015	$200 * P_{MIF}$
f_vl12545_SMACB	SMBCB	SMACB	1015	$200 * P_{MIF}$
f_vl8554_SMBCA	MIMUC	SMBCA	185	$200 * P_{MIF}$
f_vl8554_CMRIUA	MIMUC	CMRIUA	185	$200 * P_{MIF}$
f_vl1941_CMACA	CMBCB	CMACA	118	$200 * P_{MIF}$
f_vl1941_SMACB	CMBCB	SMACB	118	$200 * P_{MIF}$
f_vl4167_SMRIUA	DUB	SMRIUA	1438	$200 * P_{MIF}$
f_vl4167_SMACB	DUB	SMACB	1438	$200 * P_{MIF}$
f_vl12254_STARTRA	SMBCB	STARTRA	311	$200 * P_{MIF}$
f_vl12254_CMBCB	SMBCB	CMBCB	311	$200 * P_{MIF}$
f_vl3300_MIMUA	DUA	MIMUA	89	$200 * P_{MIF}$
f_vl3300_BFCU	DUA	BFCU	89	$200 * P_{MIF}$
f_vl10460_DUA	SBANDB	DUA	1213	$200 * P_{MIF}$
f_vl10460_SMBCB	SBANDB	SMBCB	1213	$200 * P_{MIF}$
f_vl11430_SMBCA	SMACB	SMBCA	821	$200 * P_{MIF}$
f_vl11430_DUC	SMACB	DUC	821	$200 * P_{MIF}$
f_vl11844_DUA	SMBCA	DUA	1376	$200 * P_{MIF}$
f_vl11844_LCMA	SMBCA	LCMA	1376	$200 * P_{MIF}$
f_vl812_LCMB	CMACA	LCMB	1438	$200 * P_{MIF}$
f_vl812_MIMUB	CMACA	MIMUB	1438	$200 * P_{MIF}$
f_vl9461_SMBCB	RCMB	SMBCB	891	$200 * P_{MIF}$
f_vl9461_CMBCB	RCMB	CMBCB	891	$200 * P_{MIF}$
f_vl5098_SMRIUB	FCMA	SMRIUB	950	$200 * P_{MIF}$
f_vl5098_SMACB	FCMA	SMACB	950	$200 * P_{MIF}$
f_vl7630_DUA	MIMUA	DUA	832	$200 * P_{MIF}$
f_vl7630_CMRIUB	MIMUA	CMRIUB	832	$200 * P_{MIF}$
f_vl3535_FCMB	DUA	FCMB	185	$200 * P_{MIF}$
f_vl3535_MIMUA	DUA	MIMUA	185	$200 * P_{MIF}$
f_vl3534_LCMB	DUA	LCMB	592	$200 * P_{MIF}$
f_vl3534_FCMB	DUA	FCMB	592	$200 * P_{MIF}$
f_vl2254_SMBCA	CMBCB	SMBCA	932	$200 * P_{MIF}$
f_vl2254_MIMUC	CMBCB	MIMUC	932	$200 * P_{MIF}$
f_vl2867_CMACA	CMRIUB	CMACA	1431	$200 * P_{MIF}$
f_vl2867_RCMB	CMRIUB	RCMB	1431	$200 * P_{MIF}$
f_vl8016_CMACB	MIMUB	CMACB	459	$200 * P_{MIF}$
f_vl8016_SMRIUA	MIMUB	SMRIUA	459	$200 * P_{MIF}$
f_vl9263_SBANDA	RCMA	SBANDA	240	$200 * P_{MIF}$
f_vl9263_STARTRA	RCMA	STARTRA	240	$200 * P_{MIF}$

Table B.13: Flow Definition for Orion CEV use case - Part 6

Name	Source	Dest	Max. Data Size	P_f
f_vl2694_SMBCA	CMRIUA	SMBCA	313	$200 * P_{MIF}$
f_vl2694_MIMUA	CMRIUA	MIMUA	313	$200 * P_{MIF}$
f_vl7388_SMACA	LCMB	SMACA	993	$200 * P_{MIF}$
f_vl7388_RCMB	LCMB	RCMB	993	$200 * P_{MIF}$
f_vl5124_DUB	FCMB	DUB	1277	$200 * P_{MIF}$
f_vl10472_MIMUA	SBANDB	MIMUA	577	$200 * P_{MIF}$
f_vl10472_DUB	SBANDB	DUB	577	$200 * P_{MIF}$
f_vl2858_FCMB	CMRIUB	FCMB	867	$200 * P_{MIF}$
f_vl2858_CMACA	CMRIUB	CMACA	867	$200 * P_{MIF}$
f_vl11403_DUB	SMACB	DUB	1354	$200 * P_{MIF}$
f_vl11403_MIMUB	SMACB	MIMUB	1354	$200 * P_{MIF}$
f_vl3130_SMBCA	CMRIUB	SMBCA	547	$200 * P_{MIF}$
f_vl3130_LCMA	CMRIUB	LCMA	547	$200 * P_{MIF}$
f_vl2183_SMACB	CMBCB	SMACB	619	$200 * P_{MIF}$
f_vl14103_CMBCB	STARTRB	CMBCB	350	$200 * P_{MIF}$
f_vl14103_MIMUB	STARTRB	MIMUB	350	$200 * P_{MIF}$
f_vl11696_CMACA	SMBCA	CMACA	376	$200 * P_{MIF}$
f_vl1410_LCMB	CMBCA	LCMB	529	$200 * P_{MIF}$
f_vl14214_MIMUC	STARTRB	MIMUC	487	$200 * P_{MIF}$
f_vl14214_DUC	STARTRB	DUC	487	$200 * P_{MIF}$
f_vl11262_MIMUB	SMACB	MIMUB	523	$200 * P_{MIF}$
f_vl11262_CMACB	SMACB	CMACB	523	$200 * P_{MIF}$
f_vl118_DUB	BFCU	DUB	759	$200 * P_{MIF}$
f_vl118_CMBCB	BFCU	CMBCB	759	$200 * P_{MIF}$
f_vl63_CMRIUB	BFCU	CMRIUB	1360	$200 * P_{MIF}$
f_vl63_CMACB	BFCU	CMACB	1360	$200 * P_{MIF}$
f_vl11126_SBANDA	SMACA	SBANDA	963	$200 * P_{MIF}$
f_vl11126_SMACB	SMACA	SMACB	963	$200 * P_{MIF}$
f_vl9127_SMRIUB	RCMA	SMRIUB	125	$200 * P_{MIF}$
f_vl9127_FCMA	RCMA	FCMA	125	$200 * P_{MIF}$
f_vl3845_CMBCA	DUB	CMBCA	1046	$200 * P_{MIF}$
f_vl3845_LCMB	DUB	LCMB	1046	$200 * P_{MIF}$
f_vl3421_CMRIUA	DUA	CMRIUA	1058	$200 * P_{MIF}$
f_vl3421_CMRIUB	DUA	CMRIUB	1058	$200 * P_{MIF}$
f_vl11724_LCMA	SMBCA	LCMA	440	$200 * P_{MIF}$
f_vl11724_CMACB	SMBCA	CMACB	440	$200 * P_{MIF}$
f_vl9095_LCMA	RCMA	LCMA	863	$200 * P_{MIF}$
f_vl9095_DUC	RCMA	DUC	863	$200 * P_{MIF}$

Table B.14: Flow Definition for Orion CEV use case - Part 7

Name	Source	Dest	Max. Data Size	P_f
f_vl19876_SMRIUA	SBANDA	SMRIUA	537	200^*P_{MIF}
f_vl19876_CMACB	SBANDA	CMACB	537	200^*P_{MIF}
f_vl2703_MIMUB	CMRIUA	MIMUB	784	200^*P_{MIF}
f_vl2703_SBANDA	CMRIUA	SBANDA	784	200^*P_{MIF}
f_vl4412_DUA	DUC	DUA	234	200^*P_{MIF}
f_vl4412_SMACA	DUC	SMACA	234	200^*P_{MIF}
f_vl11923_FCMB	SMBCA	FCMB	729	200^*P_{MIF}
f_vl11923_LCMB	SMBCA	LCMB	729	200^*P_{MIF}
f_vl10338_SMACB	SBANDB	SMACB	1193	200^*P_{MIF}
f_vl10338_CMACB	SBANDB	CMACB	1193	200^*P_{MIF}
f_vl13830_STARTRB	STARTRA	STARTRB	751	200^*P_{MIF}
f_vl2264_SMACB	CMBCB	SMACB	147	200^*P_{MIF}
f_vl2264_RCMA	CMBCB	RCMA	147	200^*P_{MIF}
f_vl11129_SBANDA	SMACA	SBANDA	757	200^*P_{MIF}
f_vl11129_SMRIUA	SMACA	SMRIUA	757	200^*P_{MIF}
f_vl80_SMACA	BFCU	SMACA	170	200^*P_{MIF}
f_vl80_CMACB	BFCU	CMACB	170	200^*P_{MIF}
f_vl10336_SBANDA	SBANDB	SBANDA	177	200^*P_{MIF}
f_vl10336_CMACB	SBANDB	CMACB	177	200^*P_{MIF}
f_vl13030_DUC	SMRIUB	DUC	1293	200^*P_{MIF}
f_vl7596_CMBCB	MIMUA	CMBCB	1154	200^*P_{MIF}
f_vl7596_SBANDA	MIMUA	SBANDA	1154	200^*P_{MIF}
f_vl8470_CMACB	MIMUC	CMACB	1189	200^*P_{MIF}
f_vl8470_LCMB	MIMUC	LCMB	1189	200^*P_{MIF}
f_vl1147_DUB	CMACB	DUB	1040	200^*P_{MIF}
f_vl8832_STARTRB	MIMUC	STARTRB	1499	200^*P_{MIF}
f_vl8832_SMRIUA	MIMUC	SMRIUA	1499	200^*P_{MIF}
f_vl10379_CMBCB	SBANDB	CMBCB	773	200^*P_{MIF}
f_vl6521_FCMA	LCMA	FCMA	621	200^*P_{MIF}
f_vl7783_SMRIUB	MIMUA	SMRIUB	780	200^*P_{MIF}
f_vl7505_DUA	MIMUA	DUA	1062	200^*P_{MIF}
f_vl7505_CMACA	MIMUA	CMACA	1062	200^*P_{MIF}
f_vl2336_FCMB	CMRIUA	FCMB	462	200^*P_{MIF}
f_vl19873_CMACB	SBANDA	CMACB	1018	200^*P_{MIF}
f_vl19873_SMACB	SBANDA	SMACB	1018	200^*P_{MIF}
f_vl4338_MIMUC	DUC	MIMUC	668	200^*P_{MIF}
f_vl4338_CMBCB	DUC	CMBCB	668	200^*P_{MIF}
f_vl17830_MIMUC	MIMUA	MIMUC	772	200^*P_{MIF}

Table B.15: Flow Definition for Orion CEV use case - Part 8

Name	Source	Dest	Max. Data Size	P_f
f_vl7830_SBANDA	MIMUA	SBANDA	772	$200 * P_{MIF}$
f_vl3212_SMACA	CMRIUB	SMACA	414	$200 * P_{MIF}$
f_vl3212_SBANDA	CMRIUB	SBANDA	414	$200 * P_{MIF}$
f_vl13888_MIMUC	STARTRA	MIMUC	539	$200 * P_{MIF}$
f_vl13888_SBANDB	STARTRA	SBANDB	539	$200 * P_{MIF}$
f_vl8638_DUC	MIMUC	DUC	989	$200 * P_{MIF}$
f_vl8638_SMACA	MIMUC	SMACA	989	$200 * P_{MIF}$
f_vl11589_STARTRB	SMACB	STARTRB	1242	$200 * P_{MIF}$
f_vl11589_RCMB	SMACB	RCMB	1242	$200 * P_{MIF}$
f_vl8457_CMACA	MIMUC	CMACA	729	$200 * P_{MIF}$
f_vl8457_STARTRB	MIMUC	STARTRB	729	$200 * P_{MIF}$
f_vl8325_STARTRB	MIMUB	STARTRB	145	$200 * P_{MIF}$

Table B.16: Flow Definition for Orion CEV use case - Part 9

B.2.2 Flow Constraints

All flows have implicit deadlines and are subject to the safety requirement. In addition, the flows of Table B.17, B.18, B.19, B.20 and B.21 have jitter constraints.

f_vl11937_STARTRB	Jitter(1 μ s)
f_vl11937_FCMB	Jitter(1 μ s)
f_vl1506_SMRIUA	Jitter(1 μ s)
f_vl1506_CMACH	Jitter(1 μ s)
f_vl4331_CMBCB	Jitter(1 μ s)
f_vl4331_FCMB	Jitter(1 μ s)
f_vl10794_CMACH	Jitter(1 μ s)
f_vl10794_LCMA	Jitter(1 μ s)
f_vl10964_SMACH	Jitter(1 μ s)
f_vl10964_DUC	Jitter(1 μ s)
f_vl4583_MIMUC	Jitter(1 μ s)
f_vl4583_STARTRA	Jitter(1 μ s)
f_vl5383_SMACA	Jitter(1 μ s)
f_vl5383_DUC	Jitter(1 μ s)
f_vl12723_DUB	Jitter(1 μ s)
f_vl12723_CMRIUA	Jitter(1 μ s)
f_vl3504_DUC	Jitter(1 μ s)
f_vl3504_SMACH	Jitter(1 μ s)
f_vl1266_LCMA	Jitter(1 μ s)
f_vl1266_SBANDA	Jitter(1 μ s)
f_vl304_SBANDB	Jitter(1 μ s)
f_vl3511_FCMA	Jitter(1 μ s)
f_vl3511_FCMB	Jitter(1 μ s)
f_vl10713_MIMUB	Jitter(1 μ s)
f_vl7101_MIMUA	Jitter(1 μ s)
f_vl7101_CMBCA	Jitter(1 μ s)
f_vl13800_LCMB	Jitter(1 μ s)
f_vl2809_RCMA	Jitter(1 μ s)
f_vl3764_LCMB	Jitter(1 μ s)
f_vl3764_BFCU	Jitter(1 μ s)
f_vl10890_CMRIUB	Jitter(1 μ s)
f_vl4205_RCMB	Jitter(1 μ s)
f_vl3812_CMACH	Jitter(1 μ s)
f_vl3812_DUA	Jitter(1 μ s)
f_vl639_LCMB	Jitter(1 μ s)
f_vl639_CMRIUB	Jitter(1 μ s)
f_vl8982_CMBCB	Jitter(1 μ s)
f_vl8982_FCMB	Jitter(1 μ s)

Table B.17: Flow QoS for Orion CEV Use Case - Part 1

f_vl3832_SMRIUB	Jitter(1 μ s)
f_vl3832_CMACB	Jitter(1 μ s)
f_vl2092_DUB	Jitter(1 μ s)
f_vl2092_SMRIUB	Jitter(1 μ s)
f_vl11710_CMACA	Jitter(1 μ s)
f_vl11710_SMRIUB	Jitter(1 μ s)
f_vl10101_RCMB	Jitter(1 μ s)
f_vl8883_MIMUC	Jitter(1 μ s)
f_vl8883_BFCU	Jitter(1 μ s)
f_vl14362_RCMA	Jitter(1 μ s)
f_vl14362_SBANDA	Jitter(1 μ s)
f_vl3758_DUC	Jitter(1 μ s)
f_vl3758_BFCU	Jitter(1 μ s)
f_vl3185_MIMUC	Jitter(1 μ s)
f_vl3185_SMBCB	Jitter(1 μ s)
f_vl11127_SBANDA	Jitter(1 μ s)
f_vl11127_SMBCA	Jitter(1 μ s)
f_vl8358_SMRIUB	Jitter(1 μ s)
f_vl8358_SMBCA	Jitter(1 μ s)
f_vl9266_SBANDB	Jitter(1 μ s)
f_vl9266_SMACB	Jitter(1 μ s)
f_vl9977_CMRIUB	Jitter(1 μ s)
f_vl9977_STARTRB	Jitter(1 μ s)
f_vl7398_SBANDA	Jitter(1 μ s)
f_vl7398_SMACB	Jitter(1 μ s)
f_vl11654_STARTRA	Jitter(1 μ s)
f_vl8306_RCMA	Jitter(1 μ s)
f_vl8306_SBANDA	Jitter(1 μ s)
f_vl3834_CMACB	Jitter(1 μ s)
f_vl3834_STARTRB	Jitter(1 μ s)
f_vl11997_LCMB	Jitter(1 μ s)
f_vl11997_SMRIUB	Jitter(1 μ s)
f_vl6964_STARTRA	Jitter(1 μ s)
f_vl6964_SMBCA	Jitter(1 μ s)
f_vl8753_MIMUA	Jitter(1 μ s)
f_vl8753_SMBCB	Jitter(1 μ s)
f_vl11888_DUC	Jitter(1 μ s)
f_vl11888_MIMUB	Jitter(1 μ s)

Table B.18: Flow QoS for Orion CEV Use Case - Part 2

f_vl1522_CMBCB	Jitter(1 μ s)
f_vl1522_MIMUB	Jitter(1 μ s)
f_vl11808_CMRIUA	Jitter(1 μ s)
f_vl11808_SMACA	Jitter(1 μ s)
f_vl4115_MIMUC	Jitter(1 μ s)
f_vl4115_SMBCB	Jitter(1 μ s)
f_vl13760_DUC	Jitter(1 μ s)
f_vl13760_STARTRB	Jitter(1 μ s)
f_vl13493_DUA	Jitter(1 μ s)
f_vl4419_DUA	Jitter(1 μ s)
f_vl4419_STARTRB	Jitter(1 μ s)
f_vl780_MIMUA	Jitter(1 μ s)
f_vl2666_SMBCB	Jitter(1 μ s)
f_vl2666_LCMA	Jitter(1 μ s)
f_vl11727_MIMUB	Jitter(1 μ s)
f_vl11727_CMACB	Jitter(1 μ s)
f_vl6641_SBANDA	Jitter(1 μ s)
f_vl6641_CMBCA	Jitter(1 μ s)
f_vl8229_SMBCA	Jitter(1 μ s)
f_vl9247_SBANDB	Jitter(1 μ s)
f_vl9247_RCMB	Jitter(1 μ s)
f_vl10050_RCMB	Jitter(1 μ s)
f_vl10050_FCMA	Jitter(1 μ s)
f_vl13172_CMBCB	Jitter(1 μ s)
f_vl13172_MIMUA	Jitter(1 μ s)
f_vl2733_SMRIUB	Jitter(1 μ s)
f_vl2733_RCMA	Jitter(1 μ s)
f_vl3305_BFCU	Jitter(1 μ s)
f_vl3305_SBANDA	Jitter(1 μ s)
f_vl1476_CMACA	Jitter(1 μ s)
f_vl1476_SMACB	Jitter(1 μ s)
f_vl9008_CMRIUA	Jitter(1 μ s)
f_vl4444_FCMA	Jitter(1 μ s)
f_vl4444_LCMA	Jitter(1 μ s)
f_vl9239_MIMUC	Jitter(1 μ s)
f_vl9239_SMACB	Jitter(1 μ s)
f_vl5556_SMACA	Jitter(1 μ s)
f_vl5556_SMRIUA	Jitter(1 μ s)

Table B.19: Flow QoS for Orion CEV Use Case - Part 3

f_vl14198_SBANDB	Jitter(1 μ s)
f_vl14198_DUB	Jitter(1 μ s)
f_vl10922_DUA	Jitter(1 μ s)
f_vl10922_SBANDB	Jitter(1 μ s)
f_vl11054_LCMA	Jitter(1 μ s)
f_vl11054_STARTRA	Jitter(1 μ s)
f_vl4241_BFCU	Jitter(1 μ s)
f_vl4241_SMRIUA	Jitter(1 μ s)
f_vl8342_SBANDB	Jitter(1 μ s)
f_vl8342_STARTRB	Jitter(1 μ s)
f_vl5179_CMACA	Jitter(1 μ s)
f_vl5179_CMRIUB	Jitter(1 μ s)
f_vl10267_BFCU	Jitter(1 μ s)
f_vl10267_DUA	Jitter(1 μ s)
f_vl2493_CMRIUB	Jitter(1 μ s)
f_vl2493_DUC	Jitter(1 μ s)
f_vl2114_STARTRA	Jitter(1 μ s)
f_vl3703_SMACB	Jitter(1 μ s)
f_vl3703_SMRIUB	Jitter(1 μ s)
f_vl7343_MIMUA	Jitter(1 μ s)
f_vl14211_LCMB	Jitter(1 μ s)
f_vl3760_BFCU	Jitter(1 μ s)
f_vl3760_FCMB	Jitter(1 μ s)
f_vl8837_CMACA	Jitter(1 μ s)
f_vl4727_CMACA	Jitter(1 μ s)
f_vl4727_RCMB	Jitter(1 μ s)
f_vl4656_CMRIUA	Jitter(1 μ s)
f_vl7770_LCMA	Jitter(1 μ s)
f_vl13336_MIMUA	Jitter(1 μ s)
f_vl14149_CMRIUB	Jitter(1 μ s)
f_vl14149_MIMUA	Jitter(1 μ s)
f_vl13503_MIMUB	Jitter(1 μ s)
f_vl404_RCMA	Jitter(1 μ s)
f_vl14085_CMBCA	Jitter(1 μ s)
f_vl3262_CMRIUB	Jitter(1 μ s)
f_vl13418_MIMUB	Jitter(1 μ s)
f_vl13418_STARTRA	Jitter(1 μ s)
f_vl11304_CMBCB	Jitter(1 μ s)

Table B.20: Flow QoS for Orion CEV Use Case - Part 4

f_vl11304_DUB	Jitter(1 μ s)
f_vl13722_FCMB	Jitter(1 μ s)
f_vl13722_DUB	Jitter(1 μ s)
f_vl161_SMBCB	Jitter(1 μ s)
f_vl161_CMRIUA	Jitter(1 μ s)

Table B.21: Flow QoS for Orion CEV Use Case - Part 5

Appendix C

A Step Further: Relaxing Hypotheses

C.1 Relaxing Hypothesis 2

In this manuscript, we have chosen through *Hypothesis 2* that the maximum size of any message of any flow could not be greater than the Ethernet MTU. In fact, this allows us to state that one applicative message corresponds to one Ethernet frame. Relaxing Hyp. 2 will lead to having one applicative message being split (or fragmented) into several Ethernet frames. In that case, the concepts of *Production instants* and *Delivery instants* of a message shall be redefined.

Definition 79 *Production Date*

Let f be a flow ($\in \mathbb{F}$), where f does not satisfies Hyp. 2. Let f_l be the l th message of f . Let us denote $Frames(f_l)$ the set of Ethernet frames corresponding to f_l . Let $First(Frames(f_l))$ (resp. $Last(Frames(f_l))$) the first (resp. last) frame corresponding to f_l . The production instant of f_l , i.e. $T_p(f_l)$, correspond to the production instant (at applicative level) of the first bit of $First(Frames(f_l))$.

Definition 80 *Delivery Date*

Let f be a flow ($\in \mathbb{F}$), where f does not satisfies Hyp.2. Let f_l be the l th message of f . Let us denote $Frames(f_l)$ the set of Ethernet frames corresponding to f_l . Let $First(Frames(f_l))$ (resp. $Last(Frames(f_l))$) the first (resp. last) frame corresponding to f_l . The delivery date of f_l , i.e. $T_r(f_l)$, correspond to the delivery instant (at applicative level) of the last bit of $Last(Frames(f_l))$.

With these newly adapted definitions, Hyp. 2 can be relaxed. The constraints introduced in the manuscript are based on the concept of production instant and delivery instant. Since they have just been redefined, the size of applicative message is no longer limited to the Ethernet MTU.

Appendix D

Résumé Long Français

Table des matières

1	Introduction	3
2	Contexte	7
2.1	Généralités sur les satellites	7
2.1.1	Les satellites et leurs missions	7
2.1.2	Le concept "Plateforme et Charge Utile"	8
2.1.3	L'architecture de référence SAVOIR, un effort de standardisation au niveau Européen	8
2.1.4	Vue d'ensemble d'une architecture réseau générique d'un satellite	9
2.2	Time Sensitive Networking, l'état de l'art IEEE pour Ethernet	10
2.2.1	Vue d'ensemble	11
2.2.2	Focus sur IEEE 802.1Qbv	12
3	Première contribution : Sélection de technologies compatibles avec les besoins en qualité de service des réseaux satellite nouvelle génération	17
3.1	Problème 1	17
3.1.1	Technologies présélectionnées	17
3.1.2	Propriétés Haut Niveau	18
3.1.3	Aperçu de la Contribution	18
3.2	Critères pour la comparaison	19
3.2.1	Critères pour la propriété 1	19
3.2.2	Critères pour la propriété 2	19
3.2.3	Critères pour la propriété 3	20
3.3	Résultats et Analyse	21
4	Seconde contribution : Génération de configuration d'un réseau TSN pour des satellites nouvelle génération	25
4.1	Problème 2	25
4.1.1	Modélisation des applications	26
4.1.2	Modélisation des flux	26
4.1.3	Exigences de qualité de service	28
4.1.4	Considérations Industrielles	30
4.1.5	Aperçu de la contribution	31
4.2	Egress TT, une nouvelle approche pour la génération de configuration de réseaux à faible gigue	32
4.2.1	Qu'est-ce qu'Egress TT ?	33
4.2.2	Comment fonctionne Egress TT ?	34

4.2.3	Quel est l'intérêt de Egress TT?	34
4.2.4	Applicabilité de Egress TT	35
4.3	Exclusive Queue Allocation et Size Based Isolation deux implémentations de Egress TT pour les réseaux TSN	35
4.3.1	Une première implémentation de Egress TT pour les réseaux 802.1Qbv : Exclusive Queue Allocation	36
4.3.2	Une seconde implémentation pour l'amélioration du passage à l'échelle d'Exclusive Queue Allocation : Size Based Isolation	37
4.3.3	Limitations de Exclusive Queue Allocation, Size Based Isolation et Egress TT	39
4.3.4	Évaluation Expérimentale de la performance de Egress TT par rapport à l'état de l'art	39
4.3.5	Configurations de systèmes industriels	42
5	Conclusion	47

Chapitre 1

Introduction

Cette thèse a été financée par Airbus Defence and Space (dénommé l'industriel dans ce document) et l'ANRT (Association Nationale pour la Recherche Technologique). Elle est incluse dans la feuille de route TANIA-DP (Technological Assessment for New Instruments and Avionics - Data Processing) d'Airbus, qui a pour objectif, entre autres, de fournir de nouveaux standards de communication pour les réseaux embarqués satellite afin d'améliorer les performances, les coûts, la compatibilité avec les standards du segment sol, ou encore la gestion de futures missions dans l'espace.

Contexte

A l'image de l'augmentation du volume de données générées et manipulées par nos équipements sur Terre (téléphones, voitures, instruments de mesure scientifique, etc.), les satellites doivent eux aussi être capables de produire et de transmettre des quantités de données de plus en plus grandes afin de répondre aux besoins des entreprises qui les opèrent. On trouve déjà sur le marché des appareils de grande capacité, telles les caméras multi-gigabit, et de ce fait la transition technologique des instruments embarqués vers davantage de performance n'est pas un problème en soi. Néanmoins, l'augmentation de la capacité de transmission de ces données, elle, pose problème. C'est la raison pour laquelle ce document s'intéresse à la modernisation des réseaux embarqués lesquels doivent répondre à deux besoins différents. D'une part, ils sont en charge de véhiculer les informations nécessaires au bon fonctionnement du satellite (ex. contrôle des propulseurs, contrôle du système de communication, contrôle des panneaux solaires, etc.). D'autre part, ils transportent les données acquises par les instruments embarqués dans le satellite (comme par exemple des télescopes, des capteurs météorologiques, etc.) vers les antennes du satellite qui les retransmettent aux stations-sol. La plupart du temps, l'architecture réseau embarquée repose sur deux technologies : MIL-STD-1553 et SpaceWire.

Un besoin de renouveau

L'actuelle architecture du réseau embarqué satellite a démontré ses forces et sa maturité après 15 ans d'utilisation. Cependant, elle commence maintenant à montrer ses limites, que ce soit en terme de débit, de coût de développement et de maintenance, de disponibilité des composants dit "sur-étagère" (*COTS - Components-Of-The-Shelf*), ou encore de potentiel d'interaction avec d'autres

industries ou avec la communauté scientifique. C'est pourquoi l'industrie aérospatiale envisage de renouveler ses réseaux embarqués pour les futures générations de satellites.

À ce jour, la future architecture se baserait sur un réseau "unifié", c'est-à-dire qu'une seule technologie serait utilisée pour répondre aux besoins des futures missions, tant sur les aspects de gestion du fonctionnement nominal du satellite que du transfert de données des instruments. Cette technologie devra non-seulement avoir de meilleures performances que le 1553 (abréviation de MIL-STD-1553) et le SpaceWire, mais aussi être facile à analyser et à configurer, faciliter le développement et l'intégration des satellites. De plus, elle devra participer à la réduction du coût global du satellite. Plusieurs technologies ont été présélectionnées par l'industriel pour créer ce réseau "unifié" nouvelle génération : Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking et Spacefibre.

Parmi ces technologies, une opportunité est récemment apparue au début de la thèse avec Time Sensitive Networking (abrégé en TSN), le dernier standard Ethernet de l'IEEE. Ce standard serait capable de répondre à la fois aux besoins temps réels et haut débit des échanges à bord du satellite. C'est la raison pour laquelle ce sujet de thèse, intitulé "*Adaptabilité de Time Sensitive Networking aux exigences de l'industrie aérospatiale*", a été créé. Cette activité est totalement nouvelle dans le domaine spatial puisque TSN n'avait jamais auparavant été identifié ou bien considéré comme un candidat potentiel pour le renouvellement des réseaux embarqués satellite.

Notre Approche

Afin d'évaluer le potentiel de Time Sensitive Networking vis à vis des besoins envisagés pour les nouvelles générations de satellites, nous proposons un raisonnement en deux étapes.

Dans une première étape, nous souhaitons nous assurer, à l'aide d'une analyse qualitative, qu'il est bien raisonnable de considérer TSN comme candidat pour le réseau unifié nouvelle génération. Pour ce faire, nous allons définir un ensemble de propriétés haut niveau représentatives des exigences des futures missions. A l'image des méthodologies de comparaisons similaires dans l'état de l'art, nous allons ensuite détailler ces propriétés en critères. Enfin, nous discuterons de l'adaptabilité des technologies présélectionnées par l'industriel (i.e. Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking et Spacefibre) vis à vis des critères précédemment définis.

Dans une deuxième étape, nous analyserons en profondeur la compatibilité de TSN en raffinant puis en formalisant les besoins en qualité de service du réseau unifié nouvelle génération. Nous mettrons en évidence la compatibilité de cette technologie en générant une ou plusieurs configurations du réseau de systèmes représentatifs. S'il n'est pas possible de générer des configurations qui répondent aux exigences en qualité de service, cela signifierait que TSN n'est pas un bon choix pour les futurs satellites. Afin de générer ces configurations, nous nous inspirerons des méthodologies de configuration existantes dans l'état de l'art, basées sur des émissions planifiées de messages et de la programmation par contraintes, afin de proposer notre propre méthode de configuration. Ce ne sera pas chose aisée. En effet, TSN est composé de nombreux standards, incluant chacun de très nombreux paramètres à prendre en compte pour générer une configuration. Dans l'objectif de réduire l'effort de configuration dans cette thèse, nous nous sommes limités à la configuration du standard TSN IEEE 802.1Qbv uniquement. Ce raisonnement est illustré dans la figure Fig. 1.1¹.

1. Les chapitres indiqués dans cette figure sont ceux du manuscrit

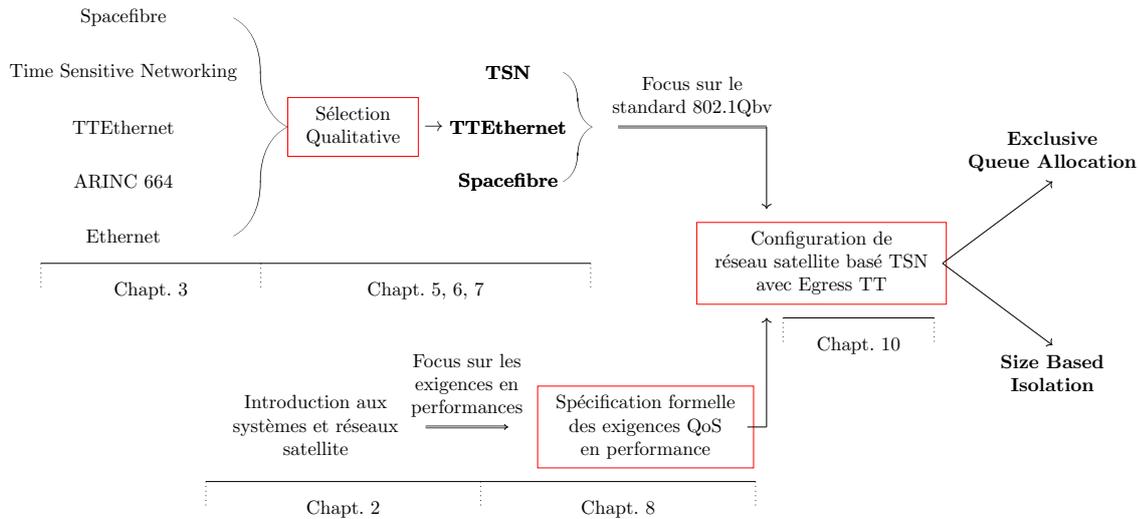


FIGURE 1.1 – Processus de sélection d’une technologie pour le réseau unifié nouvelle génération

Plan du résumé

Ce document résume en français quelques éléments de contexte ainsi que les principales contributions présentées dans le manuscrit de thèse. Le chapitre 2 présente des généralités sur les satellites et leurs réseaux embarqués ainsi que le standard TSN 802.1Qbv. Ensuite, le troisième chapitre définit formellement le premier problème, correspondant à la première étape de notre raisonnement. Il résume ensuite les principaux éléments de la première contribution du manuscrit. Elle consiste en une sélection qualitative de technologies compatibles avec les besoins de satellites nouvelle génération. Dans le chapitre 4, le second problème, correspondant à la deuxième étape de notre raisonnement est présenté. Dans la suite du chapitre, nous résumons la deuxième contribution, c’est à dire la génération de configurations d’un réseau TSN adapté aux exigences de futures missions. Pour cela, nous présentons notre nouvelle méthodologie de configuration intitulée Egress TT et l’appliquons aux réseaux TSN de deux manières différentes intitulées Exclusive Queue Allocation et Size Based Isolation.

Chapitre 2

Contexte

Ce chapitre présente des éléments de contexte nécessaires pour la compréhension de la problématique et des contributions. Il résume la Partie 1 (Part 1) du manuscrit. Dans la première section, nous introduisons des généralités sur les satellites et sur les réseaux embarqués à bord de ces satellites. Dans la seconde section, nous présentons quelques généralités sur la technologie Time Sensitive Networking puis nous détaillons certains éléments du fonctionnement du standard IEEE 802.1Qbv.

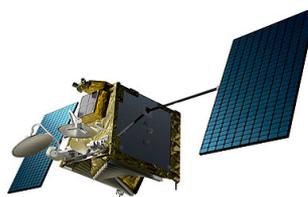
2.1 Généralités sur les satellites

2.1.1 Les satellites et leurs missions

Un satellite est un équipement électronique envoyé par l'Homme dans l'espace. En orbite autour de la terre, les satellites sont majoritairement utilisés pour des applications de télécommunications (e.x. téléphone avec *Inmarsat*, télévision avec *Eutelsat*, internet avec *OneWeb*), des applications d'observations de la Terre (militaires, services d'imageries avec *Pleiade Neo*, étude des océans avec *Sentinel-6B*) et des applications de positionnement (pour les GPS notamment). Dans le système solaire, les satellites sont utilisés à des fins scientifiques (étude du Soleil avec *Solar Orbiter*, étude de Mercure avec *BepiColombo*, étude de Jupiter et ses lunes avec *Juice*). Le format du satellite (poids, taille), son orbite et sa mission varient : tandis que des satellites en orbite géostationnaire sont d'énormes véhicules (cf. Fig. 2.1a) conçus pour une durée de vie opérationnelle d'au moins quinze ans ; les satellites en orbites basses (cf. Fig. 2.1b) sont eux beaucoup plus petits et ont des durées de vies plus courtes (jusqu'à cinq ans).



(a) Satellite SES 12 ($\sim 5500kg$, $\sim 54m^3$)



(b) Satellite OneWeb ($\sim 150kg$, $\sim 1m^3$)

FIGURE 2.1 – Différents formats de satellites

2.1.2 Le concept "Plateforme et Charge Utile"

Un satellite est généralement composé de deux parties : la *Plateforme (Platform)* et la *Charge Utile (Payload)*, comme représenté dans la figure Fig. 2.2

D'une part, la charge utile (*payload* en anglais) est la raison d'être du satellite. C'est la partie du satellite qui génère de la valeur ajoutée pour son propriétaire. La charge utile est spécifique à chaque satellite et à chaque mission. En général, la partie *Payload* est composée d'instruments comme des antennes ou des transpondeurs pour un satellite de télécommunication, de télescopes ou de caméras pour un satellite d'observation de la terre, ou bien d'instruments scientifiques pour des missions scientifiques (ex. capteur de radiation, capteur de champ magnétique, etc.).

D'autre part, la plateforme (*platform* en anglais) est l'infrastructure qui permet au satellite de poursuivre sa mission. C'est le cœur du satellite. Si cette partie venait à dysfonctionner, le satellite pourrait ne plus fonctionner correctement voire être perdu. En général, la partie *Platform* est composée de tous les systèmes et sous-systèmes qui assurent le bon fonctionnement du satellite (ex. le contrôle de l'attitude et de l'orbite, le contrôle de la distribution de l'énergie, le contrôle et la surveillance de la santé de la charge utile, le contrôle des communications avec les stations sols, etc.).

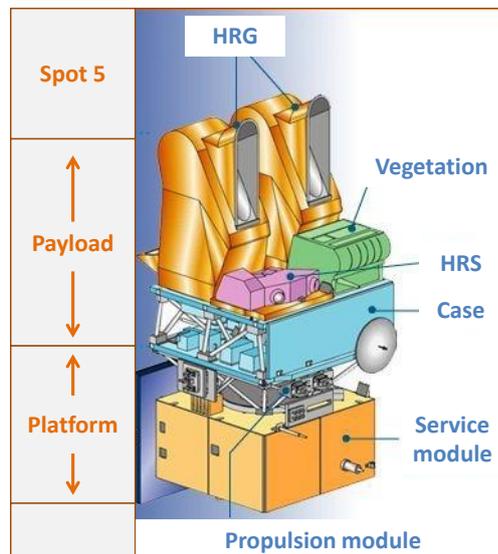


FIGURE 2.2 – *Platform* et *Payload* sur un satellite Spot-5

2.1.3 L'architecture de référence SAVOIR, un effort de standardisation au niveau Européen

Au niveau européen, l'architecture fonctionnelle de référence SAVOIR, conçue sur la base d'exigences de plusieurs types de missions (scientifique, télécommunication, observation de la terre, etc.), est une référence importante quand on traite de la conception d'architecture bord pour les satellites. Cette référence est représentée dans la figure Fig. 2.3. L'architecture du réseau proposée dans le paragraphe suivant est conforme à cette référence.

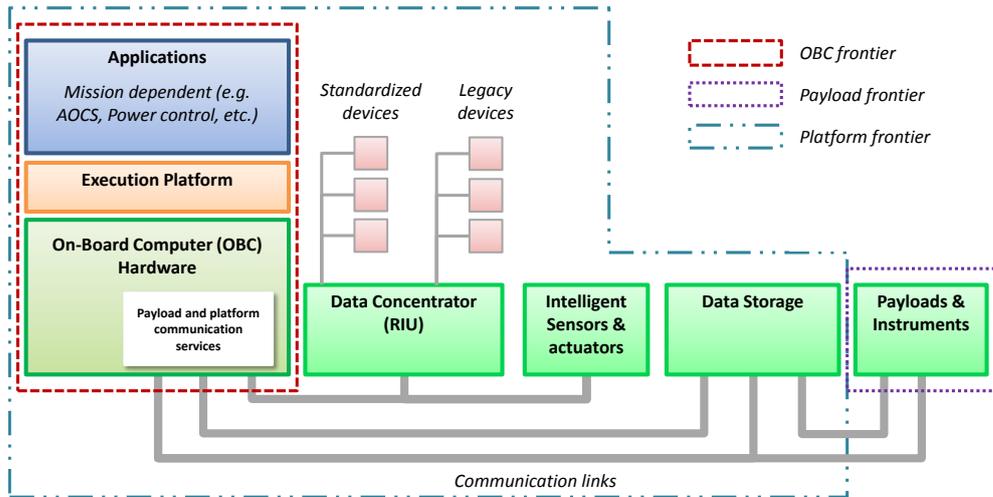


FIGURE 2.3 – Architecture de référence SAVOIR

2.1.4 Vue d'ensemble d'une architecture réseau générique d'un satellite

Cette section présente une architecture réseau qui servira de base pour notre étude. Le réseau est composé de différents équipements qui répondent à différentes fonctions interconnectées par des liens de communication. Cette architecture est considérée suffisamment générique et représentative des futurs satellites.

A l'image du satellite, le réseau embarqué est composé de deux réseaux interconnectés : le réseau plateforme et le réseau charge utile. Chacun de ces réseaux répond à des besoins différents et parfois même opposés.

D'une part, le réseau plateforme, représenté en rouge et en violet dans la figure Fig. 2.4, est chargé de transporter toutes les informations nécessaires au fonctionnement nominal du satellite. Il transporte à la fois des données provenant de différents capteurs (position, température, etc.) ainsi que les commandes de contrôle de la trajectoire du satellite. Ce trafic, souvent décrit comme temps réel, nécessite des durées de transmissions (latences) bornées et une faible variabilité de cette latence (gigue). Cependant, de par la faible taille et le faible nombre de message, une petite bande passante suffit pour les transporter. En général, le réseau plateforme repose sur la technologie MIL-STD-1553 ou bien de la technologie CAN.

D'autre part, le réseau charge utile, représenté en vert et orange dans la figure Fig. 2.4, nécessite une grande bande passante pour pouvoir transporter les gros volumes de données brutes, telles que des images, de la télémétrie météo, des données IoT, générées par les instruments à bord. Les contraintes sont plus lâches pour le réseau charge utile : un délai dans la transmission d'un message n'aura pas d'impact sur le fonctionnement nominal du satellite. En général, la technologie sous-jacente pour la charge utile est la technologie SpaceWire.

Les liens de communications de ces réseaux sont répliqués deux ou quatre fois selon les missions afin d'assurer une redondance froide. Cela signifie qu'un seul des liens n'est actif à la fois. Si ce lien venait à dysfonctionner, il serait remplacé par l'activation d'un autre lien.

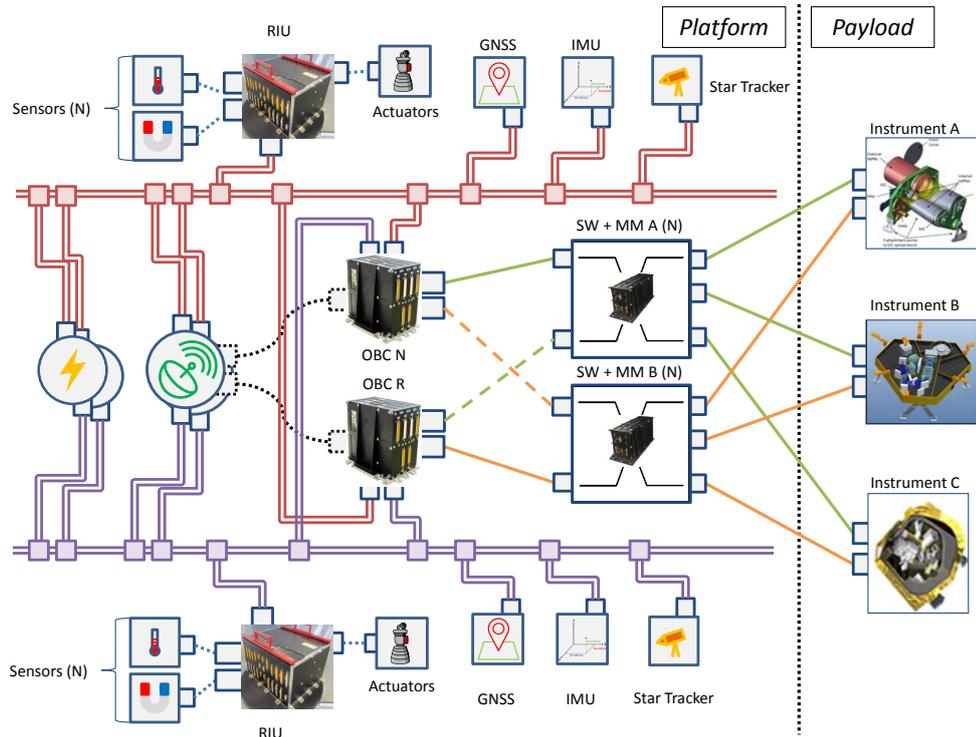


FIGURE 2.4 – Architecture générique d'un réseau satellite

Récapitulatif

Nous avons proposé dans cette section un survol rapide de ce qu'est un satellite, du concept de *Plateforme & Charge Utile* ainsi que du modèle de référence SAVOIR. Nous avons ensuite présenté l'architecture réseau générique sur laquelle notre étude se base et les deux réseaux (plateforme et charge utile) qui permettent les échanges de données à bord.

Bien que ces réseaux aient répondu aux exigences de l'industrie aérospatiale durant les 15 années passées, ils commencent à montrer leurs limites, en particulier en terme de bande passante. C'est pourquoi, ils doivent être renouvelés. Pour ce faire, nous allons étudier une liste de technologies précédemment sélectionnées par notre partenaire industriel afin de déterminer les candidats potentiels pour remplacer les technologies actuelles. Parmi ces technologies, présentées dans le chapitre 3 du manuscrit, se trouve Time Sensitive Networking, que nous introduisons plus en détail dans la section suivante.

2.2 Time Sensitive Networking, l'état de l'art IEEE pour Ethernet

Time Sensitive Networking (abrégé en TSN), est la toute dernière technologie de la famille Ethernet de l'IEEE qui prétend être capable de transporter à la fois du trafic temps réel et du

trafic haut débit. Elle est pour l'heure toujours en cours de développement par un groupe de travail nommé l'IEEE TSN Task Group. Ce groupe de travail a repris les activités du précédent groupe AVB (pour *Audio Video Bridging*) fondé en 2012. Le TSN Task Group a publié, au moment d'écrire ce résumé, plus d'une vingtaine d'amendements ou de standards de la famille IEEE 802.1 ayant pour objectif de définir un réseau qui serait à la fois temps réel, adaptable et flexible, mais aussi capable de mélanger des approches synchrones et asynchrones. Cette section résume les éléments présentés dans le chapitre 4 du manuscrit.

2.2.1 Vue d'ensemble

L'image de la figure 2.5, résume l'état des standards TSN au mois de novembre 2021. Cette liste de standards est encore en train d'évoluer à l'heure où nous écrivons ce manuscrit.

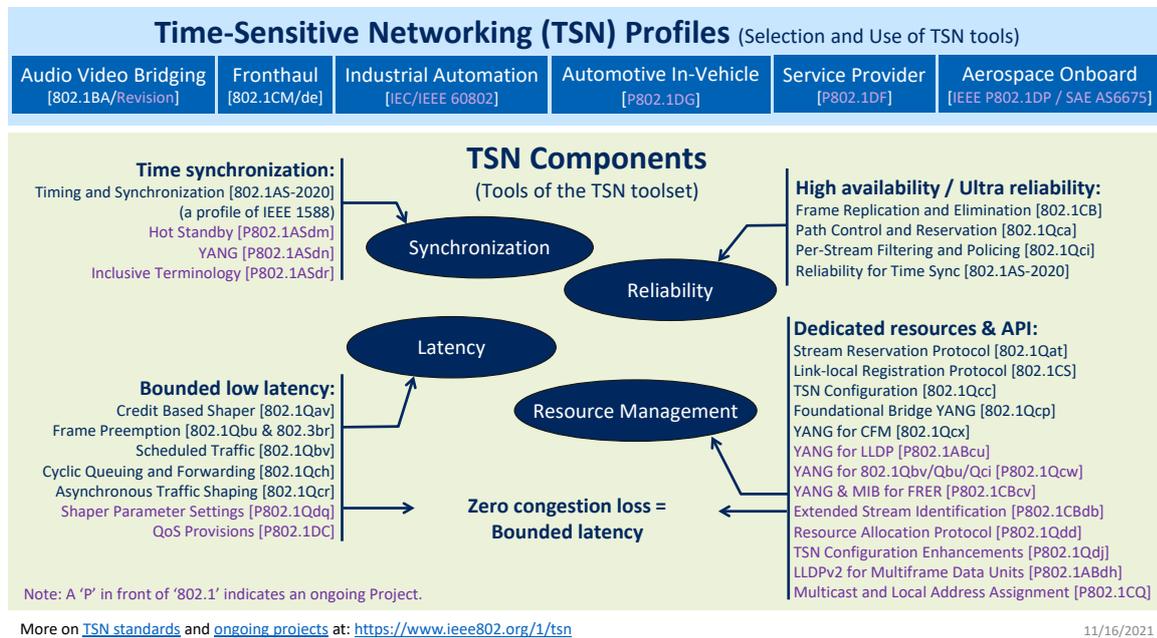


FIGURE 2.5 – État des standards TSN en Nov. 2021

Les standards, amendements et projets en cours, que nous nommerons tous standards par la suite pour des facilités d'écriture, peuvent être organisés en cinq familles : Synchronisation, Fiabilité, Latence, Gestion des Ressources et "Pas de pertes liées aux congestions". Un bref résumé des standards de chaque famille est disponible dans le manuscrit. S'il ne fallait retenir qu'un message au sujet de cette vue d'ensemble de TSN, ce serait le suivant : TSN n'est pas encore une nouvelle technologie, bien au contraire, elle se base sur le fonctionnement classique de l'Ethernet Switché Full Duplex (voir la section 3.6 dans le manuscrit) décrit dans IEEE 802.1Q; en y rajoutant des protocoles qui amendent ou améliorent le mode de fonctionnement existant.

2.2.2 Focus sur IEEE 802.1Qbv

Dans le restant de cette section, nous nous intéresserons à un standard de la famille "Latency" intitulé *IEEE 802.1Qbv - Enhancement for Scheduled Traffic*. Comme son nom le suggère, ce standard propose des améliorations aux mécanismes de contrôle de flux dans les ports de sortie des commutateurs et des stations terminales avec pour objectif d'offrir la capacité de programmer l'émission de messages.

Remarque 1. Bien que *801.1Qbv* soit maintenant inclus dans *802.1Q-2018*, dans ce résumé et dans le manuscrit, nous continuons à utiliser l'acronyme "*802.1Qbv*" pour ne pas créer de confusion. En effet, *802.1Q-2018* inclus bien plus que simplement *802.1Qbv*.

Introduction

Les ports de sorties améliorés avec *802.1Qbv* disposent maintenant de 8 files internes (aussi appelées *classes de trafic* ou encore *files de transmissions* dans le standard) dans lesquelles les messages (ou *trames* dans le vocabulaire Ethernet) peuvent être placés dans l'attente de leur émission.

Chacune de ces files est associée, à la fois avec un algorithme de sélection de transmission (*Transmission Selection Algorithm - TSA*) et avec une porte de transmission (*Transmission Gate - TG*). C'est la combinaison de ces deux mécanismes qui permettra de décider, au sein d'une file, quand une trame est autorisée à tenter d'accéder au médium physique. Cette architecture de port améliorée est présentée dans la figure Fig. 2.6.

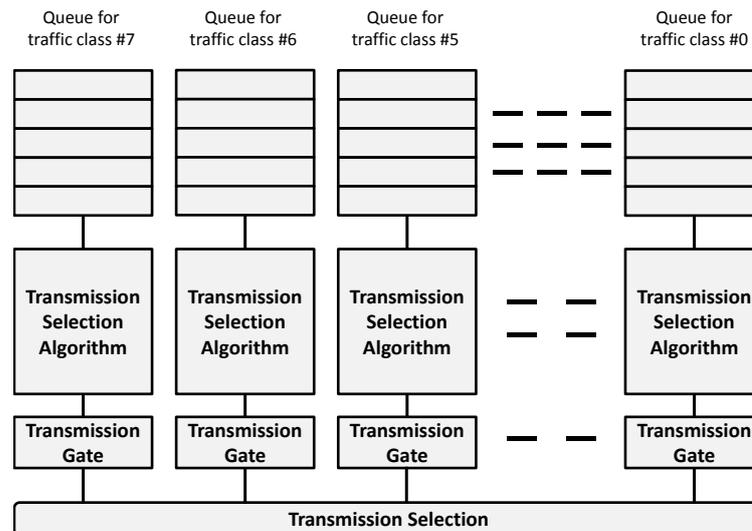


FIGURE 2.6 – Port 802.1Qbv d'un switch

Quand une trame arrive dans un port *802.1Qbv*, elle est placée dans une file. Le choix de la file est important et pourra se paramétrer dans la configuration (voir Annexe A.2 dans le manuscrit). Ces files sont les structures de données qui stockeront les trames dans l'attente de disponibilité du

lien physique. Les files sont numérotées et ordonnées en priorité depuis la file #7 jusqu'à la file #0. Chaque file a un comportement spécifique défini par son TSA et son TG.

Définition 1 (Disponible pour la transmission). *Pour qu'une trame soit autorisée à tenter d'accéder au lien physique, celle-ci doit être marquée "Disponible pour la transmission". Pour qu'elle soit marquée de la sorte, cela nécessite plusieurs conditions :*

1. la trame doit être en tête de sa file,
2. le TSA de la file doit avoir autorisé la transmission de la trame,
3. le TG de la file doit avoir autorisé la transmission la trame.

Puisqu'un port dispose de plusieurs files, il est possible que plusieurs trames, retenues dans des files différentes, soit marquées "disponibles pour la transmission" en même temps. De fait, il est nécessaire de définir une stratégie d'arbitrage entre les files d'un même port. La stratégie implémentée dans TSN est *Static Priority* (priorité statique) i.e. la trame dans la file avec la plus haute priorité sera émise en premier suivie des autres trames par ordre de priorité décroissant.

Intéressons-nous maintenant aux *Transmission Gates*. Dans cette thèse nous n'avons pas considéré d'algorithme de sélection de transmission (TSA), ils ne sont donc de fait pas présentés dans ce résumé. Néanmoins, ils sont détaillés dans le manuscrit.

Transmission Gates

Le mécanisme de Transmission Gates est le deuxième élément qui rentre en jeu dans le processus de marquage des trames en "disponible pour la transmission". Il est souvent appelé *Time Aware Shaper* ou *TAS*. En effet, chaque file est associée avec une porte qui peut être ouverte ou fermée à des instants et pendant des durées configurables. Cela permet de décider quand et de combien de temps chaque file d'un port a la possibilité d'émettre des trames. Détaillons un peu plus le fonctionnement de ces portes. Une porte de transmission peut avoir deux états : Ouverte (o) ou Fermée (C). Quand la porte d'une file est :

- Fermée → même si le TSA de la file autorise l'émission de trames, aucune trame de cette file ne peut tenter d'accéder au lien physique,
- Ouverte → la trame en tête de la file sera marquée "Disponible pour la transmission" si le TSA de la file le permet et qu'il reste assez de temps pour émettre cette trame avant la fermeture de la porte.

Il devient clair avec l'explication ci-dessus que l'état des portes doit évoluer au cours du temps. En effet, si une porte reste fermée en permanence, cela signifie que le trafic placé dans la file associée à cette porte ne sera jamais émis. Dans ce contexte, le port amélioré par 802.1Qbv dispose d'une structure intitulée liste de contrôle de portes (*Gate Control List* ou *GCL*) qui décrira l'évolution du statut des portes de chaque file d'un port au court du temps. Les paramètres présentés dans le standard pour la configuration des GCL sont détaillés dans le manuscrit.

La configuration du standard 802.1Qbv

La diversité des algorithmes de sélection de transmission et des ordonnancements de files définis par les GCLs créent un immense champ des possibles pour la configuration du standard 802.1Qbv. Tandis qu'imager une configuration du standard semble possible, trouver une configuration (i.e. une sélection de valeurs des paramètres pour les TSA et les GCL) qui satisfasse les exigences de qualité de service n'est pas simple. La figure Fig. 2.7 propose une vue synthétique des mécanismes configurables dans 802.1Qbv.

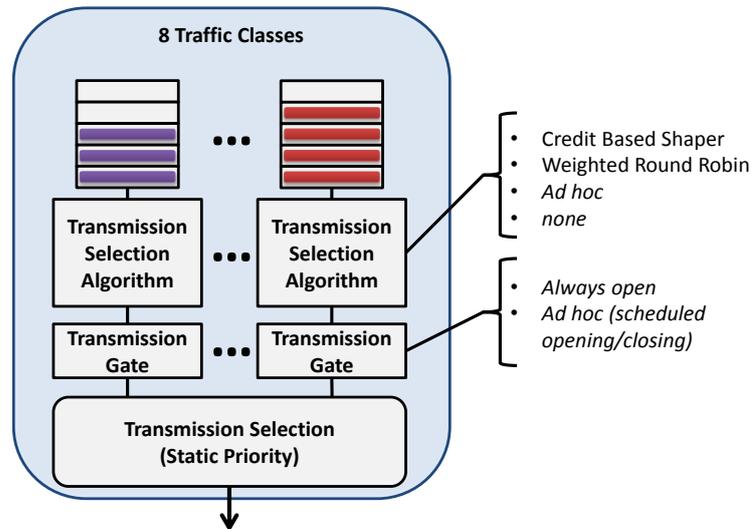


FIGURE 2.7 – Mécanismes configurables dans 802.1Qbv

Configurations courante de 802.1Qbv

Nous présentons dans la suite du résumé français une configuration courante du standard 802.1Qbv. Dans le manuscrit, nous en introduisons plusieurs autres, provenant toutes de l'état de l'art. Ces configurations ou *Architectures of Interest* sont en somme des modes d'utilisation du Qbv où un choix de TSA et de l'usage ou non de TG a été fait.

End-to-End TT. Dans cette configuration dite End-to-End TT, tous les ports utilisent des transmission gates (régies par des GCL) mais n'utilisent pas de TSA. En l'absence de TSA, une trame en tête de file est automatiquement marquée "Disponible pour la transmission" si la porte de sa file est ouverte. L'évolution du statut des portes au cours du temps décrit donc la manière avec laquelle les files sont servies et de fait, les instants auxquels les trames tentent d'accéder au lien physique.

Parmi ces configurations End-to-End TT, une sous-famille est très représentée dans l'état de l'art : les configurations dites en exclusion mutuelle de portes (exclusive gating).

Définition 2 (Exclusive Gating). *Dans ces configurations, les GCL sont conçues de manière bien spécifiques : quand la porte d'une file est ouverte, la porte de toutes les autres files sont fermées et inversement. Cette exclusion mutuelle permet d'éviter les conflits entre les trames "Disponibles pour la transmission" pour l'accès au médium.*

Nous avons représenté dans la Fig. 2.8 une visualisation de la GCL d'un port donné qui implémente la stratégie d'exclusion mutuelle pour deux files. Quand la porte d'une de ces deux files est ouverte, les portes de toutes les autres files sont fermées et inversement. Par contre, quand les portes de ces deux files sont fermées, les portes de toutes les autres files sont ouvertes.

En résumé

En résumé, le standard TSN 802.1Qbv - *Enhancement for Scheduled Traffic* ajoute, dans tous les ports qui l'implémentent, non seulement de nouveaux mécanismes pour le partage de la bande

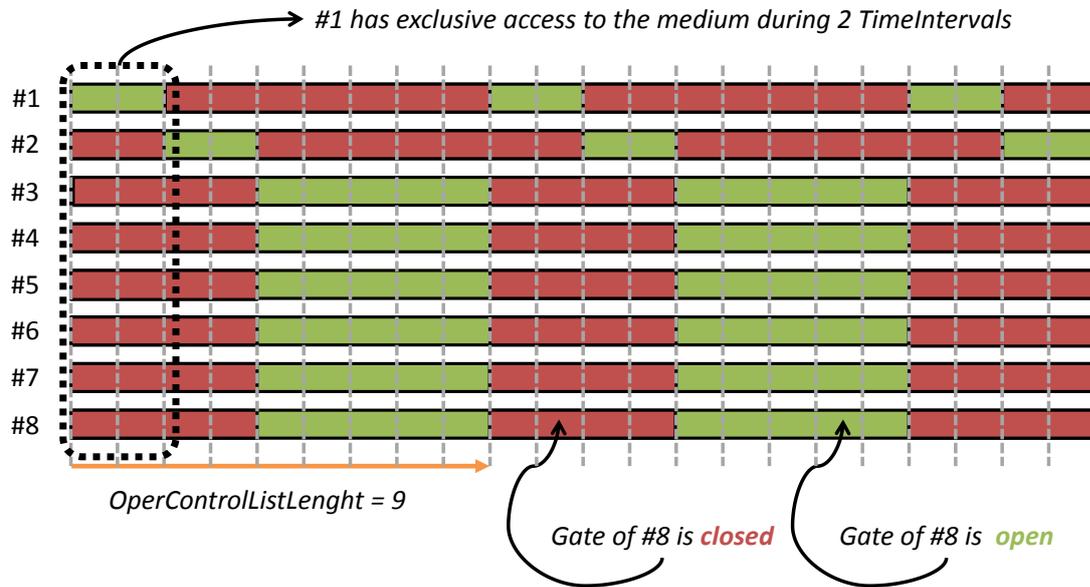


FIGURE 2.8 – GCL d'un port appliquant le principe d'exclusion mutuelle de porte pour 2 files

passante (ex. Credit Based Shaper, Weighted Round Robin, Static Priority, etc.) à travers des TSA, mais aussi des capacités de programmation d'émissions de trames à l'aide des portes de transmissions. Sur la base de ces nouveaux mécanismes, TSN sera capable de garantir des délais de traversée du réseau bornés ainsi qu'une faible gigue sur la date de réception des trames.

Chapitre 3

Première contribution : Sélection de technologies compatibles avec les besoins en qualité de service des réseaux satellite nouvelle génération

Ce chapitre présente le premier problème considéré dans la thèse et résume les principaux éléments de notre première contribution (chapitres 6 et 7 du manuscrit). Dans la première section, nous posons le problème. Dans la deuxième section, nous détaillons la méthodologie utilisée pour la sélection des technologies. Nous listons ensuite les propriétés haut niveau représentatives des exigences du réseau nouvelle génération et les critères qui en découlent. Dans la troisième section, nous présentons les conclusions de la comparaison qualitative et l'identification de trois technologies candidates.

3.1 Problème 1

La première question qui vient naturellement quand on cherche à remplacer le réseau embarqué est la sélection d'une ou plusieurs technologies candidates.

3.1.1 Technologies présélectionnées

Cinq technologies ont été présélectionnées par l'industriel au début de la thèse : Ethernet, ARINC 664, TTEthernet, Time Sensitive Networking et Spacefibre. Elles sont présentées en détail dans le manuscrit dans les chapitres 3 et 4. Toutes ces technologies offrent de nombreux avantages détaillés dans le chapitre 5 du manuscrit.

3.1.2 Propriétés Haut Niveau

De par la diversité des applications qui communiquent à travers le réseau, il arrive que plusieurs messages venant d'applications différentes avec des niveaux d'importance (ou de criticité) différents doivent partager le même équipement ou le même lien. Ces applications ont d'ailleurs probablement des exigences de qualité de service en performance réseau différentes. Le futur réseau devra donc être capable de satisfaire ces exigences. De plus, pour que les applications puissent communiquer correctement, elles ont besoins de partager une notion ou une base de temps commune (en étant par exemple synchronisées). Ainsi, le futur réseau devra être capable de satisfaire cette exigence. Finalement, puisque des ressources seront amenées à être partagées entre applications, le futur réseau devra être capable de satisfaire des exigences de tolérance aux fautes. Dans le manuscrit, nous avons défini 3 propriétés haut niveau : *Mixité des trafics*, *Gestion du temps* et *Tolérance aux fautes*.

Mixité des trafics

Propriété Haut Niveau 1 (Mixité des trafics). *Capacité du système réseau à véhiculer, avec les mêmes ressources, plusieurs flux avec des caractéristiques différentes tout en garantissant que des flux de criticités différentes n'interféreront pas entre eux.*

Par exemple, un réseau compatible avec la propriété Prop. 1 doit être capable de transporter, au sein des mêmes équipements, du trafic (plateforme) faible débit avec exigence de qualité de service (ex. latence, gigue) ainsi que du trafic (charge utile) haut débit sans contraintes particulières.

Gestion du temps

Propriété Haut Niveau 2 (Gestion du temps). *Capacité du système réseau à gérer le temps, soit en proposant une horloge globale commune pour tous les équipements ou bien en proposant une distribution du temps au niveau applicatif.*

Remarque 2. *Bien que l'industriel puisse considérer dans de futures études qu'une partie de la gestion du temps s'effectue au niveau applicatif, dans cette thèse, nous avons supposé que la gestion du temps soit faite au niveau MAC (niveau 2 du modèle OSI).*

Tolérance aux fautes

Avant de définir la troisième propriété, nous devons définir la notion de faute. On distingue plusieurs situations : pertes de messages, messages erronés, messages ne respectant pas leurs contrats de trafic ou bien encore messages arrivant en avance/retard. Il en existe d'autres mais nous ne les avons pas considérées dans l'étude. Une situation de type message erroné pourrait par exemple arriver à un message envoyé par une application à une autre dont le contenu a été altéré (par exemple à cause d'une inversion de bit, phénomène relativement courant dans l'environnement spatial).

Propriété Haut Niveau 3 (Tolérance aux fautes). *Capacité du système de réseau à continuer à fonctionner en présence de fautes, en détectant, en isolant puis en réparant si possible certaines fautes ; ou bien en générant des rapports d'erreurs (qui seront utilisés par des instances de gestions de fautes de plus haut niveau) dans le cas où la faute ne pourrait être réparée localement, le tout de manière transparente vis-à-vis des applications.*

3.1.3 Aperçu de la Contribution

La première étape de sélection est exprimée par le problème suivant :

Définition 3 (Problème 1). *Étant donné la liste des technologies pré-sélectionnées, la liste des propriétés attendues et les résultats de l'état de l'art, est-il possible d'identifier, dans une approche qualitative, une ou plusieurs technologies adéquates pouvant remplacer les réseaux actuels ?*

Afin de répondre à ce problème, nous avons défini un ensemble de critères, par propriété, qui serviraient de points de comparaison. Les critères et les résultats de la comparaison ont été publiés dans [18] et dans [19].

Dans la section suivante, nous détaillons les critères et les résultats de la comparaison.

3.2 Critères pour la comparaison

Afin d'évaluer si une technologie peut remplir ou non une propriété, cette propriété doit être raffinée sous forme de critères plus précis. Dans la littérature, la *Federal Aviation Administration* a proposé une liste de critères qui peuvent être utilisés pour évaluer la pertinence de technologies pour les réseaux embarqués avionique existants et futurs. Les critères que nous présentons ci-dessous en sont grandement inspirés. Ils ont été pensés pour être représentatifs du système satellite.

3.2.1 Critères pour la propriété 1

La propriété 1 *Mixité des trafics* se traduit sous la forme de quatre critères : *Haut Débit*, *Latence bornée*, *Très faible gigue* et *Criticité mixte*.

Critère 1 (Haut Débit). *La technologie considérée est-elle capable de fonctionner à un débit supérieur ou égal à 1Gbit/s ?*

Ce critère est utilisé pour éliminer les technologies trop "lentes" par rapport aux besoins futurs. Par exemple, même si en l'occurrence elles n'ont pas été présélectionnées, 1553 et SpaceWire ne satisfont pas ce critère.

Critère 2 (Latence bornée). *La technologie considérée est-elle capable de garantir des bornes sur les latences d'une ou de plusieurs communications à travers le réseau ?*

Ce critère et le suivant sont utilisés pour éliminer les technologies qui ne seraient pas capable de fournir le niveau de qualité de service en performance réseau recherché pour les futurs satellites.

Critère 3 (Très faible gigue). *La technologie considérée est-elle capable de garantir une gigue en réception inférieure ou égale à 1 μ s pour une ou plusieurs communications à travers le réseau ?*

Critère 4 (Criticité mixte). *La technologie considérée est-elle capable de transporter dans le même lien du trafic provenant de la plateforme (avec des exigences de latences bornées et faibles gigue) et du trafic provenant de la charge utile (haut débit) sans qu'un des trafics n'affecte les performances de l'autre ?*

3.2.2 Critères pour la propriété 2

Trois critères sont identifiés pour la propriété 2 - *Gestion du temps* : *Synchronisation au niveau MAC*, *Robustesse des algorithmes de synchronisation* et *Interfaces avec la couche applicative*.

Critère 5 (Synchronisation au niveau MAC). *La technologie fournit-elle un protocole de synchronisation au niveau MAC ?*

Critère 6 (Robustesse des algorithmes de synchronisation). *L’algorithme de gestion du temps/de synchronisation a-t-il des mécanismes de robustesse ?*

Ce critère anticipe la haute importance de la fonction de synchronisation/distribution du temps dans le satellite. Si toutes les opérations se déroulent sur la base d’une information de temps venant du réseau, le mécanisme en charge de distribuer et maintenir cette information ne doit pas tomber en panne.

Critère 7 (Interfaces avec la couche applicative). *L’algorithme de gestion du temps/de synchronisation est-il associé à une manière standardisée d’interagir vers et depuis les applications ?*

Ce critère est lié au premier critère de la propriété 2 : si le réseau est en charge de la distribution du temps dans le satellite et que la synchronisation a lieu au niveau MAC, cela pourrait être agréable d’avoir une manière standardisée de partager les informations de temps vers et depuis les applications qui s’exécutent au-dessus du réseau. En effet, si l’interface application-réseau varie d’un fournisseur à l’autre, cela engendrerait un surcoût supplémentaire d’adaptation des applications à l’équipement sous-jacent.

3.2.3 Critères pour la propriété 3

Quatre critères sont définis pour la propriété 3 - *Tolérance aux fautes* : *Capacité de détection de fautes*, *Capacité de signalement de fautes*, *Capacité de redondance* et *Capacité de confinement des fautes*.

Critère 8 (Capacité de détection de fautes). *La technologie est-elle capable de détecter des fautes correspondant aux situations suivantes :*

- *Message incorrect ?*
- *Message perdu ?*
- *Message ne respectant pas le contrat de trafic ?*
- *Message arrivant en avance/retard ?*

Ce critère est utilisé pour identifier la capacité des technologies candidates à détecter les messages incorrects (ex. checksums incorrects), les messages perdus, les messages arrivant trop tôt ou trop tard (ex. un flux qui émettrait du trafic en dehors d’une fenêtre temporelle qui pourrait lui être allouée) ou encore les messages ne respectant pas leur contrat de trafic (ex. un flux qui émettrait plus de données qu’il n’y est autorisé par la réservation de ressources).

Critère 9 (Capacité de signalement de fautes). *La technologie est-elle capable de signaler les erreurs qu’elle a précédemment identifiées de manière directe ou indirecte ?*

Le réseau nouvelle génération doit être capable de signaler les erreurs de manière directe avec par exemple des interruptions ou de manière indirecte avec par exemples des statistiques périodiquement consultées par un logiciel dédié à la gestion des erreurs.

Critère 10 (Capacité de redondance). *La technologie dispose-t-elle d’un mécanisme pour la redondance ?*

La redondance a en effet été identifiée par l’industriel comme une manière de répondre à la tolérance à la perte d’un message. C’est pourquoi la technologie du futur réseau doit fournir un tel mécanisme.

Critère 11 (Capacité de confinement des fautes). *La technologie est-elle capable d’isoler/de contenir voire même d’éliminer les erreurs qu’elle a détectées ?*

Ce comportement est classique dans un réseau temps réel industriel. Ce faisant, l'objectif est d'éviter que l'erreur ne se propage i.e. qu'elle perturbe le fonctionnement nominal d'autres systèmes utilisant le réseau. Par exemple, un commutateur connecté à un équipement dit *babbling-idiot* (c'est à dire un équipement qui émet en permanence des messages, en dehors de son contrat de trafic) doit être capable d'éliminer le surplus de messages afin qu'ils ne se propagent pas dans le réseau et provoquent d'éventuels débordements de files.

Nous allons maintenant évaluer la pertinence de chaque technologie présélectionnée puis les comparer entre elles sur la base de l'analyse de chacun des critères précédemment définis.

3.3 Résultats et Analyse

Pour réaliser la comparaison, nous avons évalué la compatibilité de chaque technologie avec chaque critère.

Remarque 3. *Un point important à mentionner est que cette comparaison et les critères qui lui sont associés évaluent ce qu'il est possible de réaliser en utilisant la technologie concernée au niveau MAC et non ce qu'il serait possible de faire à travers une implémentation ou un usage astucieux de la technologie couplé à d'autres éléments dans le système.*

Dans ce résumé, nous ne détaillons pas le processus de décision de la compatibilité de chaque technologie pour chaque critère. Ces éléments sont disponibles dans le chapitre 7 du manuscrit. Ceci étant, nous proposons ci-après trois tableaux récapitulatifs des résultats de la comparaison.

TABLE 3.1 – Compatibilité avec la propriété 1 - Mixité des trafics

Critère	Ethernet	ARINC 664	TTEthernet	TSN	SpF
Haut Débit	✓	✓	✓	✓	✓
Latence bornée	✗	✓	✓	✓	✓
Très faible gigue	✗	✗	✓	✓	✓
Compatibilité	✗	✗	✓	✓	✓

TABLE 3.2 – Compatibilité avec la propriété 2 - Gestion du temps

Critère	Ethernet	ARINC 664	TTEthernet	TSN	SpF
Synchronisation au niveau MAC	✗	✗	✓	✓	✓
Robustesse des algorithmes de synchronisation	✗	✗	✓	✓	✗
Interfaces avec la couche applicative	✗	✗	✗	✓	✓
Compatibilité	✗	✗	✓	✓	✓

L'analyse des résultats de la comparaison résumée dans les trois tableaux ci-dessus semble indiquer que trois technologies, Spacefibre, TTEthernet et Time Sensitive Networking, pourraient être capables, d'un point de vue qualitatif, de répondre aux exigences des futures missions. Les deux autres technologies, Ethernet et ARINC 664, ont été écartées car, entre autres, elles ne permettraient pas d'atteindre les niveaux de qualité de service recherchés. Nous souhaitons profiter de ce paragraphe pour détailler d'autres arguments non liés à la qualité de service ou au fonctionnement du réseau, qu'il est nécessaire de prendre en compte dans le processus de sélection.

TABLE 3.3 – Compatibilité avec la propriété 3 - Tolérance aux fautes

Critère	Ethernet	ARINC 664	TTEthernet	TSN	SpF
Capacité de détection de fautes	✘	✘	✓	✓	✓
Capacité de signalement de fautes	✓	✓	✓	✓	✓
Capacité de redondance	✘	✓	✓	✓	✓
Capacité de confinement des fautes	✘	✘	✓	✓	✓
Compatibilité	✘	✘	✓	✓	✓

Premièrement, les composants embarqués dans les satellites ont de fortes contraintes tant au niveau matériel que logiciel. Ces contraintes, contrairement au domaine aéronautique, ne sont pas liées à un processus de certification mais plutôt à une capacité à fonctionner dans un environnement avec des températures, des champs électromagnétiques et des niveaux de radiations bien particuliers. Cela signifie que l'industriel en charge du réseau des satellites doit soit acheter des stations terminales et des commutateurs conçus pour cet environnement spatial spécifique (i.e. spatialisables) ou bien qu'il investisse dans des logiciels (appelées IP - *Intellectual Property*) qui seraient instanciés dans des composants spatialisables dont il dispose. TTEthernet (via TTTech) et Spacefibre ont déjà été, ou seront bientôt, utilisés dans diverses missions dans l'espace en Europe comme aux Etats Unis. Leur usage dans un contexte spatial est standardisé par l'Agence Spatiale Européenne (ESA) dans ce qui s'intitule un standard ECSS. Il serait donc possible de trouver sur le marché des composants spatialisables supportant le TTEthernet ou le Spacefibre. Concernant TSN, les composants disponibles actuellement ne sont spatialisables mais la technologie gagne tellement d'intérêt en particulier dans l'industrie automobile et dans le secteur de l'automatisation industrielle, qu'il serait tout à fait envisageable d'acheter des IP TSN qui seraient ensuite instanciées dans du matériel spatialisable de l'industriel.

Deuxièmement, les industriels du secteur spatial espèrent que l'utilisation de composants sur étagère (dit *COTS*) provenant d'une technologie largement répandue dans d'autres secteurs d'industrie pourrait participer à la réduction du coût global du satellite en profitant des coûts de conception, d'achat d'équipements et de développement de logiciel réduits. Dans ce sens, le fait que TTEthernet et Spacefibre ne soient produits et maintenus que par un nombre très limité de fournisseurs serait un frein à leur usage pour le futur réseau embarqué. A l'inverse, le nombre grandissant de fournisseurs de composants TSN pourrait attirer les industriels du secteur vers cette technologie. Cependant, les produits TSN actuellement disponibles sur le marché pourraient ne pas être compatibles avec l'environnement spatial ou les exigences de qualité de service. Cela signifierait en outre qu'un effort serait nécessaire pour adapter ces produits au satellite, ce qui risquerait d'augmenter les coûts. Néanmoins, la réduction attendue des coûts non-récurrents liée à l'utilisation du TSN pourrait tout de même attirer certains industriels. C'est pourquoi définir un profil spatial (comme celui défini pour l'industrie automobile) serait un bon point de départ pour donner une identité au secteur aérospatial vis-à-vis des fournisseurs de composants TSN.

Troisièmement, concernant les aspects validation et certification, il y a un certain avantage à utiliser TTEthernet ou bien Time Sensitive Networking plutôt que Spacefibre. En effet, TTEthernet et Time Sensitive Networking sont tous deux basés sur Ethernet, pour qui de très nombreuses recherches sur la validation/certification sont disponibles dans l'état de l'art. De plus, puisque ces deux technologies suscitent l'intérêt des industriels de divers secteurs, elles ont aussi suscité l'intérêt de plus d'acteurs du monde académique que Spacefibre. Il y a déjà des outils disponibles pour simuler, configurer et valider des réseaux TTEthernet et les outils similaires pour TSN sont en train de gagner en maturité. A l'inverse, SpaceFibre a été modélisé dans le simulateur OMNET++ mais de

plus amples analyses sont nécessaires pour valider les performances temps réelles proposées par sa politique d'accès au médium.

Il n'y a donc pas de grand gagnant parmi ces trois technologies finalistes mais plutôt des compromis à préciser et à discuter pour choisir quelle(s) technologie(s) sera(ont) utilisée(s) dans le futur.

Conclusion sur la première contribution

Pour conclure sur cette première contribution, la comparaison qualitative, basée sur un ensemble de critères représentatifs des propriétés attendues, nous a amenés à identifier trois candidats pertinents pour remplacer les réseaux embarqués actuels. En dehors des aspects performances du réseau, de nombreuses considérations doivent être faites quant à l'usage de ces technologies dans l'espace. Des études plus approfondies sont nécessaires afin de décider sur quelle(s) technologie(s) le réseau "unifié" nouvelle génération reposerait. Nous avons proposé dans le paragraphe précédent un aperçu des avantages et des inconvénients de chaque technologie et de certains compromis à prendre en compte dans le choix. La route est encore longue avant qu'une décision ne soit prise sur le futur réseau embarqué. Une des pistes les plus plausibles est que, non pas une mais plusieurs technologies seront sélectionnées, chacune adaptée à un type spécifique de mission. Par exemple, Time Sensitive Networking pourrait être utilisé pour des constellations de satellites dans un contexte *New Space* où le nombre de composants pour la constellation entière est relativement élevé. L'industriel pourrait ainsi bénéficier du marché TSN existant. TTEthernet ou Spacefibre pourraient, quant à eux, être utilisés pour des missions spécifiques mono-satellite où des composants conçus pour l'environnement spatial sont vraiment nécessaires.

Puisque Time Sensitive Networking était totalement nouveau dans l'industrie aérospatiale, nous continuons notre analyse de la compatibilité de TSN en creusant davantage la capacité de la technologie à répondre aux besoins.

Chapitre 4

Seconde contribution : Génération de configuration d'un réseau TSN pour des satellites nouvelle génération

Ce chapitre présente les principaux éléments de notre seconde contribution. Il résume les chapitres 9, 10 et 11 du manuscrit. Dans la première section, nous posons le second problème. Puis, nous présentons dans la section suivante notre méthodologie novatrice pour la génération de configuration réseau. Nous appliquons, dans la troisième section, cette méthodologie aux réseaux TSN dans deux "implémentations" (comprendre, de deux manière différentes) : Exclusive Queue Allocation et Size Based Isolation.

4.1 Problème 2

TSN est composé d'une vingtaine de standards et ce nombre est encore en train de grossir. Monter en compétence et développer une expertise sur cette technologie et tous les mécanismes qu'elle propose afin de les exploiter à leur plein potentiel est une activité ardue. Néanmoins, tous les standards (et leurs mécanismes) ne sont peut-être pas nécessaires pour satisfaire les exigences des futures missions. C'est pourquoi l'objectif de ce deuxième problème est de réduire l'effort d'expertise de la technologie et de conception du réseau, d'une part en identifiant un sous-ensemble de standards suffisant pour répondre aux besoins de l'industriel et d'autre part en proposant une méthode de configuration automatique du sous-ensemble de standards prenant en compte ces besoins.

Pour traiter ce deuxième problème, nous proposons une modélisation formelle du réseau satellite (notamment des flux qui le traversent). De même, nous précisons puis formalisons les exigences en qualité de service qui pourraient être rencontrées dans différents systèmes spatiaux. Enfin, une fois que le modèle et les exigences ont été proprement définis, nous présentons à la fin de cette section le problème et un aperçu de notre seconde contribution.

4.1.1 Modélisation des applications

Dans le satellite, des applications communiquent entre elles à travers le réseau. Ces échanges se matérialisent par des flux de messages. Avant de modéliser ces flux dans le réseau, nous introduisons ci-après le comportement de ces applications. Pour cela, nous définissons deux grandeurs P_{MIF} et P_{MAF} .

Définition 4 (P_{MIF} cycle MIF). *Les exécutions d'applications à bord (ex. Navigation Basée Vision, Contrôle d'Attitude et d'Orbite, Traitement des données des charges utiles, etc.) suivent un pattern cyclique. Ce cycle, intitulé cycle MIF ou période MIF, est de durée P_{MIF} , constante par satellite.*

Définition 5 (P_{MAF} cycle MAF). *La période MIF représente la durée d'un cycle applicatif. Puisque plusieurs applications, avec des périodes potentiellement différentes, coexistent dans le satellite, on peut définir une période du système correspondant à l'hyper-période de toutes les applications. Cette période dite période MAF ou cycle MAF est de durée P_{MAF} . Dans un cycle MAF, il y a k cycles MIF si bien que $P_{MAF} = k * P_{MIF}$, $k \in \mathbb{N}^*$.*

Nous illustrons les cycles MIF et MAF dans la figure Fig. 4.1.

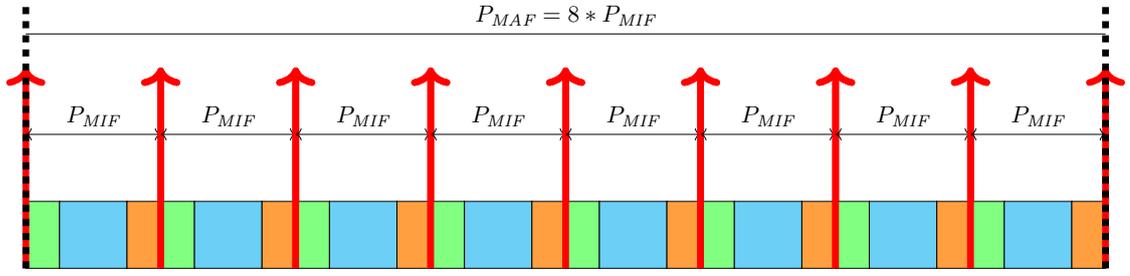


FIGURE 4.1 – Lien P_{MIF} - P_{MAF} avec $k = 8$

Remarque 4. *Dans les satellites de l'industriel, une valeur usuelle pour P_{MAF} était 1000ms. A partir de cette valeur, nous listons des valeurs usuelles de k et les durées P_{MIF} associées dans le tableau Tab. 4.1.*

k	P_{MIF}
8	125ms
16	62.5ms
32	31.25ms

TABLE 4.1 – Valeurs usuelles de k et P_{MIF}

4.1.2 Modélisation des flux

Définitions

Les applications que nous venons d'introduire dans le paragraphe précédent communiquent entre elles à l'aide de messages envoyés dans le réseau. Ces messages sont regroupés en *flux*.

Définition 6 (Flux, f , \mathbb{F}). *Un flux f est une séquence unidirectionnelle de messages d'une source vers une destination. L'ensemble des flux du système est dénoté \mathbb{F} . Chaque flux est caractérisé par le tuple suivant :*

$$\forall f \in \mathbb{F} < Src_f, Dest_f, Size_f, r_f > \quad (4.1)$$

où :

- $Src_f \in \mathcal{D}$ est la station terminale où le message est généré puis émis,
- $Dest_f$ est le destinataire du message,
- $Size_f$ est la taille constante d'un message,
- r_f est notre manière d'exprimer la période du flux.

Hypothèse 1. *Dans cette étude, nous considérons que les messages applicatifs ne dépassent pas la taille maximale d'une trame Ethernet. On pourra donc utiliser indifféremment les mots messages ou trames pour décrire à la fois le message applicatif et la trame dans le réseau.*

Hypothèse 2. *Dans cette étude, nous n'avons pas modélisé de flux multicasts, il n'y a donc qu'un destinataire par flux.*

La période P_f d'un flux f est reliée à son ratio r_f par l'équation suivante :

$$\forall f \in \mathbb{F}, \begin{cases} r_f \leq -1 & \implies P_f = |r_f| * P_{MIF} \\ r_f > 1 & \implies r_f \text{ messages par } P_{MIF} \iff P_f \text{ est la période pendant laquelle } r_f \text{ messages sont envoyés} \\ r_f = 0 & \implies P_f = NA \text{ (} f \text{ est non périodique)} \end{cases} \quad (4.2)$$

Dans la thèse, nous nous sommes limités à des flux à ratio négatif. Ainsi, les flux considérés sont périodiques à l'échelle des applications c'est-à-dire que exactement un message est produit à chaque occurrence de l'application qui lui est associée. Cependant, les émissions de messages ne sont pas strictement périodiques i.e. espacées exactement d'une période du flux. Au contraire, les instants d'émissions peuvent varier du moment qu'ils respectent la période de l'application. Nous avons représenté dans la figure Fig. 4.2, le concept de ratio pour deux flux, en vert avec $r_f = 2$ et en bleu avec $r_f = -4$. Les flèches vertes et bleues représentent les instants d'émissions de messages dans le réseau.

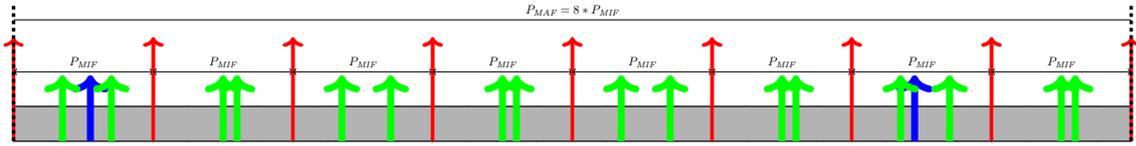


FIGURE 4.2 – Concept de ratio pour deux flux ($r_{f1} = 2$, $r_{f2} = -4$) and $k = 8$

Définition 7 (f_l , τ_{f_l}). *Un flux est une séquence de messages (donc de trames). Nous notons f_l , $l \in \mathbb{N}$, le l -ième message de f et τ_{f_l} la durée d'émission de ce message f_l . Puisque la taille de message est constante par flux, la durée d'émission des messages est elle aussi constante par flux.*

Restriction du modèle au niveau flux

Dans le modèle complet publié dans [16], les spécifications et les contraintes du système étaient exprimées au niveau des applications. Dans le manuscrit, nous détaillons la manière dont une application émet et reçoit des messages puis nous proposons une restriction de notre modèle au niveau des flux afin de le simplifier dans une première approche.

4.1.3 Exigences de qualité de service

Dans les prochains paragraphes, nous proposons un aperçu des type d'exigences en qualité de service existant dans les systèmes de l'industriel. Ils sont dérivés des propriétés 1, 2 et 3 de la première contribution.

Choix 1. *Dans notre étude, nous nous sommes limités aux exigences de la propriété 1 et à une exigence issue de la propriété 3. En effet, la priorité était d'abord de s'assurer qu'il était possible de générer des configurations répondant aux besoins en performances avant de s'intéresser à la tolérance aux fautes.*

Date de référence

Avant de présenter les exigences en qualité de service, nous devons définir la notion de date de référence.

Définition 8 ($Ref(f_l)$). *La date de référence d'un message f_l est définie par la formule suivante : $Ref(f_l) = l \times P_f$. Elle correspond au début de la l -ième période applicative i.e. la période dans laquelle f_l doit être émis. Il devra être émis au plus tard avant la date de référence du message qui le suit (ici f_{l+1}).*

Nous illustrons ce concept avec la figure Fig. 4.3.

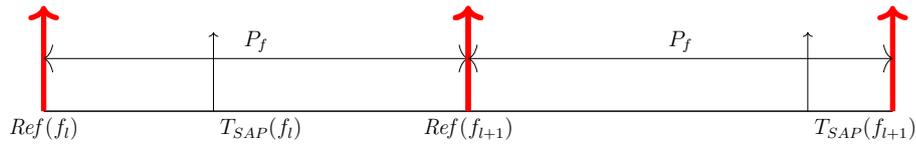


FIGURE 4.3 – Dates de référence de deux messages f_l et f_{l+1} avec $r_- = -2$

Remarque 5. *Cette définition implique que, contrairement à la majorité des travaux existant dans l'état de l'art, les flux du modèle ne sont pas strictement périodiques. La seule contrainte étant que l'émission du message f_l soit faite dans la fenêtre $[Ref(f_l), Ref(f_{l+1})]$ et non exactement P_f après l'émission de f_{l-1} .*

Exigences de type Mixité des trafics

Dans ce paragraphe, nous proposons un aperçu des contraintes de qualité de service en performance identifiées pour les flux. Nous restons volontairement haut niveau dans la formalisation pour ce résumé. Les équations détaillées du modèle sont disponibles dans le manuscrit.

Exigence de Performance 1 (Deadline). *Soit un flux $f \in \mathbb{F}$, ce flux a obligatoirement une contrainte de deadline. Cette contrainte implique que la réception des messages du flux f doit avoir lieu avant une certaine date à chaque période de f .*

En particulier, cela signifie que, au plus tard, le message f_l doit être reçu avant le début de la fenêtre d'émission du message f_{l+1} . Dans le cas où la deadline est égale à la période du flux, le flux est dit "à échéance sur requête". Nous avons représenté dans la figure Fig. 4.4 des exemples de contraintes de deadlines pour deux flux à échéance sur requête.

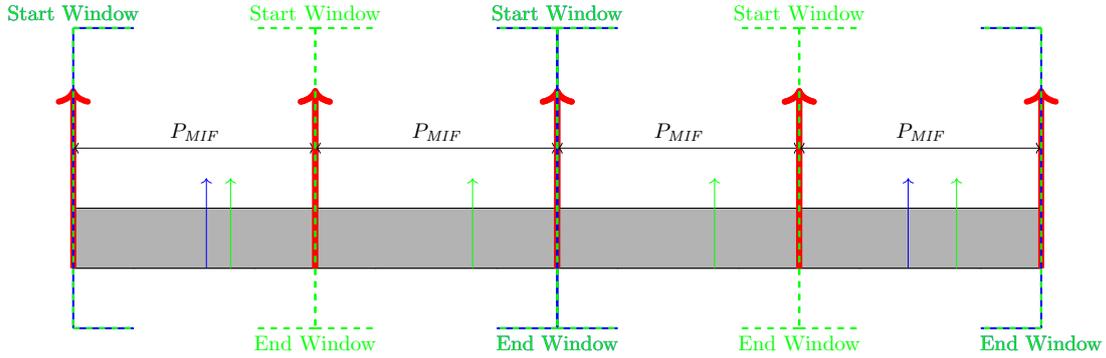


FIGURE 4.4 – Deadlines pour deux flux ($r_{f1} = 2 * P_{MIF}$, $r_{f2} = 4 * P_{MIF}$) et $k = 8$

Définition 9 (Gigue en réception). La gigue en réception ou plus simplement gigue entre deux trames f_l et f_m est définie comme la variabilité des instants auxquels ils sont reçus. Nous noterons $Jit_{f_l, m}$ la gigue entre les messages f_l et f_m du flux f . La gigue du flux sera notée Jit_f telle que $\forall f \in \mathbb{F}, Jit_f = \max_{l, m} Jit_{f_l, m}$.

Remarque 6. La latence d'un message peut être définie dans ce modèle comme la durée entre la date de référence de ce message et l'instant auquel il est reçu. Alors, une définition alternative mais équivalente de la gigue serait la variabilité de la latence entre plusieurs trames d'un même flux.

La gigue du message f_m par rapport au message f_l est illustrée dans la figure Fig. 4.5.

Exigence de Performance 2 (Jitter). Un flux f peut avoir une exigence de gigue, définie par $f.jitter \in \mathbb{N} \cup \{NA\}$. NA signifie qu'il n'y a pas de contrainte, autrement, $f.jitter$ est la valeur maximale acceptée pour la gigue pour le flux f .

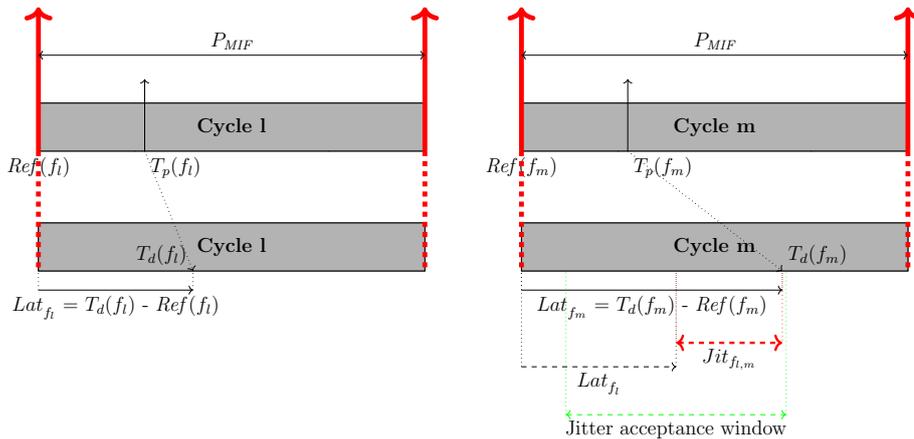


FIGURE 4.5 – Gigue entre les messages f_l et f_m

Définition 10 (Flux avec gigue / sans gigue, \mathbb{F}_j). Nous appelons les flux ayant des exigences de gigue, des flux avec gigue et ceux n'en n'ayant pas des flux sans gigue. Nous notons $\mathbb{F}_j = \{f \in \mathbb{F} \mid f.jitter \neq \text{NA}\}$ l'ensemble des flux avec gigue.

Exigences de type Tolérance aux fautes

En plus des contraintes de performance, nous avons considéré une exigence de tolérance aux fautes concernant la configuration en elle-même.

Définition 11 (Insensibilité à la perte d'un message). Un système est dit "insensible à la perte d'un message" si pour tout flux f , la perte d'un message de f n'a pas d'impact négatif sur les performances (deadline/gigue) des autres flux.

Exigence de Tolérance aux Fautes 1. Toute configuration du réseau satellite doit être insensible à la perte d'un message.

4.1.4 Considérations Industrielles

Contrat de Production et Date de Libération

Dans les systèmes de l'industriel, les applications sont soumises à des contrats de production. Cela signifie que chaque flux devra émettre ses messages dans une fenêtre temporelle, définie par contrat (voir Fig. 4.6). Si les émissions de messages ont bien lieu dans cette fenêtre, les exigences de qualité de services et de tolérance aux fautes de ces messages seront automatiquement satisfaites. Ces contrats sont discutés entre les architectes du satellite pendant les phases de design. Nous considérons dans notre modèle que les applications respectent toujours leurs contrats et que de fait le réseau assure le respect des exigences de qualité de service des applications.

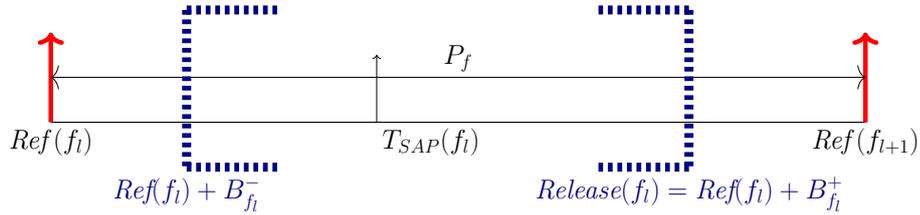


FIGURE 4.6 – $Ref(f_i)$, $T_{SAP}(f_i)$ and $Prod(f_i)$

Définition 12 (Contrat de production). Soit f_i le i -ième message de f . Le contrat de production associé à f_i est l'intervalle $Prod(f_i) = [Ref(f_i) + B_{f_i}^-, Ref(f_i) + B_{f_i}^+] \subseteq [Ref(f_i), Ref(f_{i+1})[$, ou $B_{f_i}^-$ (resp. $B_{f_i}^+$) est l'instant de production au plus tôt (resp. au plus tard). La borne supérieure de $Prod(f_i)$ est appelée date de libération. Nous la notons $Release(f_i) = Ref(f_i) + B_{f_i}^+$. Le contrat de trafic d'un flux f , noté $Prod(f)$, est défini comme suit : $Prod(f) = \cup_{i \in \mathbb{N}} Prod(f_i)$.

Schémas d'émission

Dans la version courante du système satellite, les communications sur le réseau plateforme sont réalisées la plupart du temps sur un bus 1553. Les émissions des applications sont gérées à l'aide d'accès programmés pré-calculés à une liste de descripteurs qui "prépareront" la trame avant son émission et non à l'aide de file d'attentes comme ce serait le cas pour un réseau Ethernet par

exemple. Grâce à ce schéma d'émission, les applications peuvent déposer leurs messages quand elles le souhaitent. Il n'y a pas de contraintes particulières du moment que le message est déposé avant la lecture du descripteur.

Les logiciels applicatifs exécutés à bord du satellite s'appuient sur ce schéma d'émission. Ainsi, changer ce schéma aurait un impact significatif sur le volume de code qui devrait être redéveloppé. De fait, les configurations du réseau TSN que nous allons générer doivent minimiser l'effort de redéveloppement et de manière générale l'impact sur les applications.

L'idée est la suivante : en maximisant la fenêtre pendant laquelle une application peut émettre ses messages, nous nous rapprochons au plus près du schéma d'émission du 1553. Le réseau sera alors en charge de transporter convenablement le message pour qu'il atteigne ses exigences de qualité de service. Pour ce faire, nous proposons la contrainte suivante sur les configurations :

Exigence sur l'Effort de Développement 1. *Toute configuration doit maximiser la longueur de $Prod(f)$ (voir Def. 12). En particulier, les configurations où $B_{f_i}^- = 0$ permettent aux applications de commencer à exécuter des tâches au début de la période MIF et la maximisation de $B_{f_i}^+$ (idéalement $B_{f_i}^+ = P_f$) laisse plus de temps disponible pour exécuter des tâches durant le cycle MIF.*

Discussions sur le coût de la modernisation du réseau

En plus de l'impact applicatif, la modernisation du réseau a aussi un coût important sur les composants qui constituent le réseau. En effet, passer de l'architecture 1553+SpaceWire à une architecture Ethernet/TSN a deux impacts notables. Premièrement, le remplacement de tous les composants actuels en composants supportant le TSN représente un coût considérable que l'industriel n'est pas prêt à accepter. Deuxièmement, nous l'avons déjà dit, TSN est composé de plus de vingt standards. Plus il y aura de standards implémentés dans les composants, plus l'analyse de la défaillance de ces composants (et du logiciel associé) sera complexe.

Dans l'architecture actuelle, les stations terminales qui reçoivent des messages (par exemple des capteurs) sont extrêmement simples et c'est l'ordinateur central du satellite (OBC - On-Board Computer) qui concentre toute l'intelligence. De fait, les configurations du réseau se doivent de tenter de minimiser le coût de la modernisation en limitant l'utilisation des mécanismes TSN et le nombre d'équipements implémentant ces mécanismes. Une autre solution aurait peut-être été de remettre en cause la simplicité des receveurs et de distribuer l'intelligence à travers le réseau, mais cela n'a pas été réalisé pendant cette thèse.

4.1.5 Aperçu de la contribution

Afin de démontrer la compatibilité de Time Sensitive Networking avec les exigences de l'industrie aérospatiale, nous proposons de générer des configurations du réseau qui satisferaient ces exigences. Nous nous sommes d'abord demandés :

Définition 13 (Problème 2 - Étape 1). *Sachant les exigences de qualité de services des futures missions, quel est le plus petit sous-ensemble de standards TSN nécessaire pour satisfaire ces exigences ?*

En effet, la configuration des standards TSN n'est pas aisée, de par leur nombre et leur complexité. Afin de réduire l'effort de l'architecte réseau dans sa conception d'un réseau TSN pour les futures générations de satellites, nous trouvons pertinents de réduire au maximum la liste des standards à utiliser.

Choix 2 (Choix des standards TSN). *Nous avons présenté un bref aperçu d'un sous-ensemble, que nous pourrions appeler Profil dans le vocabulaire IEEE, dans [14] puis dans [17]. Ce profil*

est toujours en train d'être consolidé dans un groupe de travail commun entre l'IEEE et la SAE (voir IEEE/SAE P802.1DP TSN Profile for Aerospace). Ce profil contient un certain nombre de standards mais dans cette étude nous l'avons réduit au standard 802.1Qbv uniquement afin de traiter les exigences de qualité de service du réseau nouvelle génération.

Ce choix est motivé par trois raisons. D'abord, il n'existait parmi les standards TSN, au moment d'écrire ce manuscrit, que deux standards capables d'assurer une faible gigue en réception : 802.1Qbv et 802.1Qcr. Ensuite, la majorité des travaux de l'état de l'art étaient basés sur 802.1Qbv pour la gestion de trafic avec gigue. Enfin, à l'inverse de 802.1Qbv, il n'y avait aucun composant implémentant le standard 802.1Qcr sur le marché.

Une fois le sous-ensemble de standard choisi, nous nous sommes demandés :

Définition 14 (Problème 2 - Étape 2). *Sachant les exigences de qualité de service à bord et sachant les méthodologies existantes dans l'état de l'art, est-il possible de générer des configurations valides du standard 802.1Qbv ?*

Définition 15 (Configuration valide). *Une configuration est dite "valide" si elle permet de satisfaire toutes les exigences de qualité de service de tous les flux.*

Il est en fait relativement facile, à l'aide des méthodologies dans l'état de l'art, de générer des configurations valides. Cependant, nous souhaitons générer des configurations valides qui soient plus acceptables sur le plan industriel, c'est-à-dire, avec un impact/coût modéré sur les composants, les logiciels et le processus d'intégration du satellite.

Afin de traiter ce second problème, nous proposons de mettre en place une manière automatisée de générer des configurations valides. En se basant sur les méthodologies de l'état de l'art, nous allons générer des configurations à l'aide de la programmation par contraintes. Dans le manuscrit, nous présentons dans le chapitre 9 les méthodologies de l'état de l'art, un modèle de configuration du standard 802.1Qbv et des cas d'usages industriels sur lesquels nous allons appliquer notre processus de configuration automatique. Nous ne les détaillerons pas dans ce résumé. Dans la prochaine section, nous introduisons le concept novateur de configurations dites "Egress TT". Nous discutons brièvement de ses avantages et ses inconvénients puis nous présentons deux adaptations (ou implémentations) de ce concept pour des réseaux TSN.

4.2 Egress TT, une nouvelle approche pour la génération de configuration de réseaux à faible gigue

Dans le chapitre 9 du manuscrit, nous avons présenté End-to-End TT, l'approche de configuration la plus répandue dans l'état de l'art. Elle consiste en la génération de configuration à l'aide de la programmation par contrainte. Les configurations consistent en un ordonnancement statique des émissions de toutes les trames sur tous les équipements du réseau. Cet ordonnancement est ensuite converti en une configuration du mécanisme Time Aware Shaper de TSN. Dans la section 9.3 dans le manuscrit, nous avons identifié les limitations en termes d'impact sur les applications, de passage à l'échelle et de coût de mise à niveau de l'approche End-to-End TT vis à vis du problème que nous considérons. De fait, nous présentons dans cette section une nouvelle approche de configuration intitulée Egress TT où un ordonnancement des émissions n'est calculé que sur le dernier commutateur du réseau dans le chemin de n'importe quel message. Les configurations de l'approche Egress TT, dites configurations Egress TT, sont générées à l'aide de la programmation par contrainte à l'image de l'état de l'art. L'approche Egress TT a été conçue afin de fournir une méthodologie plus compatible/plus industrialisable que celles existantes dans de l'état de l'art vis à vis du système satellite..

4.2.1 Qu'est-ce qu'Egress TT ?

Nous définissons ce qu'est une configuration Egress TT par comparaison à une configuration End-to-End TT. Dans une configuration End-to-End TT, la faible gigue est obtenue à travers un ordonnancement des émissions de toutes les trames dans tous les équipements du réseau. Cela signifie que les dates d'émission mais aussi de réceptions de n'importe quelle trame de n'importe quel flux dans n'importe quel port est fixe, et connue a priori. Ce faisant, les latences et les giges de tous les flux sont maîtrisées. Nous illustrons une configuration End-to-End TT avec un émetteur, un receveur et deux commutateurs (SWA et SWB) dans la figure Fig. 4.7.

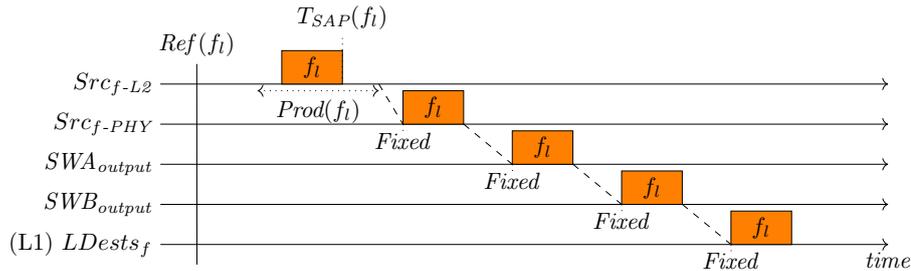


FIGURE 4.7 – Configuration End-to-End TT

Un problème majeur avec ce type de configuration est le coût en temps et en ressources informatiques nécessaires pour déterminer les paramètres d'une configuration valide. En effet, un grand nombre de date d'émission doit être calculé pour chaque configuration. Dans ce contexte, nous proposons Egress TT, un nouveau type de configuration où l'idée est la suivante : au lieu d'ordonnancer toutes les émissions sur tous les équipements, nous proposons plutôt de programmer uniquement les émissions des trames sur le dernier commutateur dans le chemin de n'importe quel message.

Définition 16 (Configurations Egress TT). *Dans une configuration Egress TT, les dates d'émission des flux avec gigue sont pré-calculées dans le dernier commutateur dans leur chemin. Dans tous les autres équipements, la stratégie d'accès au médium n'est pas spécifiée. Un architecte réseau pourra ainsi choisir la stratégie d'accès qu'il souhaite, à partir du moment où une borne sur la latence des trames pourra être calculée.*

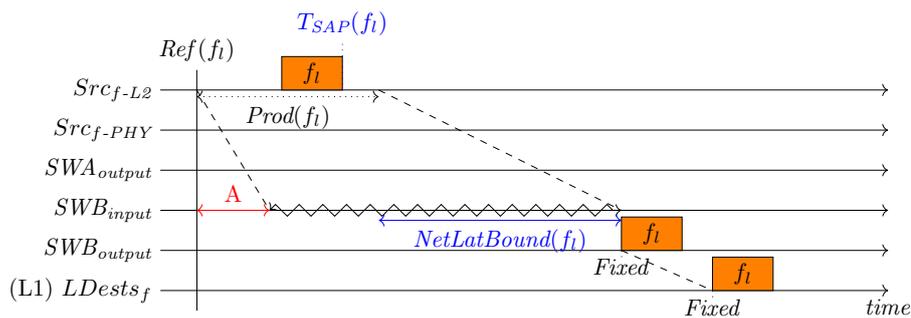


FIGURE 4.8 – Configuration Egress TT

Nous illustrons dans la figure Fig. 4.8 une configuration Egress TT avec un émetteur, un receveur et deux commutateurs (SWA et SWB).

Remarque 7. Nous présenterons dans la prochaine section une application de cette approche à des réseaux basés Ethernet et TSN mais elle peut, en principe, être appliquée à n'importe quelle technologie capable de véhiculer des trames de manière dirigée par le temps.

Remarque 8. Cette approche Egress TT (appelée LETT dans [8]) est inspirée des approches LET (pour Logical Execution Time) où des applications peuvent absorber la variabilité du délai de communication (et donc des dates de réceptions) en bufférisant les messages avant de les délivrer à leurs destinataires à des instants fixes prédéfinis. Néanmoins, Egress TT se distingue par le fait qu'elle est adaptée aux systèmes dans lesquels l'architecte ne souhaite pas avoir à gérer la gigue des trames dans l'équipement de destination. Par exemple, dans le système de l'industriel, les équipements receveurs n'ont pas ce genre de capacité pour le moment. Afin d'éviter de devoir concevoir de nouveau ces équipements, Egress TT propose de déléguer au réseau (dont les équipements et les logiciels seront obligatoirement remplacés) cette gestion des giges.

4.2.2 Comment fonctionne Egress TT ?

Nous décrivons ci-après le principe de fonctionnement des configurations Egress TT en étudiant comment un message f_l d'un flux f avec gigue est géré dans ces configurations.

Par définition, le message f_l doit être émis dans une fenêtre définie par son contrat de trafic et l'émission peut avoir lieu à n'importe quel instant durant cet intervalle de temps. Par ailleurs, nous supposons que le délai de traversée du réseau pour le message f_l peut être borné et nommons $NetLatBound(f_l)$ une borne. Nous illustrons cette situation dans la figure Fig. 4.8 (A représente le délai de traversée le plus favorable). Dans la situation la plus défavorable, le message f_l arrivera dans le dernier commutateur dans son trajet après $NetLatBound(f_l)$.

Définition 17 ($NetLatBound$). Une borne supérieure sur le délai de traversée du réseau est dénotée $\forall f \in \mathbb{F}, \forall l \in \mathbb{N}, NetLatBound(f_l)$.

En pratique, il n'y a pas d'exigences particulières sur la politique d'accès au médium avant le dernier commutateur sur le chemin de f à partir du moment où une borne $NetLatBound$ peut être calculée. Cela implique notamment que nous autorisons un message à être retardé par un conflit d'accès au médium (par exemple avec Static Priority).

Le commutateur du dernier saut est en charge d'absorber la variabilité du délai (c'est-à-dire la gigue) et de fournir le message à l'application en respect de ses contraintes de gigue. En particulier, si le message est arrivé en avance, il sera bufférisé afin d'être émis à une date fixe. Pour s'assurer que le message sera reçu en respectant les contraintes de gigue, il suffit que :

- le message soit reçu dans le dernier commutateur avant de devoir être envoyé (à la date fixe pré-calculée)
- le message ne soit pas envoyé du dernier commutateur vers le destinataire avant sa date d'émission prévue.

De cette situation, indépendamment de quand le message est émis par l'application, il sera envoyé à son destinataire à une date fixe, ce qui permet de satisfaire les exigences de gigue.

4.2.3 Quel est l'intérêt de Egress TT ?

L'approche Egress TT est intéressante pour plusieurs raisons :

Coût en temps et en ressources. Tout d'abord, elle limite le nombre de date d'émission à calculer pour chaque nouvelle configuration. Ce résultat est illustré dans la section 4.3.4 des expérimentations.

Impact sur les applications. De plus, ce concept de configurations Egress TT où les émissions de messages sont programmées uniquement dans les derniers sauts offre plus de flexibilité pour les contrats de production. Ces contrats sont plus grands que ceux offerts par End-to-End TT, seule solution à l'heure actuelle à permettre une faible gigue. En ce sens, Egress TT répond à nos critères.

Transition vers des réseaux nouvelle génération. Enfin, les configurations Egress TT pourraient faciliter la transition des architectures de réseaux embarqués actuelles vers de nouvelles technologies. En effet, seul le dernier commutateur dans le chemin de n'importe quel flux nécessite véritablement d'être remplacé. De fait, si le cœur de réseau existant est déjà compatible avec les autres exigences du système considéré (notamment la bande passante), seuls quelques équipements auraient à être remplacés, conduisant à un coût nettement moins important que le remplacement de l'intégralité du réseau imposé par des configurations End-to-End TT.

4.2.4 Applicabilité de Egress TT

L'approche Egress TT convient bien à des systèmes où la gigue en réception est une contrainte très forte et où la latence l'est moins. Afin de ne pas sur-contraindre le cœur de réseau, la faible gigue est obtenue en "sacrifiant" la latence. En effet, si un message est émis au début de son contrat de production et bénéficie d'une situation favorable pour traverser le réseau, il devra attendre une certaine durée dans le dernier commutateur avant de pouvoir être effectivement envoyé au receveur. Cette situation va résulter en une latence bien supérieure à celle que l'on pourrait observer dans des configurations End-to-End TT. Bien que ce compromis latence/gigue soit compatible avec les exigences de l'industrie aérospatiale, les configurations Egress TT pourraient ne pas être adaptées à des systèmes où non seulement les giges mais aussi les latences doivent être minimales.

Dans les prochains paragraphes, nous décrivons brièvement deux implémentations de cette approche pour des réseaux Ethernet/TSN.

4.3 Exclusive Queue Allocation et Size Based Isolation deux implémentations de Egress TT pour les réseaux TSN

Intéressons-nous maintenant à l'application du concept de configuration Egress TT aux réseaux TSN basés sur le standard 802.1Qbv. Nous proposons dans le manuscrit deux implémentations (voir définition ci-après) intitulée Exclusive Queue Allocation et Size Based Isolation. Pour obtenir des paramètres de configurations de ces deux implémentations, comme détaillé dans le chapitre 9 du manuscrit, nous nous appuyons sur la technique récurrente dans l'état de l'art, c'est-à-dire, une modélisation du système dans un problème de programmation par contraintes. Il faut donc définir un modèle formel du réseau ainsi qu'un ensemble de contraintes décrivant à la fois les exigences du système.

Définition 18 (Implémentation). *Dans ce document et dans le manuscrit, nous utilisons le mot implémentation pour décrire un ensemble de contraintes utilisées pour la génération des configurations réseaux. De fait, une implémentation de Egress TT appliquée aux réseaux TSN est un ensemble de contraintes liées au modèle de configuration de TSN (définis dans le chapitre 9 du manuscrit) qui permettra de générer, à travers un solveur de contraintes, les configurations Egress TT du réseau. Le terme implémentation ne sera pas utilisé pour décrire la configuration d'équipements réels sur un démonstrateur sur la base des paramètres de configuration que nous générons.*

4.3.1 Une première implémentation de Egress TT pour les réseaux 802.1Qbv : Exclusive Queue Allocation

Dans cette première implémentation de Egress TT, nous nous appuyons sur la contrainte d'Exclusive Queue Allocation (détaillée ci-après) pour satisfaire l'exigence d'indépendance à la perte d'un message du système. Une explication détaillée de l'origine et la nécessité de cette contrainte d'isolation dans les réseaux TSN est disponible dans la section Related Works du chapitre 9.

Hypothèses pour l'adaptation de Egress TT aux réseaux TSN

Nous considérons deux hypothèses sur notre modèle pour pouvoir générer ces configurations.

Hypothèse 3 (Synchronisation). *Nous prenons l'hypothèse que les émetteurs et les commutateurs du dernier saut¹ sont synchronisés et que l'erreur de synchronisation est négligeable vis-à-vis des exigences du système.*

Hypothèse 4 (Chemin fixe). *De manière similaire à l'état de l'art, nous prenons l'hypothèse que le chemin des flux dans le réseau est connu a priori et n'évolue pas durant la vie du système (hors mise à jour).*

Choix 3 (Ethernet + Static Priority). *Dans le reste de ce document et dans le manuscrit, nous avons choisi Static Priority, disponible dans Ethernet, comme politique d'accès au médium dans le cœur du réseau.*

Choix 4 (TSN Time Aware Shaper). *Dans le reste de ce document et dans le manuscrit, nous avons choisi le mécanisme TSN Time Aware Shaper comme mécanisme d'ordonnancement des émissions dans les commutateurs des derniers sauts.*

Choix 5 (Politique d'accès au médium dans les ports du derniers saut). *Dans les ports du derniers saut, nous définissons la politique d'accès au médium, dérivée du principe d'exclusion mutuelle de file, suivante : quand la porte (i.e. la transmission gate) d'une file contenant des messages de flux avec gigue est ouverte, toutes les autres portes du port sont fermées. Cela signifie en particulier que les portes des files contenant uniquement du trafic provenant de flux sans gigue peuvent être ouvertes en même temps.*

Choix 6 (Indépendance à la perte d'un message). *Nous considérons que la propriété d'indépendance à la perte d'un message ne s'applique qu'aux flux avec gigue. Cela implique que si la perte d'un message provenant d'un flux sans gigue affecte un autre flux sans gigue, la configuration sera tout de même valide.*

Nous pouvons maintenant définir la contrainte d'Exclusive Queue Allocation.

Concept d'Exclusive Queue Allocation

Afin de satisfaire l'exigence de tolérance aux fautes dans les configurations Egress TT pour les réseaux TSN, au lieu de réutiliser les contraintes d'isolation de l'état l'art qui amènent à un impact applicatif fort, nous introduisons notre propre contrainte : Exclusive Queue Allocation.

Définition 19 (Exclusive Queue Allocation). *Dans un système respectant la contrainte d'Exclusive Queue Allocation, chaque flux avec gigue est associé à une file spécifique dans son port du dernier saut. Aucun autre flux ne peut utiliser cette file.*

1. Comprendre le dernier commutateur dans le chemin de n'importe quel message.

En assurant une isolation spatiale entre les flux (puisqu'il ne peut y avoir que des messages d'un seul flux dans une file), le non déterminisme introduit par le mécanisme Time Aware Shaper est éliminé et la contrainte d'indépendance à la perte de message est satisfaite. En effet, si un message d'un flux avec gigue est perdu dans (ou avant) le port du dernier saut, puisque la stratégie d'accès au médium sur le dernier saut est basée sur de l'exclusion mutuelle de file, aucun message provenant d'une autre file ne peut prendre la place du message perdu. Si cela arrivait, cela aurait pour effet de créer de la gigue pour ce message. Nous l'illustrons dans la figure Fig. 4.9.

Remarque 9. *Puisqu'un port TSN peut, au plus, avoir huit files pour stocker des trames, cette contrainte implique qu'il ne peut y avoir plus de 8 flux avec gigue dans un même port du dernier saut, c'est-à-dire, allant vers la même destination. Pire, s'il y a aussi des flux sans gigue dans ce port, le nombre de flux avec gigue acceptable décroît encore plus (selon le nombre de files occupées).*

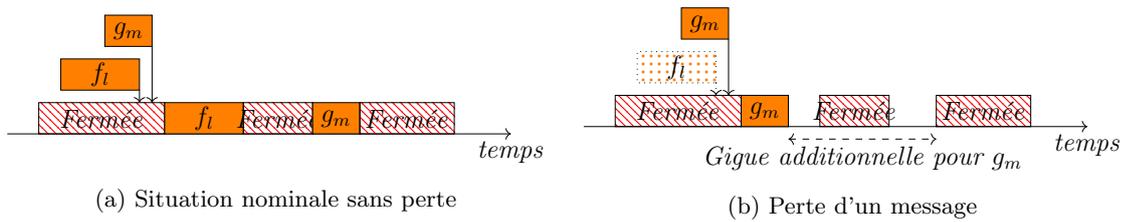


FIGURE 4.9 – Gigue introduite par la perte d'un message

Effectivement, au sein d'une même file, la perte d'un message pourrait introduire de la gigue pour les messages suivants mais puisqu'une file ne contient que des messages d'un même flux, ce cas n'est pas couvert dans l'exigence d'indépendance à la perte d'un message.

Définition des contraintes pour l'utilisation du solveur de contraintes

Dans le chapitre 10 du manuscrit, nous détaillons l'ensemble des contraintes nécessaires pour la modélisation et la résolution du problème de configuration de réseau. Le lecteur peut se reporter à ce chapitre pour plus de détail sur la première implémentation.

4.3.2 Une seconde implémentation pour l'amélioration du passage à l'échelle d'Exclusive Queue Allocation : Size Based Isolation

La première implémentation du concept Egress TT appliquée à des réseaux TSN est intéressante. Néanmoins, elle souffre d'une limitation majeure : un port du dernier saut ne peut recevoir plus de 8 flux avec gigue différents. Bien que cela puisse suffire à certains réseaux embarqués, cela ne convient pas à tous les systèmes industriels considérés dans la thèse (voir Chapitre 9 du manuscrit). C'est pourquoi, nous avons également proposé une seconde implémentation, avec pour objectif d'augmenter d'augmenter le nombre de flux avec gigue par port de réception.

Hypothèses additionnelles

Afin de pouvoir définir la contrainte de Size Based Isolation, nous devons rajouter une hypothèse supplémentaire sur notre modèle.

Hypothèse 5 (Chemin des flux). *Nous prenons l'hypothèse que deux flux ayant la même source et la même destination prennent le même chemin.*

Cette hypothèse servira à réduire l'impact applicatif induit par les configurations obtenues avec Size Based Isolation.

Concept de Size Based Isolation

Avec la contrainte d'Exclusive Queue Allocation, une isolation spatiale était mise en place entre les flux avec gigue dans les ports du dernier saut afin de répondre à l'exigence d'indépendance à la perte d'un message. De fait, une file était réservée pour un flux et aucun autre flux ne pouvait l'utiliser. Avec Size Based Isolation, nous relâchons cette isolation spatiale en autorisant, sous conditions, le partage d'une même file à plusieurs flux avec gigue.

Définition 20 (Size Based Isolation). *Dans un système respectant la contrainte de Size Based Isolation, toutes les trames partageant une même file doivent être placées dans cette file par ordre de taille croissant.*

En s'assurant que les trames respectent cet ordre de taille croissant et en générant des ouvertures de transmission gates de durée spécifique à chaque message, si une trame est perdue, la trame suivante ne prendra pas sa place puisque sa taille sera supérieure à la durée courante d'ouverture de la porte. Cela aura pour effet que le message ne sera pas émis dans le slot libéré par le message perdu mais bien à la date qui avait été configurée au départ. Nous illustrons ce concept dans la figure Fig. 4.10 : nous montrons la situation nominale dans 4.10a et le comportement attendu en cas de perte d'un message dans 4.10b. Grâce à la contrainte de Size Based Isolation, même si le message f_l est perdu, le message j_k n'est pas perturbé. Cette propriété de taille croissante peut par exemple être obtenue en ajoutant du padding aux trames au moment de leur émission dans le réseau.

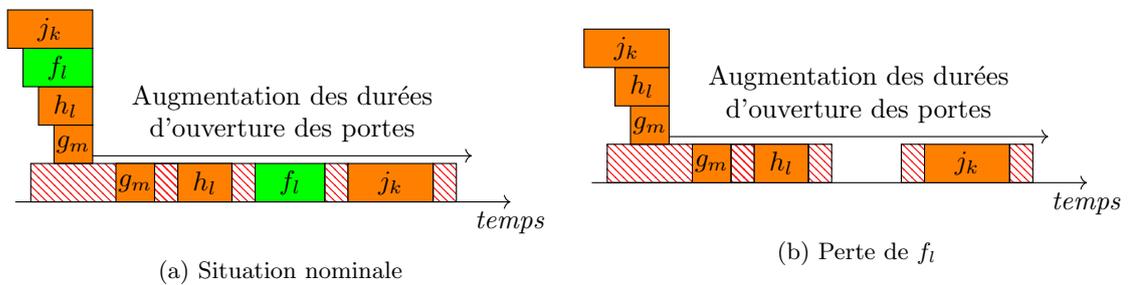


FIGURE 4.10 – Size Based Isolation

De cette manière, il est possible d'augmenter le nombre de flux avec gigue par port du dernier saut. Cependant, cette solution a elle aussi un coût : il faut garantir que l'ordre croissant des tailles sera bien respecté. Puisqu'il nous est impossible de garantir un ordre entre des messages venant de différentes sources au sein d'un même port du dernier saut sans impact négatif sur les applications, nous imposons que des flux partageant la même file doivent venir du même émetteur par le même chemin (ce qui correspond justement à notre précédente hypothèse). Dans cette situation, l'impact applicatif sera légèrement dégradé par rapport à Exclusive Queue Allocation : en plus du contrat de trafic, les émetteurs devront respecter une contrainte d'ordre à l'émission.

4.3.3 Limitations de Exclusive Queue Allocation, Size Based Isolation et Egress TT

Maintenant que nous avons résumé brièvement le principe des configurations Egress TT et de ses deux implémentations, nous présentons dans le prochain paragraphe les limitations de ces configurations.

Tout d'abord, les implémentations avec Exclusive Queue Allocation ne fonctionneront jamais dans des situations où un équipement est censé recevoir plus de 8 flux avec gigue.

Ensuite, les implémentations avec Size Based Isolation ne fonctionneront jamais dans les situations où un équipement est censé recevoir des flux venant de plus de 8 sources différentes.

De plus, le nombre de flux avec gigue par file dans les implémentations avec Size Based Isolation est limité par la granularité i.e. la plus petite durée d'une ouverture de porte dans le mécanisme Time Aware Shaper. Par exemple, avec une granularité de $1\mu s$, la plus petite ouverture de porte sera capable de transmettre 125 bytes (à 1Gbit/s). Alors, en considérant que la taille maximale des trames est 1518 octets, une file dans ce port ne pourra pas accepter plus de 13 flux avec gigue différents. Les détails de ce calcul sont disponibles dans le manuscrit.

Enfin, d'autres limitations intrinsèques au concept de configuration Egress TT existent, nous les détaillons dans le manuscrit.

Dans toutes les situations mentionnées précédemment, End-to-End TT sera toujours une meilleure approche puisqu'elle permettra de générer des configurations valides. Néanmoins, nous sommes convaincus que l'amélioration du passage à l'échelle, en particulier le coût en temps et en ressources réduit, et l'impact applicatif léger peuvent intéresser certains industriels pour leur réseaux temps-réels.

Dans les prochains paragraphes, nous résumons quelques résultats d'expérimentations obtenues en configurant différents cas d'usages avec notre méthode Egress TT et l'approche End-to-End TT de l'état de l'art.

4.3.4 Évaluation Expérimentale de la performance de Egress TT par rapport à l'état de l'art

Dans le chapitre 11 du manuscrit, nous avons proposé une évaluation de la performance des configurations Egress TT et de leurs implémentations pour un réseau TSN par rapport à une implémentation de configuration End-to-End TT pour ce même réseau. Nous résumons ci-après quelques résultats expérimentaux.

Coût en temps et en ressources d'une configuration

Dans cette première expérience, nous avons souhaité comparer le coût en temps et en ressources informatiques pour la génération d'une configuration Egress TT et d'une configuration End-to-End TT. Nous avons proposé une topologie de réseau relativement simple ainsi qu'un ensemble de flux. Nous avons fait évoluer deux paramètres sur ce réseau : sa taille en terme de nombre d'équipements et le nombre de flux qui transitent dans le réseau. L'évolution de ces deux paramètres est représentée dans les figures Fig.4.11 et Fig. 4.12. La liste des flux, leurs caractéristiques et leurs routes dans le réseau sont disponibles dans le manuscrit.

La comparaison des performances entre les deux configurations sera basée sur deux métriques : le nombre de contraintes nécessaires pour la description du problème dans le solveur de contraintes et le temps nécessaire pour la génération d'une configuration avec le solveur. Toutes les configurations ont été implémentées dans le langage OPL et la résolution du problème de programmation par contrainte

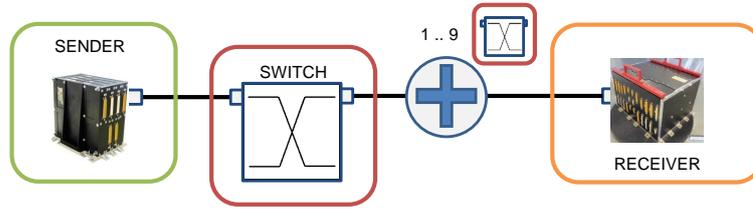


FIGURE 4.11 – Évolution de la taille des chemins

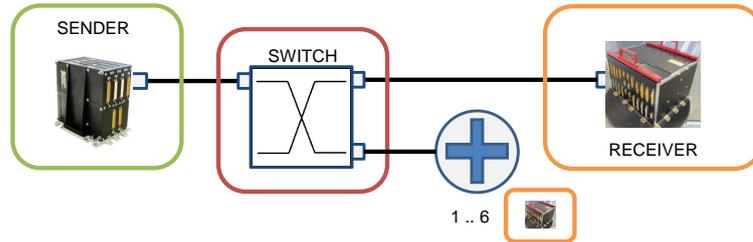


FIGURE 4.12 – Évolution du nombre de flux

a été faite via l'outil CPLEX en version v12.9.0 sur une machine Ubuntu avec un processeur *Intel Xeon E5-2600 v3 @ 2.6GHz* et 62GiOctets de mémoire.

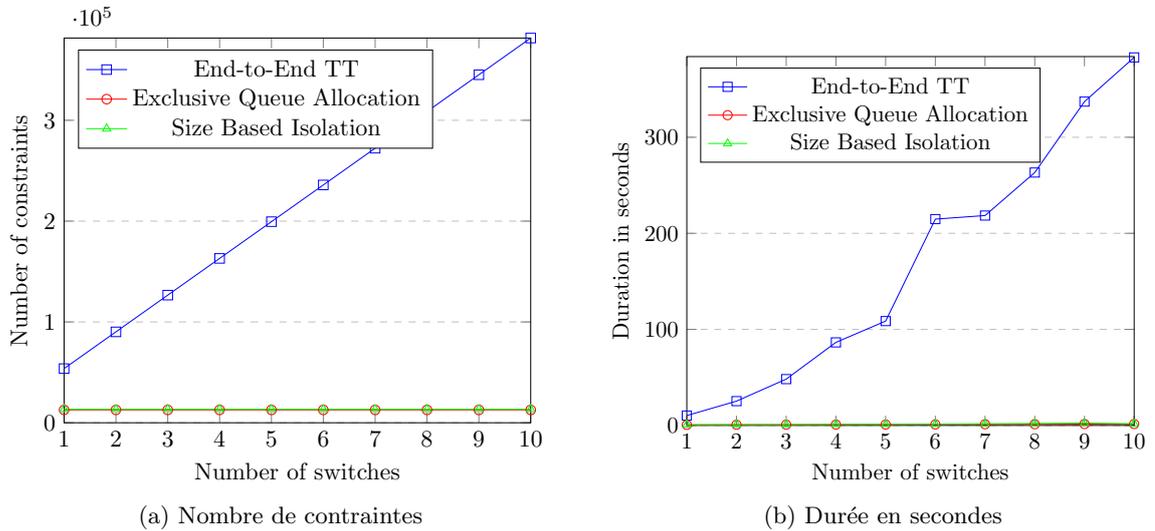


FIGURE 4.13 – Augmentation de la taille des chemins : End-to-End TT vs. Egress TT

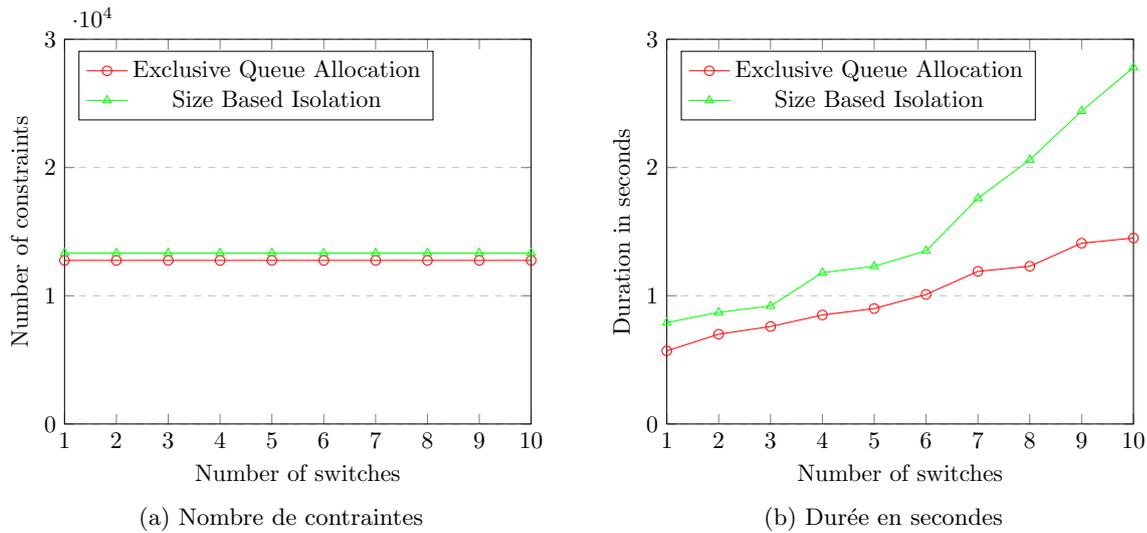


FIGURE 4.14 – Augmentation de la taille des chemins : Exclusive Queue Allocation vs. Size Based Isolation

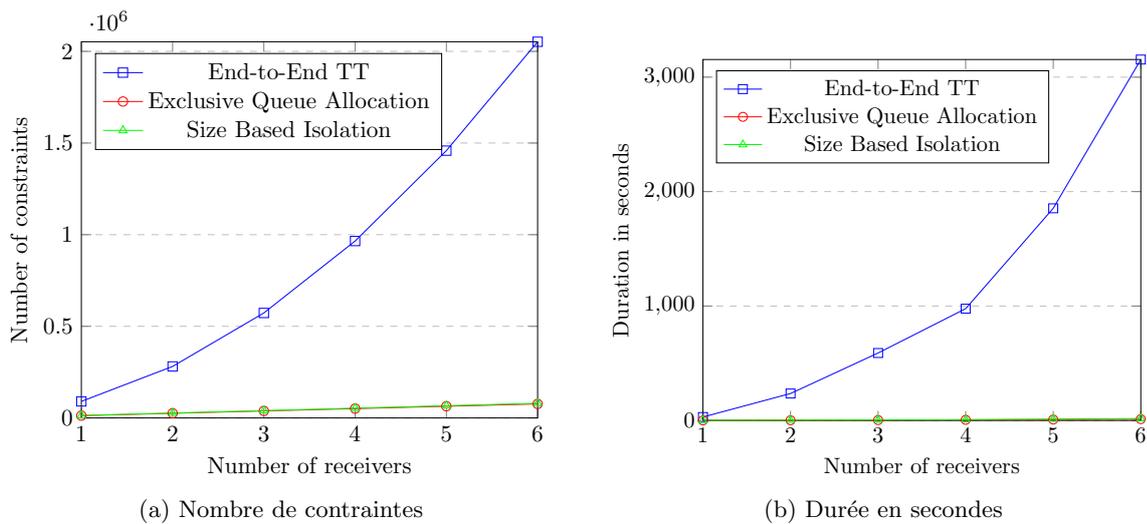


FIGURE 4.15 – Augmentation du nombre de receveurs : End-to-End TT vs. Egress TT

Évolution de la taille des chemins Pour cette première expérience, les résultats expérimentaux présentés dans les figures Fig. 4.13a, Fig. 4.14a, Fig. 4.15a et Fig. 4.16a montrent que notre approche Egress TT est considérablement moins coûteuse en temps et en ressources informatiques que l'approche End-to-End TT. De plus, l'implémentation avec Size Based Isolation apparaît un peu plus coûteuse que l'implémentation avec Exclusive Queue Allocation mais cette augmentation (du coût) reste négligeable par rapport au coût d'une configuration avec l'approche End-to-End TT. Ce résultat était attendu puisque Egress TT a justement été conçue avec la volonté de réduire le coût d'une configuration. Une analyse plus détaillée de ces résultats est disponible dans le chapitre 11 du

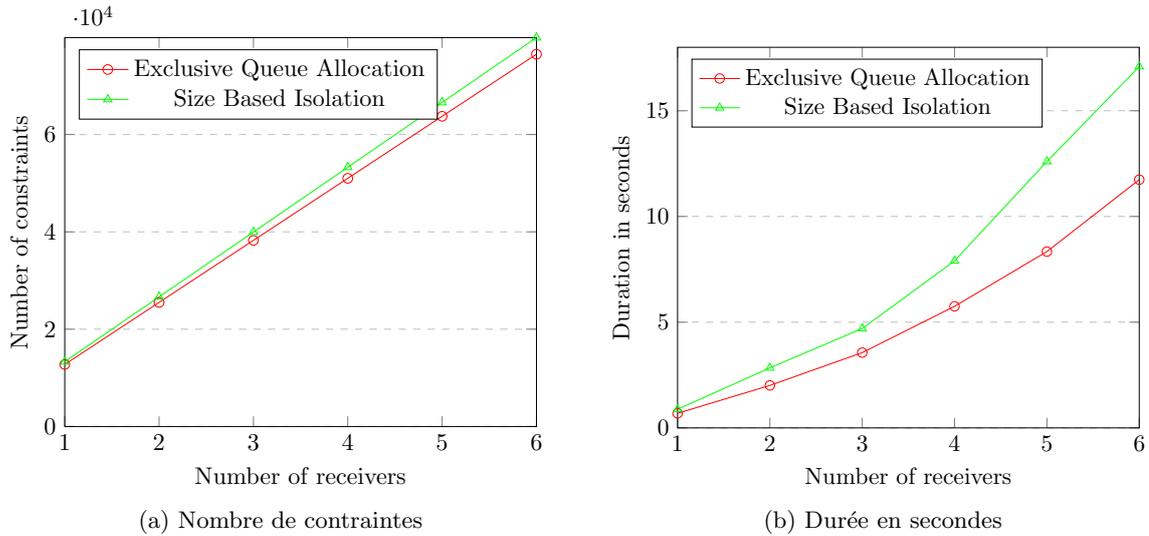


FIGURE 4.16 – Augmentation du nombre de receivers : Exclusive Queue Allocation vs. Size Based Isolation

manuscrit.

Impact sur les Latences

Dans cette deuxième expérience, nous avons souhaité évaluer l'impact sur les latences dans le réseau des implémentations Exclusive Queue Allocation et Size Based Isolation par rapport aux configurations End-to-End TT. Pour se faire, nous avons utilisé la topologie représentée dans la figure Fig. 4.17. La liste des flux utilisée est détaillée dans le manuscrit.

Les courbes suivantes représentent une moyenne par flux, des latences observées pour les messages de ce flux. Il a fallu 5 secondes pour calculer une configuration avec Exclusive Queue Allocation, 12 secondes pour une configuration avec Size Based Isolation et 30 minutes pour une configuration avec End-to-End TT.

Les résultats expérimentaux présentés dans la figure Fig. 4.18 montrent que les configurations Egress TT génèrent de plus grandes latences que les configuration End-to-End TT. Encore une fois, ce résultat était attendu. En effet, afin d'obtenir de faibles gigue dans Egress TT, la latence est sacrifiée.

Les deux expériences présentées ci-dessus montrent que l'approche Egress TT réduit bien l'effort nécessaire pour générer une configuration d'un réseau Time Sensitive Networking au coût d'une latence plus grande que celles qui seraient obtenues avec l'approche End-to-End TT de l'état de l'art, plus coûteuse en temps et en ressources.

4.3.5 Configurations de systèmes industriels

Après ces expériences sur des systèmes relativement petits, nous avons souhaité évaluer l'approche Egress TT sur des systèmes du domaine spatial existant dans l'état de l'art. Nous en avons considéré deux : le système satellite générique d'Airbus (disponible dans [16]) et le système du Crew

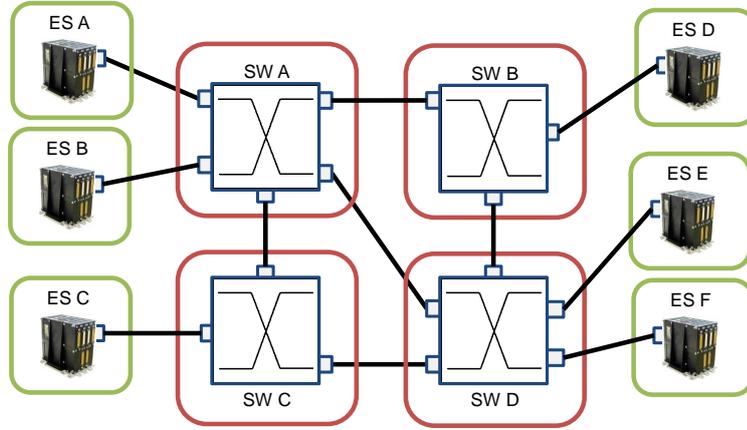


FIGURE 4.17 – Topologies pour l'évaluation de l'impact des latences

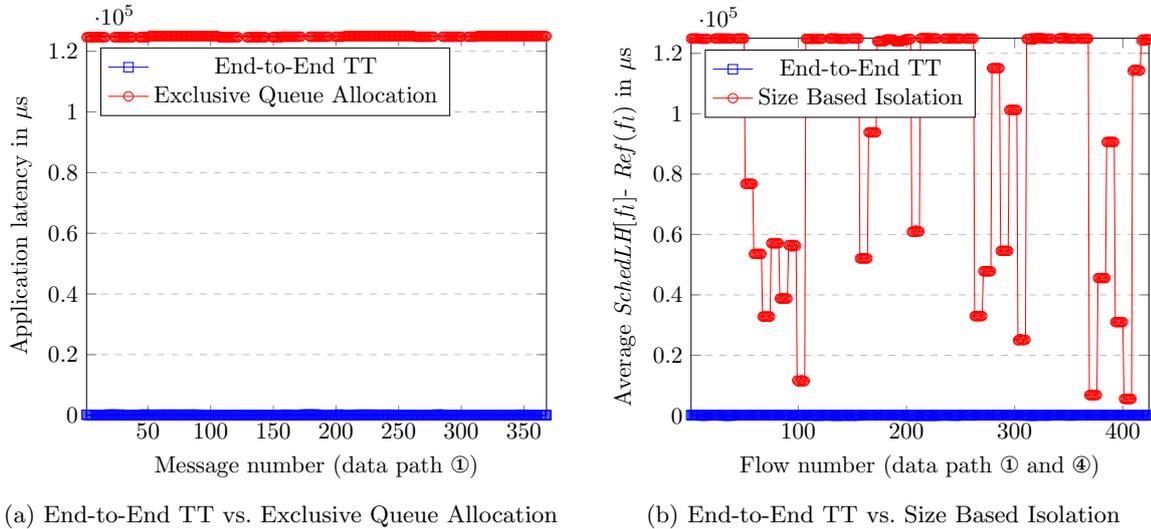


FIGURE 4.18 – End-to-End TT vs. Exclusive Queue Allocation

Exploration Vehicule d'ORION (disponible dans [122]). Nous détaillons ces deux systèmes industriels dans le chapitre 9 du manuscrit. Pour ces deux expériences, nous avons mesuré la latence moyenne par flux obtenu avec chaque implémentation. Nous proposons deux types de courbes, des courbes représentant des latences réseaux (i.e. avec la définition de latence de l'état de l'art) par flux et des courbes représentant des latences applicatives (i.e. avec la définition de latence de notre modèle). Dans ces expériences, le nombre de flux par port du dernier saut dépasse parfois 8 flux avec gigue, ce qui rend impossible l'utilisation de l'implémentation Exclusive Queue Allocation. De fait, pour pouvoir tout de même évaluer cette implémentation, nous réduisons le nombre de flux dans les ports pour qu'il ne dépasse pas 8. Nous évaluons alors Exclusive Queue Allocation et notre implémentation de End-to-End TT sur ce système réduit. Pour le système Orion, nous n'avons pas tenté de générer

des configurations avec Exclusive Queue Allocation.

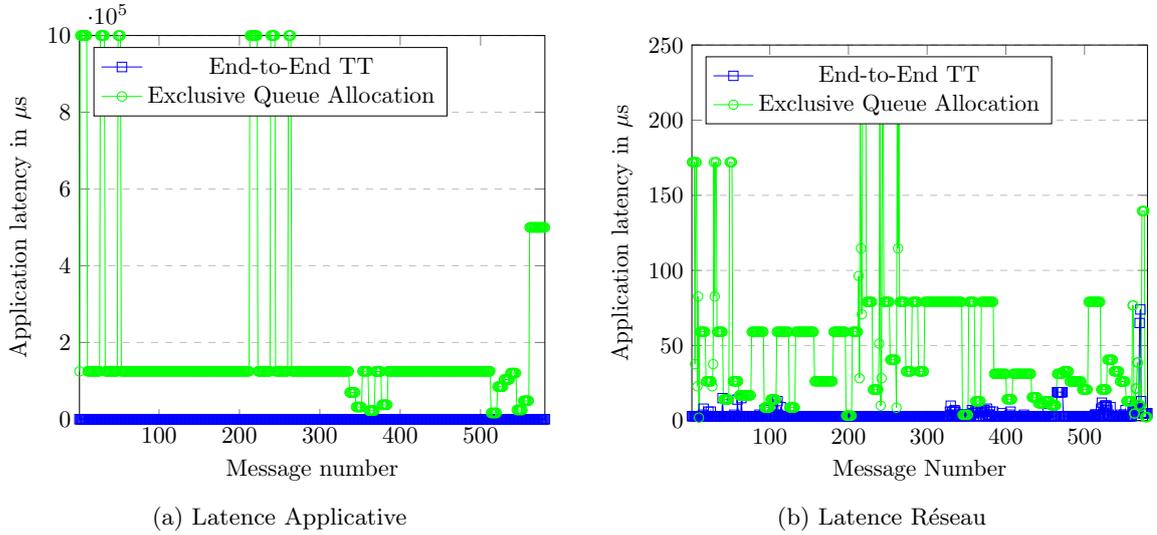


FIGURE 4.19 – Satellite Airbus Générique : End-to-End TT vs. Exclusive Queue Allocation

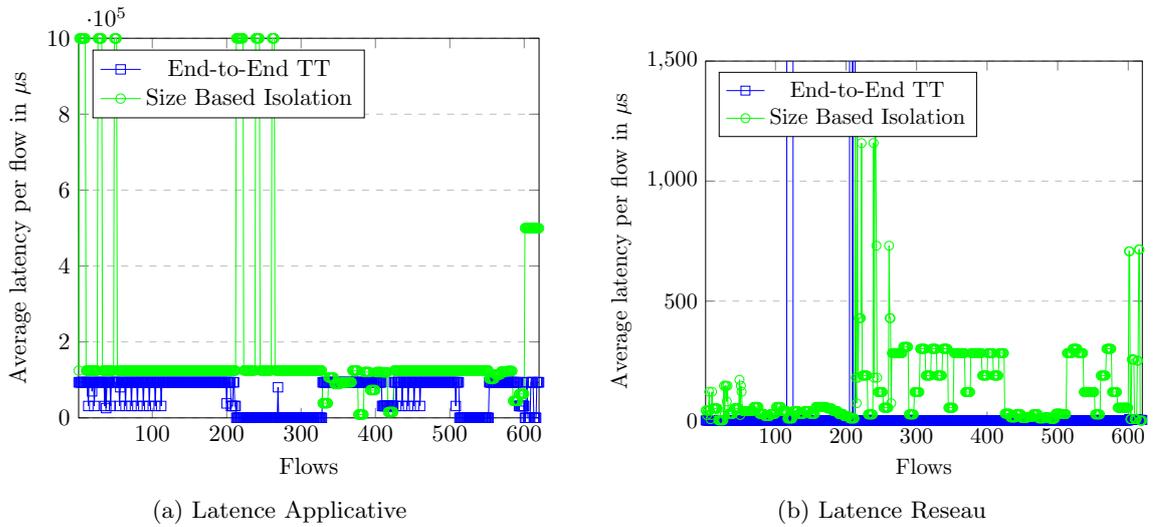


FIGURE 4.20 – Satellite Airbus Générique : End-to-End TT vs. Size Based Isolation

Les résultats expérimentaux présentés dans les figures Fig. 4.19, Fig. 4.20 et Fig. 4.21 confirment les observations effectuées dans nos premières expérimentations, c'est-à-dire que l'effort pour obtenir une configuration Egress TT est réduit au coût d'une latence plus grande. Une analyse plus poussée de ces résultats expérimentaux est proposée dans le manuscrit.

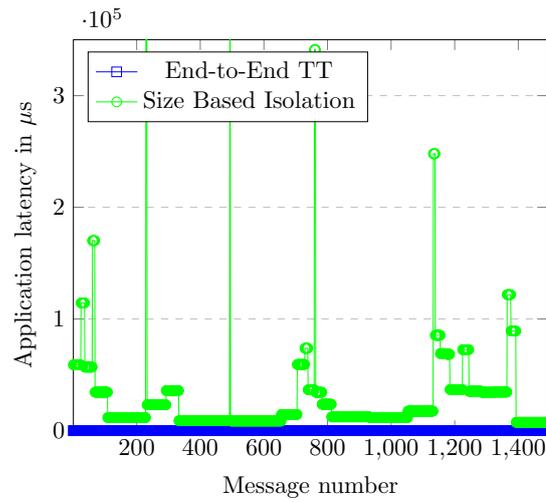


FIGURE 4.21 – Orion CEV : End-to-End TT vs. Exclusive Queue Allocation

Conclusion sur la seconde contribution

Dans cette seconde contribution, nous avons mis en place une démarche pour configurer des réseaux temps-réel avec une approche nouvelle Egress TT. Cette approche, basée sur des émissions planifiées de messages dans le dernier équipement dans le chemin de n'importe quel flux, permet de réduire l'effort nécessaire pour générer une configuration par rapport aux approches End-to-End TT de l'état de l'art. Cependant, afin d'obtenir de faibles gigue, l'approche Egress TT augmente les latences des messages dans le réseau. Elle est donc applicable tant que la contrainte de latence peut être assurée à l'aide des mécanismes asynchrones implémentés dans le cœur de réseau. Nous avons évalué sur des petits systèmes puis sur des systèmes industriels du domaine notre approche Egress TT et deux implémentations, Exclusive Queue Allocation et Size Based Isolation, pour des réseaux TSN. Ces expérimentations ont confirmé l'intérêt et les limites de l'approche Egress TT.

Chapitre 5

Conclusion

A la lumière de l'augmentation du besoin en performance dans l'industrie aérospatiale, l'objectif de cette thèse était de discuter de l'adaptabilité d'une technologie émergente nommée Time Sensitive Networking (TSN), vis-à-vis des exigences des prochaines générations de satellites. Cette étude est incluse dans la feuille de route stratégique TANIA-DP (Technological Assessment for New Instruments and Avionics - Data Processing) d'Airbus, ayant pour objectif, entre autres, de fournir de nouveaux standards de communication pour les réseaux *Plateforme & Charge Utile* afin d'améliorer les performances, les coûts, la compatibilité avec les standards du segment sol, ou encore la gestion de futures missions dans l'espace.

Les Standards TSN

Time Sensitive Networking est le dernier standard Ethernet de l'IEEE. Développé à la fin des activités du groupe de travail IEEE sur AVB (Audio Video Bridging), le groupe de travail TSN a proposé plus d'une vingtaine de standards dans l'objectif de fournir, à un réseau Ethernet, des capacités de gestion de trafic temps réel et de trafic haut-débit, de gestion des fautes, de synchronisation et de distribution du temps ainsi que des capacités de gestion de configurations. En 2017/2018, la popularité de TSN a commencé à grandir, en particulier dans trois domaines industriels i.e. l'automobile, l'automatisation industrielle et la 5G. Très vite, d'autres industriels ont manifesté leur intérêt pour TSN, notamment le secteur aérospatial.

La liste des standards de TSN n'est toujours pas fixée : de nouveaux documents continuent à être produits par le groupe de travail au moment où nous écrivons ce manuscrit. Avec cette augmentation du nombre de standards, les industriels s'intéressant à TSN ont proposé de définir des *profils* décrivant un sous-ensemble de standards et/ou de paramètres pour l'usage du TSN dans leurs contextes spécifiques (ex. Profil pour l'automobile [63]). Dans ce sens, nous fûmes les premiers à mettre en avant à la fois l'intérêt de l'industrie aérospatiale pour la technologie TSN et un premier aperçu d'un profil pour l'usage du TSN dans l'espace. Ce profil est maintenant en train d'être développé dans un effort partagé entre l'IEEE et la SAE.

Contributions

Afin d'évaluer l'adaptabilité de Time Sensitive Networking à l'industrie aérospatiale, nous avons décrit dans le manuscrit la démarche en deux étapes mise en place durant la thèse. D'abord, nous

nous sommes assurés, dans une approche qualitative, que TSN était bien un candidat pertinent pour le réseau unifié nouvelle génération. Ensuite, dans une approche quantitative, nous avons évalué la capacité de TSN à satisfaire les besoins en qualité de service des futures missions.

Perspectives sur la méthodologie Egress TT

Nous envisageons plusieurs perspectives pour de futures activités sur l'approche Egress TT.

Nouvelles fonctionnalités dans le modèle

Tout d'abord, de nouvelles fonctionnalités pourraient être ajoutées au modèle et donc être traduites dans le problème de programmation par contrainte sans véritable effort. Nous en détaillons quelques unes ci-après. D'abord, le modèle pourrait être modifiée pour supporter des flux multicast (i.e. avec une source mais plusieurs destinations) afin de pouvoir appliquer l'approche à une plus grande variété de systèmes. Comme expliqué dans le chapitre 11 dans le manuscrit, nous avons dû adapter le use case Orion à notre modèle puisqu'il y avait originellement des flux multicasts. De fait, les configurations que nous avons générées ne sont pas tout à fait applicables au système d'origine. De même, le modèle pourrait être modifié pour supporter des tailles de messages applicatifs plus grandes. Cela signifierait qu'un message applicatif serait maintenant divisé en plusieurs trames Ethernet. Nous proposons une intuition sur la modification du modèle pour cette nouvelle capacité dans l'Annexe C.1 du manuscrit. En restant sur la thématique des trames, une autre modification pourrait être apportée au modèle : l'introduction de la *frame preemption* (préemption de trames), décrite par les standards TSN IEEE 802.1Qbv et IEEE 802.13br. Ces standards sont présentés dans l'Annexe A.1 du manuscrit. Pour intégrer la préemption de trames dans le modèle, il serait notamment nécessaire de modifier la formule du calcul de la borne sur le délai de traversée du réseau.

D'ailleurs, le calcul de la borne mentionnée précédemment pourrait lui aussi être amélioré. En effet, la formule encodée dans le manuscrit est peut-être un peu naïve. Ceci étant dit, elle était suffisante pour nos cas d'usages. Une des pistes d'amélioration à envisager serait de coupler le solveur de contrainte avec un outil de calcul réseau, ce dernier offrant de meilleures bornes. Il faut noter cependant que cela n'est, à l'heure actuelle, pas possible avec le solveur de contraintes que nous avons utilisé. De plus, cela pourrait potentiellement détériorer le temps réduit de configuration permis par Egress TT.

La dernière modification des fonctionnalités que nous proposerons serait d'intégrer le Credit Based Shaper (contrôle de flux basé crédit) comme algorithme de gestion de flux soit pour certains flux sans contraintes de gigue soit pour une troisième classe de trafic. Pour ce faire, plusieurs éléments devraient être modifiés dans le solveur. Premièrement, de nouvelles variables de décisions devraient être ajoutées afin de pouvoir générer les paramètres (idle slope et send slope) liés à l'usage du CBS. Deuxièmement, la formule de calcul du délai devrait être adaptée pour prendre en compte le trafic CBS. Troisièmement, la philosophie dite "exclusive gating" avec laquelle nous configurons actuellement les ports du dernier saut devrait être repensée.

Nouveaux standards

En plus des nouvelles fonctionnalités liées au standard 802.1Qbv (et à la préemption de trames), de nouveaux standards pourraient aussi être intégrés au modèle pour se rapprocher encore plus des systèmes réels. En premier, le standard TSN 802.1CB Frame Replication and Elimination for Reliability pourrait être rajouté au modèle. Cela provoquerait une augmentation du trafic dans le réseau qui aurait peut-être un impact sur le calcul du délai. Un autre standard TSN pourrait

aussi être intégré au modèle : le standard 802.1AS-2020. Ce standard est en charge de gérer la synchronisation du réseau. Encore une fois, cela aura pour conséquence d'introduire plus de trafic (avec peut-être des exigences en qualité de service différentes) dans le modèle. De plus, selon la qualité de la synchronisation recherchée, le nombre de messages pourrait peut-être varier ce qui signifie que les configurations générées par l'approche Egress TT pourraient être différentes.

Prochaines étapes vis-à-vis de l'usage du TSN dans l'industrie aérospatiale

En parallèle des nouvelles fonctionnalités et des nouveaux standards à ajouter à l'approche Egress TT, il reste des activités à mener, d'un point de vue industriel, pour terminer le processus de sélection de la technologie du réseau embarqué satellite de nouvelle génération. Cela concerne non seulement l'étude présentée dans ce manuscrit autour du TSN mais aussi les autres technologies finalistes (i.e. TTEthernet et Spacefibre). En effet, dans la seconde contribution de la thèse, nous nous sommes uniquement intéressés à la qualité de service en performance de TSN. Il serait nécessaire de formaliser un modèle de tolérance aux fautes (FDIR) clair pour ces réseaux de nouvelle génération. Les commutateurs seraient-ils utilisés en redondance chaude, en redondance tiède ou bien en redondance froide ? Les liens seraient-ils dupliqués, ou tripliqués ? La redondance serait-elle appliquée à l'intégralité du réseau ou seulement à un sous-ensemble de commutateurs et de liens ? Quelles sont les valeurs attendues pour la disponibilité et la fiabilité du système ? Comment ces paramètres peuvent-ils être évalués sur un réseau TSN/TTE/Spacefibre ? De plus, dans cette thèse, nous avons uniquement regardé les aspects couche 2 du modèle OSI. Dans une logique d'industrialisation du TSN (et des autres technologies), il serait nécessaire de s'intéresser à la disponibilité des médias physiques spatialisables capables de supporter les hauts volumes de données échangés à bord. Ce médium sera-t-il basé sur une fibre optique ? sur une paire torsadée ? ou encore un câble spécifiquement développé par l'industriel ? Serait-il raisonnable d'adapter une couche physique SpaceFibre à une couche MAC TSN ? Les premières activités en cours chez l'industriel à ce sujet laisseraient penser que le médium physique du réseau nouvelle génération pourrait être basé sur une fibre optique mais de nouvelles études seront nécessaires pour concevoir cette couche physique.

Bibliography

- [1] What is spacefibre ? <https://www.star-dundee.com/spacefibre/getting-started/what-is-spacefibre/>. Star Dundee - accessed on 10/09/2021.
- [2] Federal Aviation Administration. Data network evaluation criteria report. Technical report, FAA, 2009. https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/AR-09-27.pdf.
- [3] Aeronautical Radio Incorporated. ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network. Technical Report ARINC 664P7.
- [4] Jasmin Ajanovic. Pci express 3.0 overview. In *Proceedings of Hot Chip: A Symposium on High Performance Chips*, volume 69, page 143, 2009.
- [5] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello. Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment. In *2012 IEEE Vehicular Networking Conference (VNC)*, pages 187–194, 2012. doi:10.1109/VNC.2012.6407430.
- [6] Krzysztof Apt. *Principles of constraint programming*. Cambridge university press, 2003.
- [7] Philip Axer, Daniel Thiele, Rolf Ernst, and Jonas Diemer. Exploiting shaper context to improve performance bounds of ethernet avb networks. In *Proceedings of the 51st Annual Design Automation Conference, DAC '14*, page 1–6, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2593069.2593136.
- [8] Wojciech Baron, Anna Arestova, Christoph Sippl, Kai-Steffen Hielscher, and Reinhard German. LETT: An execution model for distributed real-time systems. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pages 1–7. IEEE, 2021.
- [9] Sanjoy K Baruah, Alan Burns, and Robert I Davis. Response-time analysis for mixed criticality systems. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 34–43. IEEE Computer Society, 2011.
- [10] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Worst-case end-to-end delay analysis of an avionics afdx network. In *Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10*, page 1220–1224, Leuven, BEL, 2010. European Design and Automation Association.
- [11] Lucia Lo Bello. Novel trends in automotive networks: A perspective on ethernet and the ieee audio video bridging. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8, 2014. doi:10.1109/ETFA.2014.7005251.

- [12] Anne Bouillard, Marc Boyer, and Euriell Le Corronc. *Deterministic Network Calculus, from Theory to Practical Implementation*. ISTE, 2018.
- [13] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)*, pages 1–9, march 2010. doi:10.1109/INFOCOM.2010.5461912.
- [14] Marc Boyer, Nicolas Navet, and Marc Fumey. Experimental assessment of timing verification techniques for AFDX. In *Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems*, Toulouse, France, February 2012. URL: <https://hal.archives-ouvertes.fr/hal-02189869>.
- [15] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Suitability of time sensitive networking for space ? TSN A Conference, 2019.
- [16] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. TSN Support for Quality of Service in Space ? In *13th Junior Researcher Workshop on Real-Time Computing (JWRTC 2019)*, Toulouse, France, 2019.
- [17] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Formal Specification of Satellite On-Board Networks Requirements. working paper or preprint, September 2020. URL: <https://hal.archives-ouvertes.fr/hal-02926971>.
- [18] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. TSN Support for Quality of Service in Space. In *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, Toulouse, France, January 2020. URL: <https://hal.archives-ouvertes.fr/hal-02441327>.
- [19] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Comparative study of ethernet technologies for next-generation satellite on-board networks. In *Proc. of the 40th Int. Conference on Digital Avionics System Conference (DASC)*, 2021.
- [20] Pierre-Julien Chaine, Marc Boyer, Claire Pagetti, and Franck Wartel. Comparative study of high throughput technologies for next-generation satellite on-board networks. In *Data Systems In Aerospace (DASIA)*, 2021.
- [21] Tom S. Chan. *Time-Division Multiple Access*, chapter 49, pages 769–778. John Wiley & Sons, Ltd, 2007. doi:<https://doi.org/10.1002/9781118256114.ch49>.
- [22] Remi Clavier, Pierre Sautereau, and Jean-Francois Dufour. TTEthernet, a Promising Candidate for Ariane 6. In L. Ouwehand, editor, *DASIA 2014 - DAta Systems In Aerospace*, volume 725 of *ESA Special Publication*, page 34, August 2014.
- [23] Silviu S. Craciunas and Ramon Serna Oliver. Smt-based task- and network-level static schedule generation for time-triggered networked systems. In *Proceedings of the 22nd International Conference on Real-Time Networks and Systems, RTNS '14*, page 45–54, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2659787.2659812.
- [24] Silviu S. Craciunas and Ramon Serna Oliver. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Syst.*, 52(2):161–200, March 2016. doi:10.1007/s11241-015-9244-x.

- [25] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in ieee 802.1qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems, RTNS '16*, 2016.
- [26] Silviu S. Craciunas, Ramon Serna Oliver, and Wilfried Steiner. Formal scheduling constraints for time-sensitive networks, 2017. [arXiv:1712.02246](https://arxiv.org/abs/1712.02246).
- [27] Rodney Cummings, Kai Richter, Rolf Ernst, Jonas Diemer, and Arkadeb Ghosal. Exploring use of ethernet for in-vehicle control applications: Afdx, ttethernet, ethercat, and avb. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 5:72–88, 05 2012. doi:10.4271/2012-01-0196.
- [28] Hugo Daigmore and Marc Boyer. Impact on credit freeze before gate closing in CBS and GCL integration into TSN. In *20th International Conference on Real-Time Networks and Systems (RTNS 2019)*, pages 1–10, 2019.
- [29] Hugo Daigmore. *Analyse des interactions entre flux synchrones et flux asynchrones dans les réseaux temps réel*. Theses, UNIVERSITE DE TOULOUSE, January 2019. URL: <https://hal.archives-ouvertes.fr/tel-02190134>.
- [30] Hugo Daigmore. *Analyse des interactions entre flux synchrones et flux asynchrones dans les réseaux temps réels*. PhD thesis, ISAE-SUPAERO, 2019.
- [31] Hugo Daigmore, Marc Boyer, and Luxi Zhao. Modelling in network calculus a TSN architecture mixing Time-Triggered, Credit Based Shaper and Best-Effort queues. working paper or preprint, June 2018. URL: <https://hal.archives-ouvertes.fr/hal-01814211>.
- [32] Leonardo de Moura and Nikolaj Bjørner. Z3: an efficient smt solver. volume 4963, pages 337–340, 04 2008. doi:10.1007/978-3-540-78800-3_24.
- [33] Department of Defense – DoD. MIL-STD-1553B – Aircraft Internal Time Division Command/Response Multiplex Data Bus. Technical report, 1978.
- [34] J. Diemer, J. Rox, R. Ernst, F. Chen, K. Kremer, and K. Richter. Exploring the worst-case timing of ethernet avb for industrial applications. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 3182–3187, Oct 2012. doi:10.1109/IECON.2012.6389389.
- [35] Jonas Diemer, Daniel Thiele, and Rolf Ernst. Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching. pages 1–10, 06 2012. doi:10.1109/SIES.2012.6356564.
- [36] Frank Dürr and Naresh Ganesh Nayak. No-wait packet scheduling for ieee time-sensitive networks (tsn). In *Proceedings of the 24th International Conference on Real-Time Networks and Systems, RTNS '16*, page 203–212, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2997465.2997494.
- [37] European Space Agency. SAVOIR Functional Reference Architecture. Technical Report Technical Note 001, ESA, 2016. <https://essr.esa.int/project/savoir>.
- [38] European Space Agency. OSRA Communication Network Specification. Technical Report Technical Note 003, ESA, 2017. <https://essr.esa.int/project/savoir>.

- [39] European Space Agency. Spacewire - CCSDS Packet Transfer Protocol. Technical Report ECSS-E-ST-50-53C, ESA, 2018.
- [40] European Space Agency. Spacewire - Protocol Identification. Technical Report ECSS-E-ST-50-51C, ESA, 2018.
- [41] European Space Agency. Spacewire - Remote Memory Access Protocol. Technical Report ECSS-E-ST-50-52C, ESA, 2018.
- [42] European Space Agency. Spacewire-Links, nodes, routers and network. Technical Report ECSS-E-ST-50-12C, ESA, 2018.
- [43] European Space Agency. SpaceFibre – Very high-speed serial link. Technical Report ECSS-E-ST-50-11C, ESA, 2019.
- [44] Albert Ferrer Florit, ALberto Gonzalez Villafranca, and Steve Parkes. SpaceFibre multi-lane: SpaceFibre, long paper. In *2016 International SpaceWire Conference (SpaceWire)*. IEEE, 2016.
- [45] Voica Gavriluț, Bahram Zarrin, Paul Pop, and Soheil Samii. Fault-tolerant topology and routing synthesis for ieee time-sensitive networking. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS '17*, pages 267–276, New York, NY, USA, 2017. ACM. URL: <http://doi.acm.org/10.1145/3139258.3139284>, doi:10.1145/3139258.3139284.
- [46] V. Gavriluț, L. Zhao, M. L. Raagaard, and P. Pop. Avb-aware routing and scheduling of time-triggered traffic for tsn. *IEEE Access*, 6:75229–75243, 2018. doi:10.1109/ACCESS.2018.2883644.
- [47] J. Goossens and S. Baruah. Multiprocessor algorithms for uniprocessor feasibility analysis. In *Proceedings Seventh International Conference on Real-Time Computing Systems and Applications*, pages 315–322, 2000. doi:10.1109/RTCSA.2000.896407.
- [48] Guy Pujolles. *Les Réseaux*. Eyrolles, 2018.
- [49] D.A. Gwaltney and J.M. Briscoe. Comparison of Communication Architectures for Spacecraft Modular Avionics Systems. Technical Report NASA/TM-2006-214431, NASA - Marshall Space Flight Center, Alabama, 2006. <http://ntrs.nasa.gov/search.jsp?R=20060050129>.
- [50] Tasnim Hamza, Jean-Luc Scharbarg, and Christian Fraboul. Priority assignment on an avionics switched Ethernet network (QoS AFDX). In *IEEE International Workshop on Factory Communication Systems - WFCS 2014*, pages pp. 1–8, Toulouse, France, May 2014. URL: <https://hal.archives-ouvertes.fr/hal-01147288>.
- [51] T.A. Henzinger, B. Horowitz, and C.M. Kirsch. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003. doi:10.1109/JPROC.2002.805825.
- [52] Oana Hotescu. *Vers la convergence de réseaux dans l'avionique*. PhD thesis, Toulouse INP, 2020.

- [53] Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, and Saad Mubeen. Synthesising schedules to improve qos of best-effort traffic in tsn networks. In *29th International Conference on Real-Time Networks and Systems (RTNS'21)*, April 2021. URL: <http://www.es.mdh.se/publications/6159->.
- [54] IEEE. IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks—Amendment 28: Per-Stream Filtering and Policing. Technical Report IEEE 802.1Qci. URL: https://standards.ieee.org/standard/802_1Qci-2017.html.
- [55] IEEE. IEEE 1588v2 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. Technical Report IEEE 1588v2, 2008.
- [56] IEEE. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks. Technical Report IEEE 802.1Q, 2008.
- [57] IEEE. Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. Technical Report IEEE 802.1AS, 2011-2020. URL: https://standards.ieee.org/standard/802_1AS-2020.html.
- [58] IEEE. IEEE Standard for Ethernet. Technical Report IEEE 802.3, 2013-2015-2018. URL: https://standards.ieee.org/standard/802_3-2018.html.
- [59] IEEE. Ieee 802.3br-2016, ieee standard for ethernet, amendment 5: Specification and management parameters for interspersing express traffic and ieee 802.1qbu, ieee standard for frame preemption. Technical report, 2016.
- [60] IEEE. IEEE Standard for Ethernet. Technical Report IEEE 802.3AS, 2016. URL: https://standards.ieee.org/standard/802_3as-2006.html.
- [61] IEEE. IEEE Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic. Technical Report IEEE 802.3br, 2016. URL: https://standards.ieee.org/standard/802_3br-2016.html.
- [62] IEEE. Time Sensitive Networking Task Group. Technical report, 2016. <https://1.ieee802.org/tsn/>.
- [63] IEEE. IEEE Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks - Amendment 34:Asynchronous Traffic Shaping. Technical Report IEEE 802.1Qcr, 2020.
- [64] IEEE. Draft Standard for Local and Metropolitan Area Networks — Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications. Technical report, 2021. <https://1.ieee802.org/tsn/802-1dg/>.
- [65] IEEE/SAE. IEEE/SAE TSN for Aerospace Onboard Ethernet Communications. Technical Report IEEE P802.1DP, 2020. URL: <https://1.ieee802.org/tsn/802-1dp/>.
- [66] IETF. Internet Protocol, DARPA INTERNET PROGRAM Protocol Specification. Technical Report RFC 791, 1981.
- [67] IETF. Transmission Control Protocol. Technical Report RFC 675 & RFC 793, 1981.
- [68] IETF. User Datagram Protocol. Technical Report RFC 768, 1981.

- [69] Ingeniars.com. SpaceWire / SpaceFibre Network Model. URL: https://www.ingeniars.com/in_product/spacewire-spacefibre-network-model/.
- [70] International Organisation for Standardization. ISO/IEC 7498-1:1994 Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. Technical report, 2000. URL: <https://www.iso.org/standard/20269.html>.
- [71] ISO 11898-2. Road vehicles – Controller area network (CAN). Technical report, 2016.
- [72] Mirko Jakovljevic, Arjan Geven, Natasa Simanic-John, and Derya Mete Saatci. Next-Gen Train Control / Management (TCMS) Architectures: “Drive-By-Data” System Integration Approach. In *ERTS 2018*, 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France, 2018. URL: <https://hal.archives-ouvertes.fr/hal-02156252>.
- [73] Hermann Kopetz. Event-triggered versus time-triggered real-time systems. In *Proceedings of the International Workshop on Operating Systems of the 90s and Beyond*, page 87–101. Springer-Verlag, 1991.
- [74] Ulf Kulau, Juergen Herpel, Ran Qedar, Patrick Rosenthal, Joachim Krieger, Friedrich Schoen, and Ivan Masar. Towards modular and scalable on-board computer architecture. *Information technology*, pages 185–197, 2021. URL: <http://hdl.handle.net/11420/10732>.
- [75] J.C. Laprie. *Guide de la sûreté de fonctionnement*. 1996.
- [76] Ana Larrañaga, M. Carmen Lucas-Estañ, Imanol Martinez, Iñaki Val, and Javier Gozalvez. Analysis of 5g-tsn integration to support industry 4.0. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1111–1114, 2020. doi:10.1109/ETFA46521.2020.9212141.
- [77] Sune Mølgaard Laursen, Paul Pop, and Wilfried Steiner. Routing optimization of avb streams in tsn networks. *ACM Sigbed Review*, 13(4):43–48, 2016.
- [78] Xiaoting Li, Jean-Luc Scharbarg, Christian Fraboul, and Frédéric Ridouard. Existing offset assignments are near optimal for an industrial AFDX network. In *Proc. of the 10th International Workshop on Real-time Networks (RTN 2011)*, Porto, Portugal, July 5th 2011.
- [79] H. Lim, D. Herrscher, and F. Chaari. Performance comparison of ieee 802.1q and ieee 802.1 avb in an ethernet-based in-vehicle network. In *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, volume 1, pages 1–6, 2012.
- [80] H. Lim, D. Herrscher, L. Völker, and M. J. Watl. Ieee 802.1as time synchronization in a switched ethernet based in-car network. In *2011 IEEE Vehicular Networking Conference (VNC)*, pages 147–154, 2011. doi:10.1109/VNC.2011.6117136.
- [81] Lucia Lo Bello and Wilfried Steiner. A perspective on ieee time-sensitive networking for industrial communication and automation systems. *Proceedings of the IEEE*, 107(6):1094–1120, 2019. doi:10.1109/JPROC.2019.2905334.
- [82] Lisa Maile, Kai-Steffen Hielscher, and Reinhard German. Network calculus results for tsn: An introduction. In *Proc. of the Information Communication Technologies Conference (ICTC 2020)*, pages 131–140. IEEE, 2020.

- [83] CPLEX User's Manual. Ibm ilog cplex optimization studio. *Version*, 12:1987–2018, 1987.
- [84] Cedric Mauclair, Marina Guitierrez, Jörn Migge, and Nicolas Navet. Do we really need TSN in Next-Generation Helicopters ? Insights from a Case-Study. In *Proc. of the 40th Int. Conference on Digital Avionics System Conference (DASC)*, 2021.
- [85] Dorin Maxim and Ye-Qiong Song. Delay analysis of avb traffic in time-sensitive networks (tsn). In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS '17*, page 18–27, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3139258.3139283.
- [86] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, 2018.
- [87] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the performance and configuration of avb and tsn in automotive ethernet networks. *Proc. Embedded Real-Time Software and Systems (ERTS 2018)*, 2018.
- [88] Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury. Tsn algorithms for large scale networks: A survey and conceptual comparison, 2019. arXiv:1905.08478.
- [89] Nicolas Navet, Jörn Migge, Josetxo Villanueva, and Marc Boyer. Pre-Shaping Bursty Transmissions under IEEE802.1Q as a Simple and Efficient QoS Mechanism. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 11(3), April 2018. URL: <https://hal.archives-ouvertes.fr/hal-02468313>, doi:10.4271/2018-01-0756.
- [90] Nicolas Navet and Françoise Simonot-Lion. In-vehicle communication networks - a historical perspective and review. In Richard Zurawski, editor, *Industrial Communication Technology Handbook, Second Edition*. CRC Press Taylor and Francis, 2013. URL: <https://hal.inria.fr/hal-00876524>.
- [91] Olivier Notebaert, Giuseppe Montano, Thierry Planche, Clement Pruvost, Franck Wartel, Andreas Schuttauf, Hans-Jurgen Herpel, Christophe Honvault, and David Jameux. Towards spacewire-2: Space robotics needs: Spacewire missions and applications, long paper. pages 1–9, 10 2016.
- [92] Maryam Pahlevan and Roman Obermaisser. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. pages 337–344, 09 2018. doi:10.1109/ETFA.2018.8502515.
- [93] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM SIGBED Review*, 16:15–20, 02 2019. doi:10.1145/3314206.3314208.
- [94] Raphaël Polonowski and Rémi Clavier. Ariane launchers digital engineering: stakes and challenges. In *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–5, 2019. doi:10.1109/MECO.2019.8760090.
- [95] Paul Pop, Michael Lander Raagaard, Marina Gutierrez, and Wilfried Steiner. Enabling fog computing for industrial automation through time-sensitive networking (tsn). *IEEE Communications Standards Magazine*, 2(2):55–61, 2018.

- [96] Francisco Pozo, Guillermo Rodriguez-Navas, and Hans Hansson. Schedule reparability: Enhancing time-triggered network recovery upon link failures. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 147–156, 2018. doi:10.1109/RTCSA.2018.00026.
- [97] Clement Pruvost, Thierry Planche, Olivier Notebaert, Alain Rossignol, Hans-Jurgen Herpel, and Andreas Schuttauf. Ethernet for space: an enabler for next-generation avionics. In *2016 Data System In Aerospace (DASIA)*, 05 2016.
- [98] Guy Pujolle. *Les Reseaux*, volume 9, pages 96–97. Eyrolles, 2018-2020.
- [99] RealTime-at-Work. RTaW-Pegase: Modeling, Simulation and automated Configuration of real-time communication architectures. <https://www.realtimeatwork.com/software/rtaw-pegase>, 2021.
- [100] Niklas Reusch, Luxi Zhao, Silviu S. Craciunas, and Paul Pop. Window-based schedule synthesis for industrial iee 802.1qbv tsn networks. In *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, pages 1–4, 2020. doi:10.1109/WFCS47810.2020.9114414.
- [101] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.
- [102] SAE. AS6802 - Time-Triggered Ethernet. Technical Report SAE AS6802. URL: <https://www.sae.org/standards/content/as6802/>.
- [103] Jorge Sanchez-Garrido, Beatriz Aparicio, José Gabriel Ramírez, Rafael Rodriguez, Mariasole Melara, Lorenzo Cercós, Eduardo Ros, and Javier Diaz. Implementation of a time-sensitive networking (tsn) ethernet bus for microlaunchers. *IEEE Transactions on Aerospace and Electronic Systems*, 57(5):2743–2758, 2021. doi:10.1109/TAES.2021.3061806.
- [104] Stefan Schneelee and Fabien Geyer. Comparison of iee avb and afdx. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 7A1–1. IEEE, 2012.
- [105] R. Serna Oliver, S. S. Craciunas, and W. Steiner. Iee 802.1qbv gate control list synthesis using array theory encoding. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 13–24, 2018.
- [106] Aakash Soni, Jean-Luc Scharbarg, and Jérôme Ermont. Quantum assignment for QoS-aware AFDX network with Deficit Round Robin. In *27th International Conference on Real Time Networks and Systems - RTNS 2019*, pages 70–79, Toulouse, France, November 2019. ACM Press. URL: <https://hal.archives-ouvertes.fr/hal-02965546>, doi:10.1145/3356401.3356421.
- [107] Aakash Soni, Jean-Luc Scharbarg, and Jérôme Ermont. Efficient configuration of a QoS-aware AFDX network with Deficit Round Robin. In *18th IEEE International Conference on Industrial Informatics*, Warwick, United Kingdom, July 2020. URL: <https://hal.archives-ouvertes.fr/hal-02965556>.
- [108] Star Dundee. SpaceWire User’s Guide. Technical report, University of Dundee, 2012. URL: https://www.star-dundee.com/wp-content/star_uploads/general/SpaceWire-Users-Guide.pdf.

- [109] T. Steinbach, H. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz. Tomorrow's in-car interconnect? a competitive evaluation of IEEE 802.1 AvB and time-triggered Ethernet (AS6802). In *2012 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, 2012.
- [110] Wilfried Steiner. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. pages 375–384, 11 2010. doi:10.1109/RTSS.2010.25.
- [111] Wilfried Steiner. Synthesis of static communication schedules for mixed-criticality systems. In *Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, ISORCW '11*, page 11–18, USA, 2011. IEEE Computer Society. doi:10.1109/ISORCW.2011.12.
- [112] Wilfried Steiner and Günther Bauer. Ethernet for Space Applications: TTEthernet. In *Second International Spacewire Conference*, 2008.
- [113] Wilfried Steiner, Peter Heise, and Stefan Schneele. Recent IEEE 802 developments and their relevance for the avionics industry. In *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, pages 2A2-1–2A2-12, 2014. doi:10.1109/DASC.2014.6979419.
- [114] Chakkaphong Suthaputthakun, Zhili Sun, Christoforos Kavadias, and Philippe Ricco. Performance analysis of AFDX switch for space onboard data networks. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1714–1727, 2016. doi:10.1109/TAES.2016.150304.
- [115] Daniel Thiele, Philip Axer, Rolf Ernst, and Jan R. Seyler. Improving formal timing analysis of switched Ethernet by exploiting traffic stream correlations. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis, CODES '14*, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2656075.2656090.
- [116] Daniel Thiele and Rolf Ernst. Formal worst-case performance analysis of time-sensitive Ethernet with frame preemption. In *21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA '16*, pages 1–9, 2016.
- [117] Daniel Thiele and Rolf Ernst. Formal worst-case performance analysis of time-sensitive Ethernet with frame preemption. In *21st IEEE International Conference on Emerging Technologies and Factory Automation, ETFA '16*, pages 1–9, 2016.
- [118] Daniel Thiele and Rolf Ernst. Formal worst-case performance analysis of time-sensitive Ethernet with frame preemption. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–9, 2016. doi:10.1109/ETFA.2016.7733740.
- [119] TTTech. Time-Triggered Ethernet – A Powerful Network Solution for Multiple Purpose. Technical report. URL: https://www.tttech.com/wp-content/uploads/TTTech_TTEthernet_Technical-Whitepaper.pdf.
- [120] Marek Vlk, Katerina Brejchova, Zdenek Hanzalek, and Siyu Tang. Large-scale periodic scheduling in time-sensitive networks. *Computers and Operations Research*, 137:105512, 2022. URL: <https://www.sciencedirect.com/science/article/pii/S0305054821002549>, doi:<https://doi.org/10.1016/j.cor.2021.105512>.
- [121] E. Webb. Ethernet for space flight applications. In *Proceedings, IEEE Aerospace Conference*, volume 4, pages 4–4, 2002. doi:10.1109/AERO.2002.1036905.

- [122] www.ruag.com. Ruag space and tttech partner to provide electronics for fast, reliable data transfer based on ttethernet for nasa lunar gateway. <https://www.ruag.com/fr/node/1463>, 2019.
- [123] Jinli Yan, Wei Quan, Xuyan Jiang, and Zhigang Sun. Injection time planning: Making cqf practical in time-sensitive networking. In *Proc. of the 39th IEEE Conference on Computer Communications (INFOCOM 2020)*, pages 616–625, 2020. doi:10.1109/INFOCOM41043.2020.9155434.
- [124] Lin Zhao, Feng He, Ershuai Li, and Jun Lu. Comparison of time sensitive networking (tsn) and ttethernet. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE, 2018.
- [125] Luxi Zhao, Paul Pop, and Silviu S. Craciunas. Worst-case latency analysis for ieee 802.1qbv time sensitive networks using network calculus. *IEEE Access*, 6:41803–41815, 2018. doi:10.1109/ACCESS.2018.2858767.