THÈSE

Pour l'obtention du Grade de

DOCTEUR DE L'UNIVERSITÉ DE POITIERS

(ÉCOLE SUPÉRIEURE d'INGÉNIEURS de POITIERS)
(Diplôme National - Arrêté du 25 mai 2016)

École Doctorale 651 :
MIMME – Mathématiques – Informatique – Matériaux – Mécanique - Energétique

Secteur de Recherche :
Informatique et applications

Présentée par :

Richard GARREAU

****

Contributions à l'analyse pire cas des réseaux embarqués temps réel

****

Directeur de Thèse :
Pascal RICHARD

****

Soutenue le 28 novembre 2024

devant la Commission d'Examen

****

JURY

Audrey QUEUDET, Maître de Conférences HDR, LS2N/Nantes Université.....Rapportrice
Jean-Luc SCHARBARG, Professeur, IRIT/ENSEEIHT, Toulouse .....Rapporteur
Ye-Qiong SONG, Professeur, LORIA/Université de Lorraine.....Examinateur
Henri BAUER, Maître de Conférences, LIAS/ENSMA, Poitiers.....Examinateur

Pascal RICHARD, Professeur, LIAS/Université de Poitiers
Directeur de Thèse

Frédéric RIDOUARD, Maître de Conférences HDR, LIAS/ENSMA, Poitiers
Co-encadrant de Thèse

REMERCIEMENTS

Probablement la partie la plus compliquée à rédiger de ce manuscrit. Si j'avais voulu rendre hommage de manière complète à toutes les personnes qui m'ont soutenu et encouragé, il aurait probablement fallu que je rédige un autre document d'une centaine de pages. Je vais donc synthétiser mes remerciements dans la section suivante.

Je tiens à remercier mon équipe d'encadrement à commencer par mon directeur de thèse Pascal, qui a été de bons conseils et de bonne humeur pendant ma rédaction et ma préparation de soutenance. Frédéric qui a su toujours tenir la barre d'un point de vue scientifique et qui a donné une direction efficace à ma thèse (même si ce n'est pas écrit dans la norme). Henri, qui a été de bons conseils sur les sujets techniques et scientifiques et avec qui j'apprécie beaucoup discuter.

Je remercie Pr. Song, Pr. Scharbarg ainsi que Dr. Queudet d'avoir accepté d'évaluer les différents travaux de ma thèse.

Je remercie aussi les personnes du laboratoire avec qui j'ai beaucoup aimé échanger. Bénédicte, avec mes excuses vaseuses pour grignoter les friandises de son bureau. Emmanuel, avec qui je pense avoir suffisamment médité pour le reste de ma vie. La fondation Chardin & Baron pour sa bonne humeur et ses conseils. Un grand merci à tous les autres membres du laboratoire pour votre aide et votre bonne humeur ! Merci aussi à tous mes collègues doctorants avec qui j'ai pris plaisir à partager mes journées : Luc, Thomas, Quentin, Boumediene, Ali, Nour, Matheus, Maxime.

Merci à l'ensemble du département d'informatique de l'université de Poitiers que j'ai eu dans un premier temps comme enseignants et dans un second temps comme collègues. Merci aux amis de Bernard, qui me connaissent depuis le berceau et qui ont été d'un grand soutien pendant ces années de thèse.

Un grand merci à mes très chers amis qui ont été à mes côtés pendant ma thèse. Benjamin et Valentin qui m'ont suivi depuis l'ensemble de mes études universitaires. Mon groupe d'amis que j'ai depuis le lycée, vous êtes trop nombreux pour que je vous nomme individuellement, mais je pense à chacun d'entre vous. L'équipe KP composée de Gabriel, Jorge, Laura et avec laquelle j'ai beaucoup aimé passer du temps. Merci à Julia, Louise et Maxime pour ces parties endiablées de coinche et ces moments uniques. Merci à David, pour être toi-même et pour ta gentillesse. Merci à Alexandre pour sa relecture, son amitié durable et sincère qui j'espère continuera de perdurer.

衷心感谢我的母亲给予我坚定不移和无条件的支持。
非常感谢你每天的帮助。

Merci à toi Karine pour ton soutien, ta bonne humeur et la joie que tu m'apportes. Et pour finir merci à Bernard pour tout ce qu'il a fait pour moi. Je suis sûr que tu serais fier du chemin que j'ai parcouru.

Sans vous, l'accomplissement de mes travaux de thèse n'aurait pas pu aboutir ; vous avez tous été moteur du manuscrit qui suit.

INTRODUCTION

CONTEXTE

L'augmentation exponentielle des systèmes numériques depuis les années 1980 dans les milieux industriels, du transport et grand public a conduit à une augmentation de leur complexité. Les systèmes spécialisés ont pu être regroupés temporairement au sein de calculateurs monolithiques dans lequel une entité est responsable de l'ensemble des calculs. Cette approche s'est rapidement révélée limitante compte tenu du nombre croissant de fonctions qui leur sont dédiées. Pour résoudre ce problème, les fonctions sont désormais réparties sur plusieurs calculateurs qui s'échangent leurs résultats et agissent sur les différents actionneurs du système via un réseau informatisé. Pour faciliter ces échanges et réduire les coûts de développement et de production, des normes de réseau sont disponibles, telles que l'Ethernet [54], l'*Avionic Full Duplex Switched Ethernet* (AFDX) [4] ou *Time Sensitive Networking* (TSN) [106].

Certains de ces systèmes sont critiques et peuvent entraîner des incidents graves impliquant des vies humaines en cas de défaillance, comme le système de freinage d'une voiture. Pour ces systèmes, il est nécessaire de fournir une preuve du bon fonctionnement des fonctions sur les différents calculateurs et sur les réseaux. Cela peut prendre la forme de garanties de déterminisme. Par exemple, qu'une tâche ou qu'un message dans un réseau soit transmis dans un délai borné ou qu'aucune trame ne puisse être perdue dans le réseau.

PROBLÉMATIQUES

Dans ce manuscrit, nous aborderons deux problèmes distincts dans les réseaux temps réel.

Dans un système temps réel distribué, il est difficile de relier tous les calculateurs directement les uns aux autres en raison de la complexité d'installation et du coût. Pour simplifier la mise en place des réseaux, des équipements intermédiaires appelés commutateurs réseaux sont installés. Les commutateurs réseaux sont des dispositifs physiques chargés d'écouter leurs entrées pour récupérer des trames arrivantes et de les retransmettre sur le port de sortie le plus adéquat. Le seul endroit où une trame peut être perdue est dans les ports de sortie des commutateurs, plus précisément dans les files d'attente. Cette affirmation est valable lorsque nous supposons qu'aucune défaillance physique ne peut survenir dans les commutateurs et que les liens physiques intersystèmes ne peuvent pas engendrer de collisions. En effet, si la quantité de trames excède la capacité de stockage d'une file d'attente, des trames peuvent être supprimées. Pour éviter ce problème, il est nécessaire de déterminer la quantité maximale de trames afin de s'assurer qu'aucune trame ne sera supprimée.

Certains systèmes permettent de décaler la transmission des trames pour réduire les délais d'un point à un autre du réseau. Nous appelons ces systèmes des systèmes à *offset-free*. L'objectif idéal de ces systèmes est de vérifier qu'aucune trame ne cause de congestion avec une autre, garantissant ainsi un temps de traversée minimal à travers le réseau. Des méthodes permettant d'atteindre

cet objectif ont été publiées par la communauté scientifique. Néanmoins, nous avons remarqué que certaines hypothèses prises dans ces méthodes peuvent conduire à ignorer certaines trames susceptibles de provoquer une congestion. La question qui se pose est donc : comment s'assurer que toutes les trames sont toujours comptabilisées et existe-t-il un moyen d'affecter des décalages qui ne soient pas soumis à ce problème de comptage ?

STRUCTURE DU MANUSCRIT

Ce manuscrit est divisé en deux parties : une décrivant le contexte et l'état de l'art dans les chapitres 1, 2 et 3, et l'autre décrivant les contributions dans les chapitres 4 et 5.

Dans le chapitre 1, nous décrirons ce qu'est un système temps réel et les éléments techniques nécessaires pour la compréhension des travaux décrits dans ce manuscrit. Nous agrémenterons ce chapitre d'exemples de normes de réseaux utilisées dans l'industrie et présenterons un cas d'étude représentatif.

Dans le chapitre 2, nous décrirons les normes *Time Sensitive Networking* qui visent à apporter les notions temps réel dans la norme *Ethernet*. Nous décrirons également le modèle de réseau que nous utilisons.

Dans le chapitre 3, nous aborderons l'état de l'art des problématiques que nous étudions dans les réseaux temps réel.

Dans le chapitre 4, nous étudierons le premier problème de ce manuscrit concernant le dimensionnement des buffers réseaux dans les ports de sortie des commutateurs réseaux. Nous chercherons à quantifier le nombre maximal de trames pouvant être gérées.

Dans le chapitre 5, nous étudierons le second problème en prouvant l'existence de problèmes de cyclicité d'ordonnancement dans divers travaux concernant la transmission de trames dans les ports de sortie des commutateurs réseaux. Ces problèmes peuvent être atténués en s'assurant que les trames soient envoyées en respectant un certain ensemble de contraintes. Nous proposerons également une méthode permettant d'ordonnancer le trafic réseau sans être soumis à ces contraintes.

TABLE DES MATIÈRES

I Contexte et état de l'art

1 Contexte 13
- 1.1 Systèmes temps réel 13
- 1.2 Les systèmes temps réel distribués 14
- 1.3 Réseaux temps réel embarqués 15
- 1.4 Modèle OSI et *Ethernet* 15
  - 1.4.1 Modèle OSI 15
  - 1.4.2 *Ethernet* 17
- 1.5 Exemples de normes utilisés dans l'industrie 18
  - 1.5.1 L'approche automobile 18
  - 1.5.2 L'approche aéronautique et aérospatiale 19
  - 1.5.3 Industrie 4.0 et 5.0 19
  - 1.5.4 Récapitulatif des possibilités des normes 20
- 1.6 Cas d'étude représentatif : Orion 20
- 1.7 Conclusion 21

2 Réseaux TSN 23
- 2.1 Histoire des normes TSN 23
- 2.2 Données et topologie d'un réseau TSN 24
- 2.3 Principe de fonctionnement des *bridges* et *End-Stations* TSN 26
- 2.4 Ports de sortie (ES/Bridge) 27
- 2.5 Qualités de service dans TSN 28
  - 2.5.1 Strict Priority 28
  - 2.5.2 Credit-based Shaper 29
  - 2.5.3 Time Aware Shaper 31
- 2.6 Conclusion et hypothèses définies 33

3 État de l'art 35
- 3.1 Évaluation des délais de bout en bout 35
  - 3.1.1 Problématique 35
  - 3.1.2 Simulation 37
  - 3.1.3 Network Calculus 37
  - 3.1.4 Compositional Performance Analysis 39
  - 3.1.5 La méthode des intervalles éligibles 39
  - 3.1.6 Forward end-to-end delay Analysis (FA) 40
- 3.2 Dimensionnement des files d'attente dans les commutateurs réseaux 41
  - 3.2.1 Problématique 41
  - 3.2.2 Méthodes 41
- 3.3 Génération de tables de configuration TAS 42

3.3.1 Problématique 42
3.3.2 Méthodes optimales 43
3.3.3 Méthodes heuristiques 44
3.4 Conclusion 45

II Contributions
4 Le dimensionnement des buffers de commutateurs en nombre de trames 49
4.1 Introduction 49
4.2 Motivations 50
4.3 Modèle réseau 51
4.4 Dimensionnement des files d'attente en nombre de trames 52
4.4.1 Principe 52
4.4.2 Trames entrantes 53
4.4.3 Trames sortantes 54
4.4.4 Effet de sérialisation 55
4.4.5 Calcul de NbF^h 57
4.5 Expérimentations 60
4.5.1 Étude de cas 60
4.5.2 Configuration industrielle 62
4.6 Conclusion 63
5 Problèmes d'ordonançabilité dans les réseaux dirigés par le temps 67
5.1 Notations et état de l'art 68
5.1.1 Approches d'ordonnancement par offsets et notations 68
5.1.2 Les travaux fondamentaux du domaine de l'ordonnancement *offset-free* réseau (Steiner, et al. [100]) 70
5.2 Problèmes avec les hyperpériodes 71
5.2.1 Cyclicité et absence de congestion 71
5.2.2 Importance de la contrainte sur trames 74
5.3 Cyclicité et acyclicité dans les ordonnancements avec offsets 75
5.3.1 Résultats fondamentaux sur la cyclicité 75
5.4 Une heuristique supportant la congestion : GCD# 78
5.4.1 Ordonnancement d'un seul nœud 79
5.4.2 GCD# 84
5.5 Expérimentations 87
5.5.1 Pousser GCD# à ses limites 88
5.5.2 GCD# comparé à SMT 90
5.6 Conclusion et perspectives 93

Conclusion et Perspectives

TABLE DES FIGURES

Figure 1.1 Boucle de rétroaction entre le système temps réel et le procédé 14
Figure 1.2 Aperçu de différents types de topologies utilisés couramment lors de la distribution de systèmes 16
Figure 1.3 Modèle OSI 17
Figure 1.4 Photo de la navette spatiale Orion lors de la mission *Artemis I* [82] 21
Figure 2.1 Description des domaines d'action du groupe de travail TSN et des normes produites associés[53] 24
Figure 2.2 Visualisation d'une trame *Ethernet* TSN 25
Figure 2.3 Description de la topologie d'un réseau TSN standard 26
Figure 2.4 Représentation d'un *bridge* TSN 27
Figure 2.5 Représentation d'un port de sortie TSN 28
Figure 2.6 Représentation d'un port de sortie TSN utilisant FP/FIFO 29
Figure 2.7 Représentation d'un port de sortie TSN utilisant la politique de service CBS 30
Figure 2.8 Description du fonctionnement de CBS sur un exemple 31
Figure 2.9 Exemple de fonctionnement du Time Aware Shaper (à l'instant $t2$) 33
Figure 3.1 Vocabulaire relatif aux différents temps de transmissions dans un réseau 36
Figure 3.2 Courbes d'arrivée γ et de service β 38
Figure 3.3 Combinaison des courbes d'arrivées et de service 38
Figure 3.4 Comparaison du dimensionnement des files d'attente en bits et en trames 42
Figure 3.5 Diagramme de Gantt représentant la relation entre l'arrivée d'une trame et sa transmission quand cette dernière dispose d'un offset 43
Figure 3.6 Diagramme de Gantt montrant la différence entre un écoulement contigu et non-contigu 44
Figure 3.7 Représentation du principe de fonctionnement d'une métaheuristique 45
Figure 4.1 Le retard accumulé maximal en nombre de bits et de trames ne coïncide pas toujours. 51
Figure 4.2 Modèle d'un nœud h avec n liens entrants et un débit de sortie r_h. 51
Figure 4.3 Un exemple de configuration de réseau. 52
Figure 4.4 Scénarios d'arrivée dans une fille d'attente FIFO. 53
Figure 4.5 Temps d'arrivée minimaux des trames de $v_i \in \Gamma_h$ dans une fille d'attente utilisant une politique de service conservative h. 54
Figure 4.6 Optimalité de LPT pour minimiser le nombre de trames transmises. 55

Figure 4.7 Scénario d'arrivée maximisant le nombre de trames entrantes à un temps d'inactivité x. 56
Figure 4.8 Courbes d'arrivée et de service du nœud h_4. 61
Figure 4.9 Amélioration des performances du BDF par rapport à la méthode naïve dans un cas de test industriel. 63
Figure 5.1 Partie cyclique et acyclique sur les simulations des cas d'exemple 1 et 2 avec les paramètres de la table 5.1 73
Figure 5.2 Partie cyclique et acyclique sur les simulations des cas d'exemple 3 avec les paramètres de la table 5.1 73
Figure 5.3 Cercle PGCDT_i, T_j, avec une représentation de v_i et v_j 80
Figure 5.4 Cercle modulaire de taille Ω avec les flux du cas d'exemple 83
Figure 5.5 Distribution des trames divisées en cycles 83
Figure 5.6 Topologie utilisée dans le cas synthétique pour évaluer les collisions de GCD# 88
Figure 5.7 Simulation du cas de test 1 décrit dans le Tableau 5.2 89
Figure 5.8 Simulation du cas de test 2 décrit dans le Tableau 5.2 89
Figure 5.9 Topologie d'Orion d'après [113] 91
Figure 5.10 Comparaison des performances entre GCD# et SMT sur une execution pour 100 flux 92
Figure 5.11 Comparaison des performances entre GCD# et SMT sur une execution pour 200 flux 92

Première partie

CONTEXTE ET ÉTAT DE L'ART

1

CONTEXTE

Dans ce chapitre, nous allons introduire le contexte général des travaux présentés dans ce manuscrit. Nous aborderons : les systèmes temps réel distribués, leurs réseaux, la structure de base de ces derniers, l'interconnexion et différentes normes réseaux. Nous conclurons ce chapitre par un cas d'étude représentatif qui servira de base pour plusieurs évaluations expérimentales conduites dans ce manuscrit.

1.1 SYSTÈMES TEMPS RÉEL

Un système informatique est une entité automatisée dédiée au traitement ou au stockage de données numériques. Les avancées technologiques dans le domaine de l'électronique ces dernières décennies ont permis d'obtenir des systèmes toujours plus puissants et plus économes en énergie. Ces avancées permettent d'améliorer l'efficacité des systèmes ou d'ouvrir de nouvelles possibilités dans le domaine des transports, services, santé ou encore l'industrie manufacturière. Nous pouvons citer les usines automatiques et reprogrammables en fonction de la demande ou encore des véhicules autonomes.

Les systèmes embarqués sont une classe de système informatique se trouvant au sein même du système qu'ils contrôlent. Prenons l'exemple d'un système de sûreté du personnel d'une mine. Celle-ci dispose de capteurs aidant à évaluer les conditions de sécurité et les actionneurs sont à l'intérieur de celle-ci, alors que le système informatique est à l'extérieur. Ce système n'est pas embarqué. À l'inverse, le système de freinage d'urgence d'une voiture est au sein même du véhicule qu'il contrôle, donc il est embarqué. En règle générale, un système embarqué a de plus fortes contraintes que ce soit en espace, capacité de calculs ou de contraintes énergétiques.

Un sous-ensemble des systèmes informatiques (qu'ils soient embarqués ou non) sont dits *temps réel*. C'est-à-dire, « que le système ne dépend pas uniquement des résultats logiques issus du calcul, mais également de l'instant auxquels ceux-ci ont été produits » (Stankovic) [19][99]. Ce sont les **Systèmes (embarqués) temps réel**. La notion de temps dans ce contexte est subjective, un système doit être suffisamment réactif par rapport au procédé contrôlé [24]. Par exemple, l'observation de réactions chimiques par un système peuvent être longues (échelle d'heures), mais doivent être faites à des horaires précis. Dans ce contexte, le respect des contraintes temporelles n'équivaut pas à la recherche de performance, mais à la correction du système.

Le système temps réel (l'aspect logique) est informé par l'éventuel changement d'état du procédé (l'aspect physique) par le biais de capteurs. Cette dynamique est schématisée dans la Fig 1.1 proposant une représentation de la place du système temps réel, le procédé, les capteurs, les actionneurs ainsi que la boucle de rétroaction pour contrôler le système.

À titre d'exemple, considérons un système de régulation de vitesse d'une voiture ; celui-ci est conçu pour maintenir une vitesse cible prédéfinie. Le système cherche à atteindre cette vitesse et,

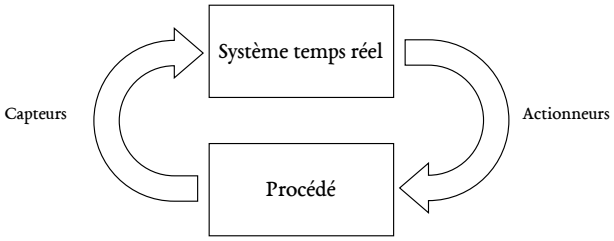

FIGURE 1.1 – Boucle de rétroaction entre le système temps réel et le procédé

une fois atteinte, la maintiendra. Ce contrôle de l'accélération se fait au travers d'une boucle de rétroaction entre le système et le procédé : le système temps réel ajuste continuellement le procédé (ici, la voiture) en fonction des paramètres utilisateur et des métriques de vitesse obtenus par les capteurs. Si le système ne réagit pas suffisant vite, cela peut engendrer des cas où la vitesse peut être maintenue alors qu'un ordre d'arrêt a été émis et potentiellement causer un accident. Ainsi, il y a une interaction constante entre le système temps réel et le procédé. Les différentes actions du calculateur doivent être faites à temps pour réguler la vitesse et assurer la sécurité.

Il y a deux catégories de systèmes temps réel, chacune caractérisée par l'impact du non-respect des contraintes temporelles :

— **systèmes temps réel souples** : le manquement d'une contrainte temporelle peut avoir des répercussions sur la qualité du service, sans impact critique. Par exemple, dans une voiture, une latence excessive dans le changement de station de l'autoradio diminue la qualité de l'expérience utilisateur sans induire de risques fonctionnels.
— **systèmes temps réel critiques/durs** : le manquement d'une contrainte temporelle peut avoir des conséquences graves, y compris des risques pour la vie humaine. Par exemple, le régulateur de vitesse, les cas de blocages des systèmes peuvent conduire à des accidents avec des conséquences dramatiques.

1.2 LES SYSTÈMES TEMPS RÉEL DISTRIBUÉS

Une architecture d'un système embarqué temps réel centré autour d'un unique calculateur pose plusieurs problèmes :

— **Limite de puissance de calcul** : les besoins de calcul dépassent les capacités d'un unique calculateur;
— **Difficultés de mise en œuvre** : les capteurs/actionneurs peuvent être répartis dans le système. Une architecture centralisée est inadaptée pour mettre en œuvre le système de contrôle.

Les systèmes temps réel décentralisés sont appelés : **systèmes temps réel distribués**. Une solution pour faire face à ces problèmes est de distribuer les fonctions sur plusieurs systèmes interconnectés. Cette démarche est appelée la distribution [81]. Pour mettre en place cette distribution, il convient de disposer de mécanismes de communication entre les différents calculateurs, appelé réseau. Il est envisageable de former un réseau, en connectant tous les systèmes en point-à-point, c'est-à-dire qu'un lien sert uniquement pour un seul capteur/actionneur. Néanmoins, cette approche est assez coûteuse et complexe, car le nombre de liens physiques requis est proportionnel

aux actionneurs/capteurs et au nombre de systèmes informatisés présents. Par conséquent, des approches pragmatiques permettant le partage des liens pour différents besoins ont été définies. Nous aborderons quelques-unes d'entre elles dans la section suivante.

1.3 RÉSEAUX TEMPS RÉEL EMBARQUÉS

Afin d'organiser les réseaux, plusieurs topologies réseaux ont été proposées [45]. Ces topologies présentent chacune des avantages et des inconvénients en termes du coût d'installation, maintenance ou encore de performances du réseau.

— **Réseau en bus**, Fig.1.2.a : cette topologie consiste en un unique lien physique partagé par plusieurs systèmes. Cette approche est peu coûteuse, car un seul lien physique est déployé. La topologie est utilisée dans des applications industrielles. Néanmoins, un des inconvénients sont les potentielles collisions sur le médium partagé lorsque deux systèmes émettent simultanément.
— **Réseau en étoile**, Fig.1.2.b : un réseau en étoile est articulé autour d'un système central chargé de transférer les données reçues au bon destinataire. Cette topologie présente l'avantage d'une gestion centralisée au sein d'un seul nœud ;
— **Réseau maillé**, Fig.1.2.c : le réseau maillé est une catégorie de réseau avec les dispositifs communicants interconnectés directement les uns aux autres. Si un dispositif reçoit un message dont il n'est pas le destinataire, il retransmettra le message vers un autre dispositif plus proche du destinataire final. Cette architecture de réseau propose une grande redondance, mais peut être compliquée à gérer/configurer.

Nous noterons que dans les topologies décrites précédemment, la topologie en étoile utilise des éléments réseaux intermédiaires, les commutateurs ou les routeurs réseaux.

Afin de structurer les communications dans ces topologies, un modèle d'interconnexion existe et sert de référence pour tous les réseaux construits. Nous allons aborder dans la section suivante le modèle *Open Systems Interconnection* (OSI) ainsi que la norme réseau *Ethernet* très utilisé dans le domaine grand public.

1.4 MODÈLE OSI ET *ETHERNET*

Dans cette section, nous allons décrire le modèle OSI et la norme *Ethernet* utilisé dans le domaine grand public.

1.4.1 *Modèle OSI*

Le modèle *Open Systems Interconnection (OSI)* [63] est un modèle standardisé d'interconnexion des systèmes ouverts conçue par l'Organisation Internationale de Normalisation. Celle-ci spécifie la communication réseau par la succession de différentes étapes effectuées par un ou plusieurs composants logiciels ou matériel. La spécification est théorique, mais décrit le fonctionnement de la communication depuis l'instant de sa génération par une application informatique jusqu'au

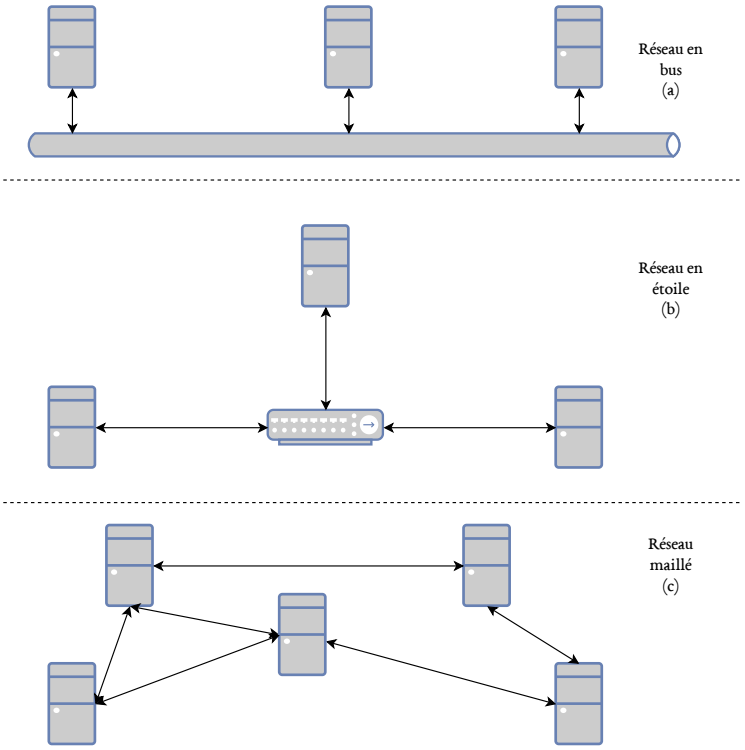

FIGURE 1.2 – Aperçu de différents types de topologies utilisés couramment lors de la distribution de systèmes

moment où elle sera transmise physiquement sur le médium. Ces étapes sont appelées couches et sont au nombre de sept. Plus le numéro de la couche est petit, plus celle-ci est proche du matériel.

Nous allons faire une description de ces couches.

Au niveau physique :

1. Physique : la couche spécifie la façon dont les données sont transmises sur le support physique. Des bits sont échangés ;
2. Liaison : la couche fournit les moyens d'établissement de connexion et la maintenance de celle-ci entre les entités. À ce niveau, les entités échangent des **trames**. Un dispositif qui s'arrête à ce niveau est un commutateur ;
3. Réseau : la couche se charge de faire transiter les données jusqu'à son destinataire en permettant l'indépendance des protocoles qui pourraient être engagés dans les couches précédentes. À ce niveau, les entités échangent des **paquets**. Un dispositif qui s'arrête à ce niveau est un routeur ;

Au niveau de chaque entité communicante ou client avec un programme applicatif du réseau :

4. Transport : cette couche permet de rendre transparente la manière dont les données sont acheminées dans le réseau (p.ex : la segmentation des données si elles sont trop grandes pour le protocole utilisé). Elle se charge aussi d'optimiser les coûts de communications pour les différentes couches sessions. Des segments sont échangés ;
5. Session : cette couche assure que les communications entre les entités soient organisées et synchronisées. Dans le cadre de l'envoi de données sans connexion préalable, elle ne sert

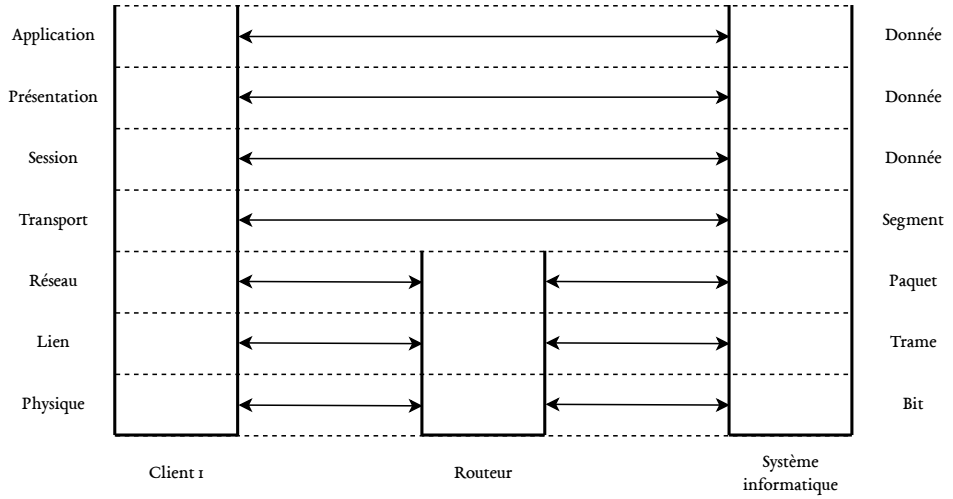

FIGURE 1.3 – Modèle OSI

qu'à retransmettre les informations entre la couche *Transport* vers les autres plus hautes. À partir de cette couche, des données sont échangées ;

6. Présentation : cette couche permet de mettre en forme les données de telle sorte à ce que l'entité émettrice et réceptrice aient une structuration des données en commun ;
7. Application : cette couche constitue le point d'entrée du modèle OSI pour les applications qui en tirent parti.

Les couches OSI communiquent toujours avec des couches OSI de même niveau dans les différentes entités du réseau ou avec la couche inférieure et supérieure au sein du même dispositif grâce au principe d'encapsulation. Chaque couche ajoute une couche d'information pour aider au partage de l'information. Afin de mieux visualiser cet aspect, nous donnons figure.1.3 une représentation de deux entités communicantes avec un élément réseau intermédiaire. Comme montré dans la figure 1.3, les données peuvent transiter par des éléments réseaux intermédiaires. Ces derniers peuvent changer des informations sur les couches qu'elles manipulent. Supposons qu'une trame soit émise depuis le premier calculateur vers le second, le routeur intermédiaire ne pourra affecter que les informations relatives aux couches physiques, lien et réseau.

Dans le cadre de ce manuscrit, nous ne nous concentrerons que sur les couches de niveau (2 et 3).

1.4.2 *Ethernet*

Afin de connecter les ordinateurs au sein d'une entreprise, les réseaux doivent être simples à mettre en place, à configurer, à déployer, peu coûteux et avec de bonnes performances moyennes.

L'*Institute of Electrical and Electronics Engineers* (IEEE) est un organisme multidisciplinaire qui a pour l'un de ses objectifs de fournir différentes normes industrielles. Une de ses normes dédié aux réseaux est : l'*Ethernet* [54]. Cette norme est polyvalente. Elle permet le fonctionnement en bus et en étoile. La norme est agnostique concernant son câblage, néanmoins le connecteur et le câblage le plus courant est le RJ45 [7]. Compte tenu de la grande flexibilité et de sa grande

utilisation, l'économie d'échelle permet de réduire les coûts. Elle se limite aux couches 1 et 2 du modèle OSI. La norme reçoit souvent des amendements par différents groupes de travaux IEEE ajoutant de nouvelles possibilités ou proposant de meilleures performances (des liaisons *Ethernet* en centaines de $Gbits.s^{-1}$).

Néanmoins, le protocole n'est pas adapté pour respecter des contraintes temporelles. Des travaux ont été réalisés afin d'ajouter la considération du déterminisme à *Ethernet* pour pouvoir l'utiliser dans un contexte temps réel.

1.5 EXEMPLES DE NORMES UTILISÉS DANS L'INDUSTRIE

Chaque application industrielle a des besoins différents et par extension, une topologie, une norme présentant un ensemble d'avantages et inconvénients pour la mise en œuvre. Dans cette section, nous allons étudier différents domaines applicatifs de réseaux et décrire les différentes normes sous-jacentes. Ces applications ont toutes besoin de déterminisme.

Le déterminisme signifie que pour deux situations identiques, le résultat produit doit toujours être le même. Dans le contexte des systèmes temps réel distribués, le déterminisme des transmissions est un élément crucial. Il peut être présent dans les liens de communications dans lequels aucune collision ne doit survenir. Mais, peut-être aussi présent dans les commutateurs réseaux pour un état de file d'attente donné, l'ordre et l'instant de départ des données doit toujours être la même. Dans ce manuscrit, nous considérerons uniquement les systèmes déterministes.

1.5.1 *L'approche automobile*

Le domaine automobile a subi de nombreuses évolutions ces dernières années, que ce soit par l'apport de nouvelles fonctionnalités facilitant la conduite (telle que le régulateur de vitesse) ou par de nouvelles réglementations imposant l'implantation de nouvelles technologies de sécurité (comme le limiteur de vitesse intelligent en 2024 dans l'Espace économique européen).

Afin de mettre en place ces nouvelles fonctionnalités, des réseaux ont été développés comme le réseau Controller Area Network (CAN) [60]. Ce réseau utilise une technologie en bus, avec des niveaux de priorités statiques et un mécanisme pour assurer l'absence de collisions dans le bus. CAN est grandement utilisé, ce qui a intrinsèquement réduit les coûts de fabrication de ses contrôleurs et facilité son utilisation dans le domaine automobile. Néanmoins, le bus possède des débits de fonctionnement plus faibles (jusqu'à 8 $Mbit.s^{-1}$) et des distances de fonctionnements limitées, affectant la vitesse de transmission.

En parallèle, un autre réseau pour l'automobile a été proposé : Local Interconnect Network (LIN) [61]. Les réseaux LIN sont moins coûteux que les réseaux CAN, mais ont des débits encore inférieurs. Par conséquent, ces réseaux sont réservés pour les dispositifs périphériques non critiques, comme le lève-vitre électrique ou le lave-glace.

Pour transmettre des flux plus lourds (comme la vidéo), le bus Media Oriented Systems Transport (MOST) [62] a été élaboré. Il fonctionne avec une fibre optique branchée en série entre les différents systèmes. Les débits peuvent monter jusqu'à $150Mb.s^{-1}$, mais le bus est plus coûteux

que le bus CAN à cause de la technologie à base de fibre optique et du coût des transmetteurs/récepteurs.

1.5.2 *L'approche aéronautique et aérospatiale*

Le choix effectué par l'avionneur Airbus est d'équiper ses avions d'un système distribué établi au dessus d'un réseau cœur. Le nombre de fonctions embarquées est grand et il est nécessaire d'interconnecter ces systèmes entre eux. La compagnie a choisi d'utiliser une norme *Ethernet* modifiée pour tenir compte des aspects de déterminisme temporel : Avionics Full DupleX switched ethernet (AFDX) [3]. La norme ajoute un système de chemins statiques et des contrats de trafic : un flux envoie des données qui ont des caractéristiques précises (délai minimum entre deux transmissions successives, taille des données, chemin). AFDX ajoute du déterminisme sur les transmissions ainsi sur les délais de bout-en-bout. Une donnée ne transitera pas dans le réseau si sa taille ou les délais d'attente entre deux émissions ne correspondent pas à ce qui a été déclaré. Ceci est nommé en anglais le *trafic policing*. Depuis son utilisation dans les Airbus A380, AFDX a été normée et fait partie de la norme ARINC 664 Partie 7[3].

Dans le contexte aérospatial, en particulier pour la navette spatiale Orion et son lanceur, un réseau *TTEthernet* [5] est utilisée. Cette norme repose sur la synchronisation de tous les dispositifs réseau et sur le principe de passage de trames prioritaires sur des créneaux temporels dédiés : le *Time-Tiggered Traffic*. Conséquemment, cela permet de prédire exactement à quel moment des données passent dans le réseau et donc de prouver le déterminisme. Cependant, un aspect freinant son développement industriel est la complexité de la mise en place de la configuration. Néanmoins, de nombreux travaux permettent d'obtenir des configurations valides [100][33][101][39][102].

Les normes *Ethernet* spécialisées sont néanmoins plus coûteuses que les normes CAN ou LIN vues précédemment [36].

1.5.3 *Industrie 4.0 et 5.0*

L'industrie 4.0 et 5.0 correspondent aux 4e et 5e révolutions industrielles, dans lesquelles les moyens de productions sont organisés de telle sorte à pouvoir les réorganiser dynamiquement. Cette réorganisation des processus de production a besoin de réseaux flexibles et peu coûteux [107].

Une des normes de l'IEEE concerne la mise en place de l'*Ethernet* ainsi que le fonctionnement de ses commutateurs. Une absence notable des normes *Ethernet* avant 2012, était l'absence de déterminisme temporel directement dans les normes de l'IEEE. Des normes annexes ont été proposées tel que l'Audio Video Bridging (AVB)[56] par-dessus les normes *Ethernet* pour introduire une couche de déterminisme sur des transmissions audio/vidéo. Cette norme a été pensée pour le domaine automobile.

Le Time Sensitive Networking (TSN) Task Group [106] a donc été constitué en 2012 pour rendre les réseaux *Ethernet* nativement sensibles au temps au-delà de seulement l'audio et la vidéo. Ce groupe de travail propose un ensemble d'amendements pour apporter du déterminisme dans les normes relatives à *Ethernet*, permettant ainsi de fournir une grande flexibilité et des coûts bas

TABLE 1.1 – Table récapitulative des caractéristiques de normes étudiées (✔ compatible, ✘ non compatible)

	Bus	Étoile	Coût	Déterminisme	Topologie flexible	Simplicité de configuration
CAN	✔	✘	€€	✔	✘	+
LIN	✔	✘	€	✔	✘	+
MOST	✔	✘	€€€	✔	−	+
Ethernet	✔	✔	€	✘	++	++
AFDX	✘	✔	€€€	✔	+	+
TTEhternet	✘	✔	€€€	✔	+	−−
AVB	✘	✔	€€€	✔	+	−
TSN	✔	✔	€€€	✔	++	−−

pour les réseaux temps réel. TSN est perçu comme pouvant être un réseau de premier choix dans l'industrie 4.0 [107][97]. Néanmoins, compte tenu de l'aspect récent et complexe des normes, le coût du matériel et l'effort nécessaire pour les configurer correctement est important.

1.5.4 *Récapitulatif des possibilités des normes*

Un récapitulatif des différents avantages et inconvénients des normes est disponible sur la table 1.1. Les données trouvées sont issues des références suivantes : [36][51][37][73] [80]. Tous les plus (+) et moins (−) sont considérés relatifs par rapport à la norme CAN. Nous pouvons remarquer que chaque norme dispose d'avantages et d'inconvénients qui peuvent se montrer plus ou moins pertinents en fonction de l'application recherchée. La norme TSN est très polyvalente, néanmoins certains aspects sont inconnus compte tenu qu'un commutateur TSN qui inclut plus de fonctionnalités sera plus coûteux et plus difficile à configurer.

1.6 CAS D'ÉTUDE REPRÉSENTATIF : ORION

Orion[83] est un véhicule spatial initié au cours du mandat présidentiel du président étasunien Georges W. Bush, faisant partie du programme spatial *Constellation*. Ce programme avait pour ambition de renvoyer des astronautes sur la Lune et à terme sur Mars. Néanmoins, si le programme fut ultérieurement annulé, le projet du véhicule spatial est lui resté. Il sert actuellement de base pour le programme spatial *Artemis* qui poursuit à peu près les mêmes objectifs. L'histoire de la conception de ce véhicule suit une approche radicalement différente du programme *Apollo*. Cette fois, la NASA a cherché à utiliser des composants grand public sur étagère et des architectures systèmes dérivées du milieu aéronautique afin de minimiser les coûts de conception [85]. Par ailleurs, une conception plus ouverte et accessible aux acteurs du domaine a été mise en place dans le but de recevoir de nouvelles propositions.

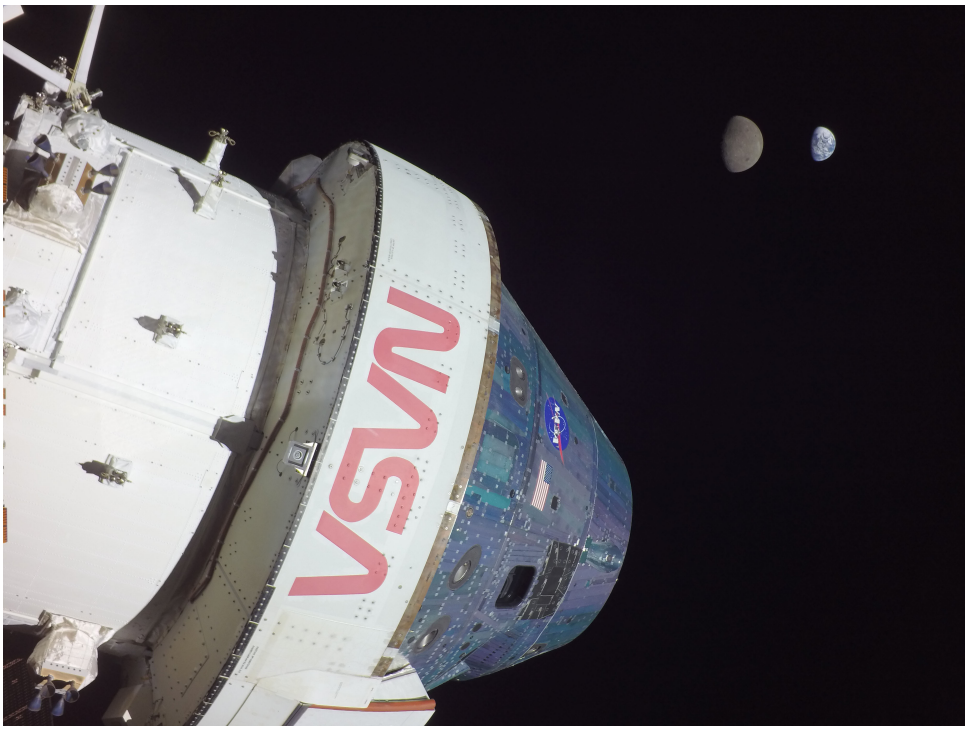

FIGURE 1.4 – Photo de la navette spatiale Orion lors de la mission *Artemis I* [82]

Ce cas d'étude nous servira de base pour différentes expérimentations que nous réaliserons. La navette dispose de contraintes fortes concernant ses niveaux de sûreté [85][76], les plus notables sont : le système doit pouvoir garantir un succès des missions à 99,9 %; lancer la navette doit être fiable à 99,99 %; le système informatique doit être tolérant aux fautes et ses tâches doivent pouvoir être faites manuellement par le personnel de bord.

L'approche choisie par les ingénieurs de la NASA afin d'effectuer les communications inter-systèmes est d'utiliser la norme *TTEthernet* 1.5.2. Ce réseau est principalement mis en place pour assurer la communication entre la navette Orion et son lanceur (originalement, *Ares-I* puis *Space Launch System*).

1.7 CONCLUSION

Dans ce chapitre, nous avons présenté les principes fondamentaux des systèmes embarqués temps réel distribués ainsi que les architectures de leurs réseaux associés. Nous avons introduit le modèle OSI et réalisé une revue détaillée de plusieurs standards réseaux essentiels. Il est apparu que les normes *Time-Sensitive Networking* (TSN) présentent des avantages significatifs par rapport à d'autres réseaux spécialisés. Nous avons également introduit un cas d'étude qui sera utilisé pour certaines des évaluations expérimentales produites. Le prochain chapitre abordera la norme TSN plus en détail, compte tenu qu'elle représente la base technologique de nos contributions.

2

RÉSEAUX TSN

Les normes *Time Sensitive Networking* (TSN) ont été développées pour répondre aux besoins spécifiques de différents acteurs industriels. Ces besoins sont adressés de façon incrémentale par l'IEEE, l'organisme en charge de l'élaboration de ces normes. Il en résulte un ensemble de normes complémentaires et combinables, permettant de fournir des réseaux sur mesure adaptés à divers besoins industriels. Il est donc essentiel d'avoir une vue d'ensemble du fonctionnement des normes TSN pour pouvoir extraire les informations nécessaires à leur mise en œuvre.

Dans ce chapitre, nous décrirons d'abord la façon dont les normes ont été produites pour répondre aux besoins des différents acteurs industriels. Ensuite, nous aborderons la topologie et le fonctionnement technique des réseaux TSN. Nous fournirons une description plus détaillée des *bridges* et *End-Stations* TSN, en particulier sur la manière dont les trames sont transmises en sortie de ceux-ci. Enfin, nous présenterons différents mécanismes permettant de fournir des qualités de services aux données dans le réseau.

2.1 HISTOIRE DES NORMES TSN

Le groupe de travail TSN [52] a pour mission d'intégrer la notion de temps dans les normes réseaux de l'organisme IEEE. Il résulte du renommage du groupe de travail *Audio Video Bridging* qui avait les mêmes objectifs. Cela a permis de rendre les avancées apportées par les normes produites moins spécifiques et adaptables à plus de cas d'utilisations industriels. Les travaux entrepris par le groupe de travail permettent d'apporter une sensibilité au temps sur des réseaux IEEE tels que le Wi-Fi, le Bluetooth ou encore l'*Ethernet*. Dans la suite de ce manuscrit, nous nous focaliserons sur l'application des travaux produits sur la norme réseau *Ethernet*. Parmi les problèmes adressés par ce groupe de travail, on peut citer :

— la prévention de la défaillance et l'atténuation des problèmes liés aux dispositifs réseaux et aux liens physiques entre ceux-ci ;
— la reconfiguration dynamique de l'infrastructure réseau pour répondre à de nouveaux besoins sans interruption du réseau ;
— l'interopérabilité des mécanismes temporels sur un standard unifié par rapport aux implémentations des concepteurs de matériel réseau ;
— le partage de l'infrastructure réseau entre des données ayant des niveaux de criticité différents (trafic plus ou moins prioritaire) ;
— la possibilité de configurer le réseau et ses aspects temporels avec une approche plus simple à configurer pour les utilisateurs, par rapport aux standards existants ;

En plus des normes produites, le groupe fournit également des profils qui guident les industriels dans l'utilisation des normes dans des contextes applicatifs précis, tels que l'audio et la vidéo critiques conjointement à d'autres types de trafic temps réel (Profil AVB de la figure 2.1) ou la mise

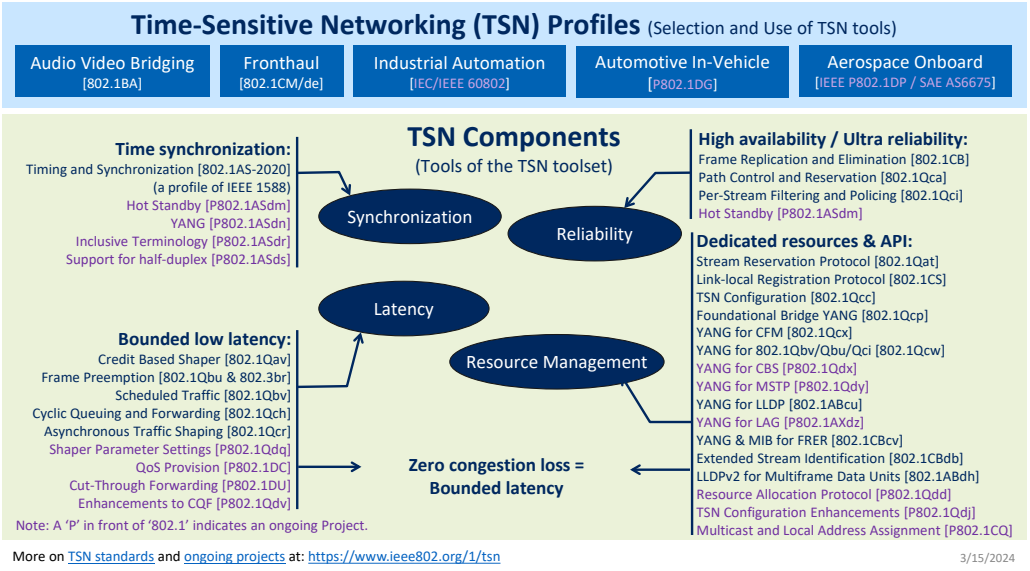

FIGURE 2.1 – Description des domaines d'action du groupe de travail TSN et des normes produites associés[53]

en place de réseaux dans les usines (profil IEC/IEEE 60802 de la figure 2.1). Les profils peuvent être vus comme un guide pour utiliser les normes dans un contexte applicatif précis.

Toutes les contributions sont rassemblées sur la figure 2.1 issue du site web du groupe de travail [53]. Toutes les normes et tous les profils préfixés par un "P" indiquent que des travaux sont encore en cours. En haut, sont décrits les différents profils et en bas, les différentes normes produites. Les normes sont catégorisées en quatre aspects : synchronisation, fiabilité, gestion des ressources et latence. Dans la suite du manuscrit, nous traiterons uniquement les aspects liés à la latence, les autres étant en dehors de nos travaux.

2.2 DONNÉES ET TOPOLOGIE D'UN RÉSEAU TSN

Dans cette section, nous étudierons la constitution physique d'un réseau TSN. L'objectif principal d'un réseau TSN est de transmettre des données entre 2 calculateurs du réseau. À l'échelle du réseau, les calculateurs sont appelés ***End Stations*** (ES). Ces données sont encapsulées dans des trames, au niveau de la couche 2 du modèle OSI, comme mentionné au paragraphe 1.4.1.

La figure 2.2 illustre la composition des trames échangées par les entités. Une trame comprend un entête permettant d'annoncer l'arrivée d'une trame à l'entité réceptrice. Elle est composée : l'adresse physique de l'émetteur et du ou des destinataires (sous la forme de l'adresse MAC), les informations liées aux réseaux virtuels (non considérées dans ce manuscrit) et le niveau de priorité, la version d'*Ethernet*, les données, une somme de contrôle permettant de vérifier l'intégrité de la trame reçue. La taille maximale d'une trame *Ethernet* TSN est de 1522 octets et sa taille minimale est de 64 octets. Le niveau de priorité est compris entre zéro et sept, rangés par ordre croissant.

Pour structurer la transmission des données, les trames sont organisées par flux, application de la couche sept du modèle OSI (voir 1.4.1). Cela permet de simplifier l'organisation des données pour le concepteur de réseaux. Prenons l'exemple du véhicule spatial Orion et de son lanceur, abordés

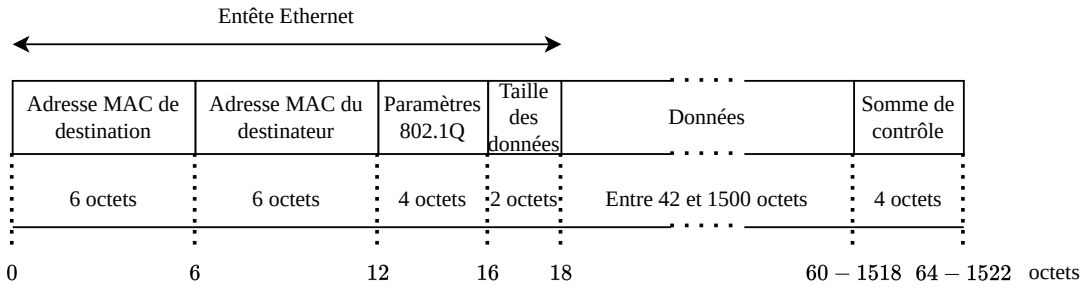

FIGURE 2.2 – Visualisation d'une trame *Ethernet* TSN

dans la section 1.6. Supposons que le réacteur du lanceur envoie des données à plusieurs systèmes de la navette et que la taille de ces données soit supérieure à la taille maximale d'une trame *Ethernet*. Il est possible de suivre et de configurer toutes les trames échangées. Cependant, il sera nécessaire de spécifier différentes trames contenant chacune des parties distinctes des données d'intérêt, avec le risque qu'elles se séparent. Les flux permettent d'abstraire ces échanges. C'est-à-dire que nous nous intéressons plus à la transmission de données d'un point à un autre du réseau qu'à analyser des trames indépendantes sans aucun lien les unes des autres. Les flux ont pour paramètres : les destinataires, l'émetteur, le chemin statique par lequel les trames doivent transiter, la taille des données à échanger en *bits*, le niveau de priorité. De plus, les flux peuvent être :

— **périodiques** : les transmissions consécutives sont exactement séparées par un délai constant T ;
— **sporadiques** : les transmissions consécutives sont séparées d'au moins un délai T.

Dans la suite du manuscrit, nous indiquerons au niveau des descriptions des modèles associés aux contributions si les flux sont sporadiques ou périodiques.

Deux types d'éléments manipulent ces flux de trames :

— les ***End-Stations*** (ES) : responsables de la production et de la réception des données. Ces dispositifs se situent aux extrémités du réseau et disposent d'un seul lien bidirectionnel avec celui-ci.
— les ***Bridges*** (Br) : ces entités se chargent de collecter les trames sur leurs entrées et de les rediriger vers au moins une de leurs sorties en fonction des chemins. Dans ce manuscrit, cette configuration se fait avant le lancement du réseau.

La topologie d'un réseau TSN suit généralement le mode de fonctionnement combinant deux topologies réseaux vues précédemment (sous-section 1.3) : celle d'un réseau en étoile et maillé. La figure 2.3 représente ce mode de fonctionnement. Nous remarquons que les ES sont toujours en périphérie du réseau et que les *bridges* sont les composants permettant de mettre en relation différents ES.

Les *bridges* et *End-Stations* sont les composants principaux où la majeure partie des besoins en termes de déterminisme est appliquée. Dans la section suivante, nous analyserons plus en détail leurs fonctionnement.

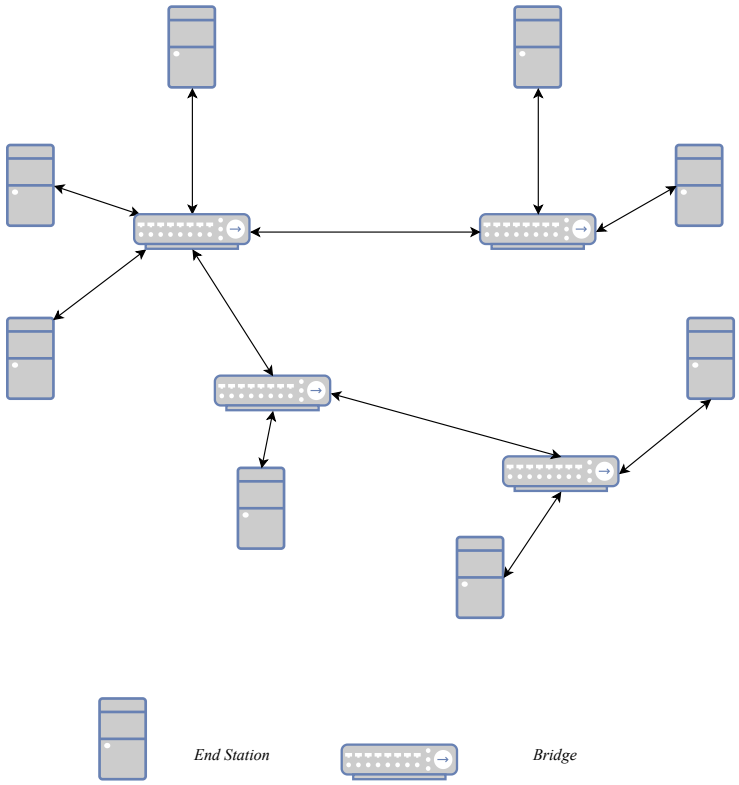

FIGURE 2.3 – Description de la topologie d'un réseau TSN standard

2.3 PRINCIPE DE FONCTIONNEMENT DES *BRIDGES* ET *END-STATIONS* TSN

Les *bridges* doivent également vérifier si les trames en entrée sont attendues/correctes. Ils peuvent, par exemple, vérifier si la somme de contrôle est valide, si la trame entrante n'est pas déjà passée une fois dans le *bridge* (nous parlons ici de boucle) ou encore si les trames appartenant à des flux respectent bien leurs paramètres (abordé sous-section 2.2). Si ce contrôle détecte une erreur, la trame est supprimée. Cependant, les *bridges* ne peuvent pas corriger les erreurs.

Un *bridge* est composé d'au moins un port d'entrée et d'un port de sortie. La figure 2.4 en représente un exemple. Celui-ci dispose de trois ports d'entrée et de trois ports de sortie. Visuellement, une trame entre par un des ports sur la gauche, traverse la couche de traitement, est redirigée vers l'une des files d'attente d'un ou de plusieurs ports de sortie, puis est transmise. Sur cette figure, les ports d'entrée et de sortie sont situés à deux endroits physiques différents. Cependant, en fonction du câblage utilisé avec le *bridge*, un connecteur peut permettre l'émission et la réception simultanées des données, sans que cela affecte la représentation faite ici.

Une *End-Station* TSN est quant à elle en charge de transmettre des données applicatives et de recevoir celles transmises par d'autres ES. La façon dont cette dernière fonctionne est analogue à celle d'un bridge, mais avec un seul port d'entrée et un seul port de sortie sans que les deux soient reliés comme pour les *bridges*.

Dans la prochaine section, nous nous concentrerons exclusivement sur le fonctionnement des files d'attente et les ports de sortie des *bridges* ou ES. La gestion des trames, l'évaluation des délais des trames passantes ainsi que leurs capacités de stockage dans leurs ports de sortie constituent le

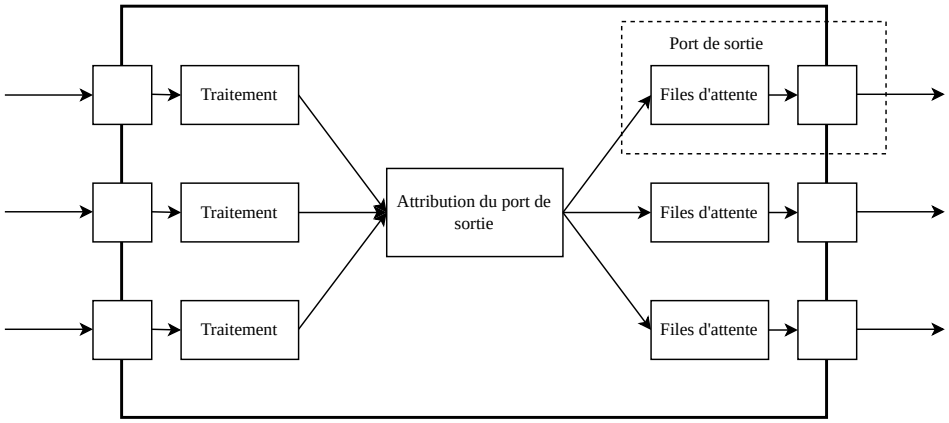

FIGURE 2.4 – Représentation d'un *bridge* TSN

principal point d'intérêt de ce manuscrit. Cela correspond à l'encadré en pointillés montré sur la figure 2.4.

2.4 PORTS DE SORTIE (ES/BRIDGE)

Dans cette section, nous allons aborder plus en détail comment opèrent les ports de sortie dans les réseaux TSN. Nous représentons sur la figure 2.5, un port de sortie et ses différents composants.

Les trames arrivant dans le port de sortie sont affectées à des files d'attente. Leur nombre est compris entre 1 et 7. Il est choisi soit par l'implémentation du fabricant du bridge, soit par le concepteur du réseau qui peut choisir de bloquer des files d'attente. Dans la norme, l'ordre dans laquelle ces trames au sein d'une file sont distribuées n'est pas défini. Néanmoins, nous faisons l'hypothèse que celles-ci sont distribuées en suivant la politique d'ordonnancement *First In First Out* (FIFO), justifiant leur appellation de file d'attente.

La file d'attente est ensuite régulée par des politiques de service permettant de garantir une qualité de service suffisantes pour les besoins du système. Les qualités de service sont décomposées en politiques de services et en mécanismes temporelles pour la transmission de trames. Elles sont autonomes et fonctionnent individuellement pour chaque file d'attente. Elles lèvent des drapeaux si celles-ci autorisent la distribution d'une trame de la part de la file d'attente. La classe de trafic la plus prioritaire ayant son drapeau levé, gagne le droit de transmission.

La combinaison des files d'attente avec les mécanismes influençant la qualité de service est appelée une **classe de trafic**.

Les mécanismes de qualité de service dans les ports de sortie constituent un axe majeur des contributions que nous fournirons dans la suite du manuscrit. Par conséquent, nous allons aborder dans la section suivante, certains de ces mécanismes.

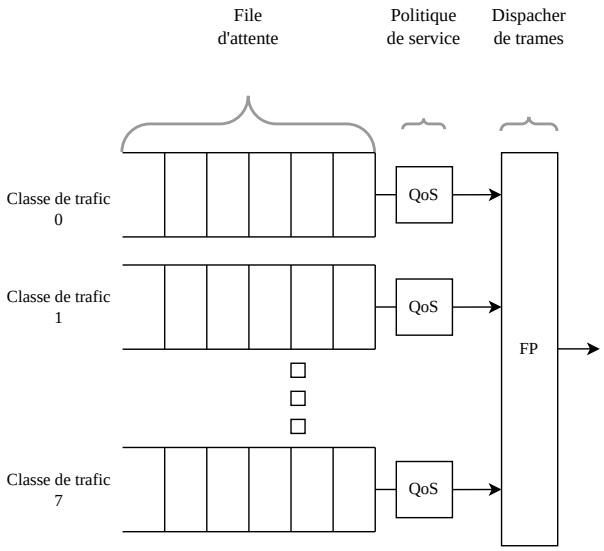

FIGURE 2.5 – Représentation d'un port de sortie TSN

2.5 QUALITÉS DE SERVICE DANS TSN

Les *bridges* et les *End-Stations* sont capables d'appliquer différents niveaux de qualité de service pour les transmissions qui transitent au niveau des portes de sortie. Les politiques de service sont un ensemble de mécanismes fourni afin d'assurer la fiabilité et de bonnes performances d'un point de vue global.

Les politiques de service disponibles dans la norme 802.1Q [57] lors de la rédaction de ce manuscrit sont au nombre de quatre. Ce sont les blocs interchangeables *Quality of Service* (QoS) que nous avons représenté sur la figure 2.5. Nous allons décrire les deux qui seront étudiés dans la suite du manuscrit : *Strict Priority* (priorité stricte) et *Credit-based Shaper*. Après avoir décrit ces politiques de service, nous aborderont le mécanisme temporel appelé *Time Aware Shaper*.

2.5.1 *Strict Priority*

Dans les *bridges* TSN, la politique de service *strict priority* est utilisée par défaut dans les réseaux TSN. Cette politique de service peut transmettre si une trame est présente dans la classe de trafic et si elle est la classe la plus prioritaire avec une trame disponible. Aucun mécanisme supplémentaire n'est proposé avec cette politique de service. Nous aborderons ce mécanisme dans la suite de manuscrit sous l'appellation de *Fixed-Priority/First In First Out* (FP/FIFO). Nous expliquons son mode de fonctionnement à l'instant : un niveau de priorité est associé à la classe de trafic et les trames en leur sein prennent leur niveau de priorité. Les trames de plus hautes priorités sont toutes distribuées avant celles moins prioritaires. Si des trames de haute priorité arrivent dans le commutateur avant d'avoir distribué celles de plus faible priorité, les nouvelles trames plus prioritaires seront distribuées avant de pouvoir reprendre les transmissions de plus faible priorité.

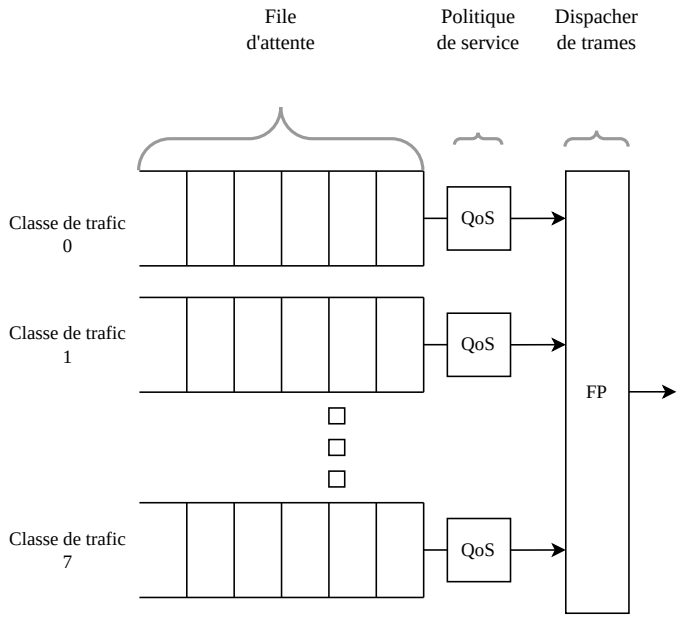

FIGURE 2.6 – Représentation d'un port de sortie TSN utilisant FP/FIFO

Cependant, les trames en cours de transmission ne sont jamais interrompues. Par convention, les classes de trafic avec le plus grand nombre sont les plus prioritaires.

Dans le cas du mélange de politiques de service entre FP/FIFO et une autre politique, le choix de la classe de trafic qui peut transmettre se fait sur la base de la classe de la trafic avec le plus grand nombre ayant son drapeau de levé. Nous représentons ce cas-là sur la figure 2.7. Dans le port de sortie, il y a trois classes de trafic visibles sur les huit disponibles et ces trois classes sont soumises à cette politique. La classe de trafic sept est la plus prioritaire et la classe zéro, la moins.

2.5.2 *Credit-based Shaper*

Le *Credit-based Shaper* (CBS) est une politique de service initialement développée pour permettre de faire transiter des données Audio/Vidéo dans le contexte de la norme réseau AVB [56] (section 1.5.3). Cette politique de service a été conçue pour répondre à certaines limitations inhérente à FP/FIFO .

Nous allons illustrer notre propos par un exemple, montrant cette limitation dans le contexte de partage de réseau entre flux audio/vidéo et d'autres types de données. Les flux audio/vidéo sont des types de flux qui lorsque émis en continu peuvent entraîner une grande utilisation de bande passante sur les liens. Si les flux vidéo sont mis en haute priorité, le reste du trafic pourrait être fortement retardé, et si ceux-ci sont mis en basse priorité, des interruptions dans le flux vidéo pourront survenir. Pour éviter ce genre de problème, le *Credit-based Shaper* (CBS) a été défini par l'IEEE [58]. Son objectif initial est de laisser du temps libre aux flux qui ne sont pas régis par le CBS pour effectuer des transmissions. Par exemple, la transmission d'un flux vidéo de haute priorité

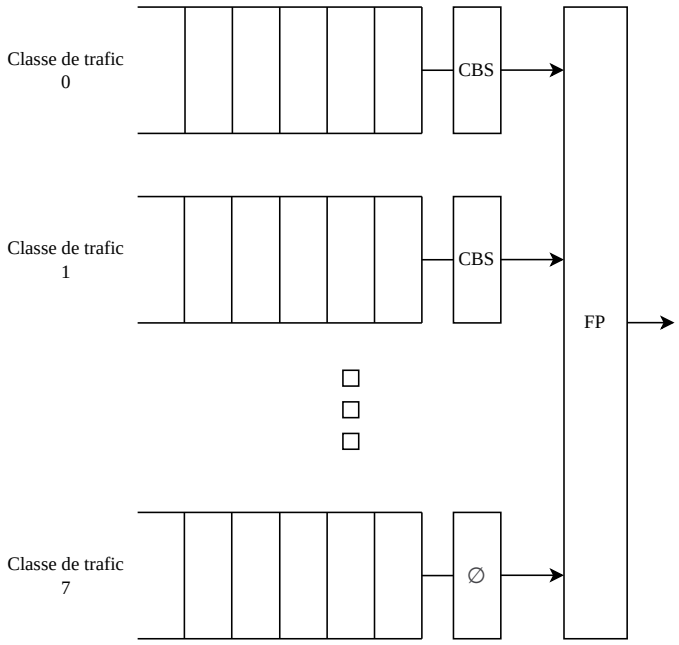

FIGURE 2.7 – Représentation d'un port de sortie TSN utilisant la politique de service CBS

dans une voiture ne doit pas affecter les messages de moins haute priorité comme la commande du lève-vitre.

CBS est une politique de service non conservative, c'est-à-dire que la politique de service peut introduire des temps d'inactivité, même si des trames sont présentes dans les classes de trafic. Nous représentons sur la figure 2.7 son positionnement dans le port de sortie. CBS prend la place des QoS et reste malgré tout soumis à FP/FIFO.

CBS repose sur l'attribution de crédit pour chaque classe de trafic qui l'utilise. Ce crédit, exprimé en bits, croît linéairement à un rythme donné par le concepteur de réseau, nommé *idle slope* (σ^+, exprimé en $bits.s^{-1}$. Le crédit décroît à la vitesse de la *send slope* (σ^-). La *send slope* est calculée comme suit : $\sigma^- = \sigma^+ - r$ où r est la vitesse de transmission du port de sortie en $bits.s^{-1}$.

Le taux de recharge du crédit, représenté par l'*idle slope*, doit être choisi suffisamment grand pour permettre de transmettre les données voulues. Par exemple, dans le cadre d'une transmission d'un flux vidéo avec un débit binaire donné, l'*idle slope* devra être au moins égal à ce débit binaire. Dans le cas contraire, des débordements des classes de trafic peuvent survenir, entraînant des suppressions de trames.

Cette politique de service suit les règles suivantes :

— lorsque le système démarre, le crédit est initialisé à zéro ;
— lorsqu'aucune trame n'est présente dans la classe de trafic, que le crédit est supérieur ou égal à zéro, le crédit de la classe est remis à zéro ;
— si le crédit est positif ou nul, une transmission depuis la file d'attente peut commencer si aucune classe de trafic de plus haute priorité ne peut transmettre ;

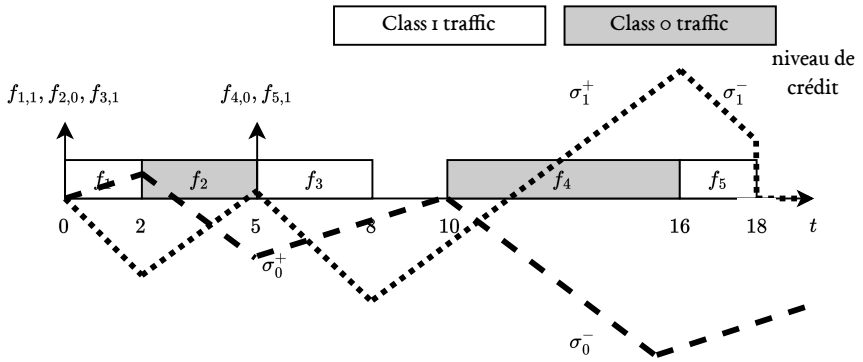

FIGURE 2.8 – Description du fonctionnement de CBS sur un exemple

— si une trame est présente dans la file d'attente, mais sans aucune transmission de celle-ci ou que le crédit est négatif, celui-ci se recharge au rythme de l'*idle slope* σ_n^+, où n est le numéro de la classe de trafic ;
— lorsqu'une trame est émise, le crédit de la file d'attente décroît au rythme de la *send slope* σ_n^-, où n est le numéro de la classe de trafic.

La fig.2.8 représente le comportement de deux classes de trafic utilisant simultanément CBS avec des paramètres d'*idle slope* et de *send slope* différents. La figure est un diagramme de Gantt temporel qui décrit les transmissions. Les flèches vers le haut indiquent l'admission des trames dans les classes de trafic.

Lors de l'initialisation du *bridge*dans la figure 2.8, les niveaux de crédit sont à 0 et 3 trames f_1,f_2,f_3 pour deux classes de trafic (classe 0 et 1) sont admises dans le *bridge* : deux appartenant à la classe 1 et une appartenant à la classe 0. Les crédits des deux classes étant à zéro, la classe de trafic 1 (plus prioritaire) remporte le droit de transmission. Une fois la trame f_1 émise à $t = 2$, la classe 1 a un crédit négatif et ne peut donc plus transmettre. La classe de trafic 0 peut transmettre, car elle a un crédit positif. Une alternance se poursuit pour la transmission de f_3 à $t = 5$ et à $t = 8$, aucune des deux classes de trafic n'a de crédit. Ce temps d'inactivité aurait pu servir à transmettre des trames de plus petite priorité. Nous montrons ce comportement dans la figure 2.8 Ainsi, les deux classes de trafic attendent que leurs crédits remontent au rythme de l'*idle slope*. La classe de trafic 0 est la première à avoir un crédit nul, donc elle peut transmettre une trame f_4 à l'instant $t = 10$. La trame finit sa transmission à $t = 16$ et la transmission de f_5 débute pour terminer à $t = 18$. À cet instant, plus aucune trame n'est à distribuer pour les classes de trafic 0 et 1. Comme la classe 1 a un crédit supérieur à zéro, son crédit est mis et maintenu à zéro. Pour la classe 1, le crédit étant négatif, il va croître au rythme de l'*idle slope*. Une fois remis à zéro (et si aucune nouvelle trame de la classe n'est arrivée), le crédit y sera maintenu.

2.5.3 *Time Aware Shaper*

Le *Time Aware Shaper* (TAS) [55], définie pour la norme 802.1Q [57], vise à apporter des délais de bout en bout constants pour certaines classes de trafic, tout en minimisant son impact sur les autres classes. Dans une approche optimale, TAS doit permettre aux trames périodiques

de traverser les *bridges* en évitant les congestions et en minimisant la gigue des trames qui en bénéficient. TAS s'ajoute au bout de chaque classe de trafic représenté sur la figure 2.5 ce qui nous donne la figure 2.9.

Prenons l'exemple du cas d'étude de la navette spatiale Orion présenté dans la section 1.6. Certains messages essentiels doivent pouvoir transiter dans le réseau avec la garantie qu'aucune congestion ne se produira. Dans le cadre de la navette spatiale, le réseau employé est *TTEthernet*, mais le principe de synchronisation temporelle et de passage de trames à des moment dédiés est semblable au TAS que nous allons présenter par la suite.

Principe de fonctionnement

Ce système fonctionne en mettant en place des portes au niveau de chaque classe de trafic. Les portes ont deux états possibles :

— **ouverte** : les trames sont envoyées dans les mêmes conditions qu'en temps normal (les différentes politiques de service opèrent normalement) ;
— **fermée** : aucune trame ne peut être transmise depuis les classes de trafic dont les portes sont fermées. Il est important de noter que lorsque TAS est utilisé conjointement avec la politique de service CBS, la recharge du crédit est interrompue si la porte de la classe de trafic est fermée.

Pour s'assurer que les transmissions ne soient pas coupées lors d'une transition de porte ouverte à porte fermée, un temps où les transmissions peuvent continuer, mais pas commencer apparait avant la transition. Ce temps s'appelle la **Guard Band**. Sa taille est soit égale à la taille de la plus grande trame qui peut passer dans le commutateur, soit dans le cadre de commutateurs plus élaborés égale à la taille de la plus grande trame dans le commutateur. Nous émettons l'hypothèse que la taille de la **Guard Band** correspond à la taille de la plus grande trame pouvant passer dans le *bridge*/ES pour fonctionner dans les commutateurs TSN de base.

Ces portes s'ouvrent et se ferment en suivant un planning élaboré hors-ligne ou en ligne. Un planning hors-ligne est généralement utilisé lorsque le système n'est pas soumis à des changements pendant son exécution, comme la navette Orion qui ne sera pas reconfigurée en service. Un planning en ligne suppose des reconfigurations dynamiques pendant l'exécution du système, comme dans le cas des systèmes industriels 4.0 ou 5.0 présentés dans la section 1.5.3, qui peuvent dynamiquement changer en fonction des demandes. Les systèmes temps réel critiques suivent généralement une configuration hors-ligne. Les plannings peuvent se répéter indéfiniment et sont appelés *Gate Control List* (GCL). Cette liste fournit des intervalles de temps et l'état de toutes les portes des classes de trafic du port de sortie correspondant. Selon la norme 802.1Q [57], le choix de l'horloge pour contrôler le TAS est libre à l'implémentation. Cependant, dans la suite de ce manuscrit, nous considérerons que toutes les horloges guidant le TAS sont parfaitement synchronisées.

La Figure 2.9 représente un port de sortie TSN avec 3 classes de trafic, différents shapers et le système de portes. Dans cette figure, la GCL est représentée sur la droite. La GCL dispose de plusieurs états tx où x représente un intervalle de temps. La longueur des intervalles est définie par le concepteur du réseau. Lorsque le curseur est sur un état, les portes correspondantes s'ouvrent ou se ferment selon la configuration de cet état. L'intervalle de temps sélectionné dans la Figure

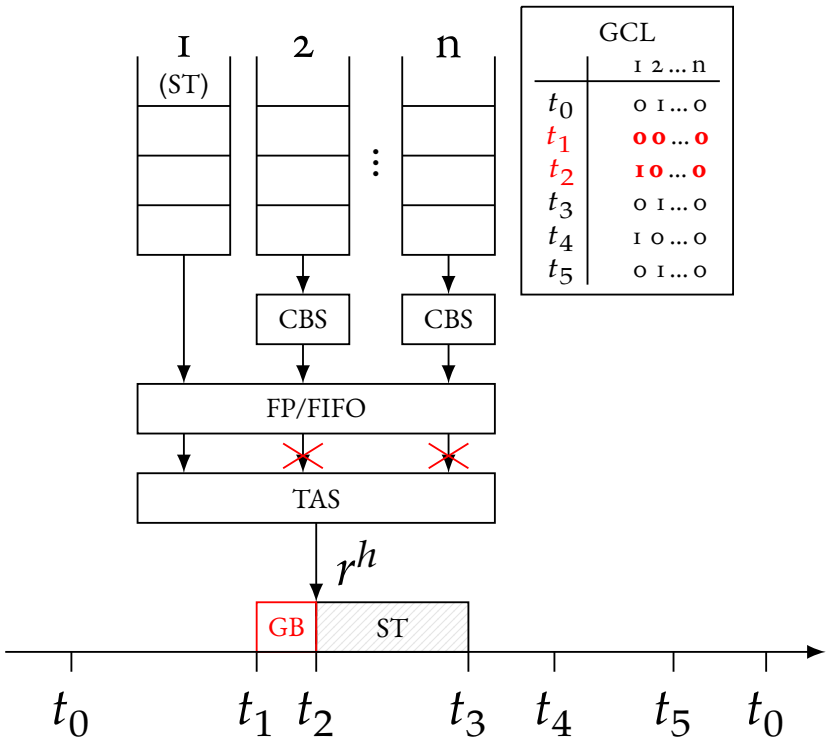

FIGURE 2.9 – Exemple de fonctionnement du Time Aware Shaper (à l'instant $t2$)

2.9 est *t2*. Dans cette configuration, la porte 1 est ouverte, donc la classe de trafic correspondante peut transmettre ses trames. À l'inverse, la porte 2 est fermée, par conséquent, aucune trame de la classe deux ne peut être transmise.

La norme fournit des indications sur la configuration de ces tables (Annexe T de la norme [57]), mais ne propose pas de solutions concrètes pour y parvenir.

2.6 CONCLUSION ET HYPOTHÈSES DÉFINIES

Dans ce chapitre, nous avons détaillé la topologie réseau standard d'un réseau TSN orienté temps réel ainsi que le mode de fonctionnement d'un *bridge* TSN standard. Nous avons abordé plus en détail les actions qui se déroulent dans les ports de sortie des *bridges* et les différentes politiques de services pouvant affecter l'ordre de transmission des trames. L'utilisation et la combinaison de ces politiques de services définissent la politique d'ordonnancement des trames sur le réseau et en conséquence influencent leurs délais de transmission de bout en bout. Il convient de noter que ce qui a été présenté dans ce chapitre sont des outils. Ces outils ne fournissent aucune preuve de déterminisme. Utilisés de manière inappropriée, les possibilités décrites peuvent même dégrader les performances et le déterminisme du système entier (exemple dans le paragraphe 2.5.2). Nous récapitulons sur la table 2.1, les hypothèses prises pour la suite du manuscrit concernant les réseaux que nous utiliserons.

Dans le chapitre suivant, nous allons aborder plus en détail les différentes manières d'apporter des preuves de déterminisme et leurs méthodes associées dans les réseaux basés sur *Ethernet* et TSN.

Catégories	Hypothèses
Synchronisation Temporelle	Le réseau est parfaitement synchronisé
	Aucune dérivation d'horloge ne peut se produire pendant l'exécution du système
Fiabilité	Les trames ne peuvent pas entrer en collision dans les liens
	Les commutateurs sont en mode de fonctionnement nominal et ne peuvent pas tomber en panne
	Aucune attaque malicieuse compromettant le réseau ne peut survenir
Gestion des ressources	Les configurations données pour les flux et commutateurs sont correctes
	Aucune reconfiguration n'est possible une fois le système lancé

TABLE 2.1 – Table des hypothèses prises pour la suite du manuscrit

3

ÉTAT DE L'ART

Dans le chapitre 1, nous avons abordé le besoin de déterminisme pour les acteurs industriels afin de garantir que les systèmes conçus ne présentent pas de risques pour les investissements et les individus. Nous avons vu comment fonctionnent les commutateurs TSN dans le chapitre 2. Les normes TSN fournissent des outils, mais ne prouvent pas le déterminisme. Des travaux de recherche ont proposé des résultats permettant aux industriels de produire des preuves de déterminisme dans les réseaux TSN.

Apporter une preuve de déterminisme à l'échelle globale d'un système temps-réel distribué est une tâche complexe. Dans ce manuscrit, nous nous concentrerons sur des preuves à l'échelle du réseau et des ports de sortie des commutateurs réseaux TSN.

La preuve de déterminisme peut être effectuée sur différents aspects fonctionnels. Nous avons choisi de nous intéresser aux problématiques suivantes dans leur ordre d'apparition dans ce chapitre :

— prouver que les trames traversent le réseau dans un temps borné ;
— déterminer, à partir de contrats de trafic, la quantité maximale de données en nombre de trames dans un port de sortie ;
— configurer formellement les commutateurs pour garantir l'absence de congestion des trames prioritaires dans les ports de sortie.

Dans ce chapitre, nous allons décrire les méthodes existantes pour les différents points cités précédemment.

3.1 ÉVALUATION DES DÉLAIS DE BOUT EN BOUT

3.1.1 *Problématique*

Un message transmis d'un ES à un autre dans un réseau TSN traverse physiquement un ou plusieurs commutateurs. Le délai de transmission d'une application critique doit être déterminé et borné. Afin de garantir le respect des contraintes temporelles d'un système concernant la traversée des messages dans un réseau, il est nécessaire d'évaluer les délais de transmission de bout en bout. Cette évaluation permet de s'assurer que les trames qui traversent un réseau seront toujours temporellement valides une fois celui-ci traversé. En d'autres termes, il faut garantir que le temps de transmission de chaque trames du réseau est borné et inférieur à un temps de traversée maximal défini par le concepteur du système.

Il existe plusieurs définitions liées aux temps de traversée, représentées sur la figure 3.1 :

— la borne inférieure pire cas : une borne inférieure optimiste du temps de passage dans le réseau ;

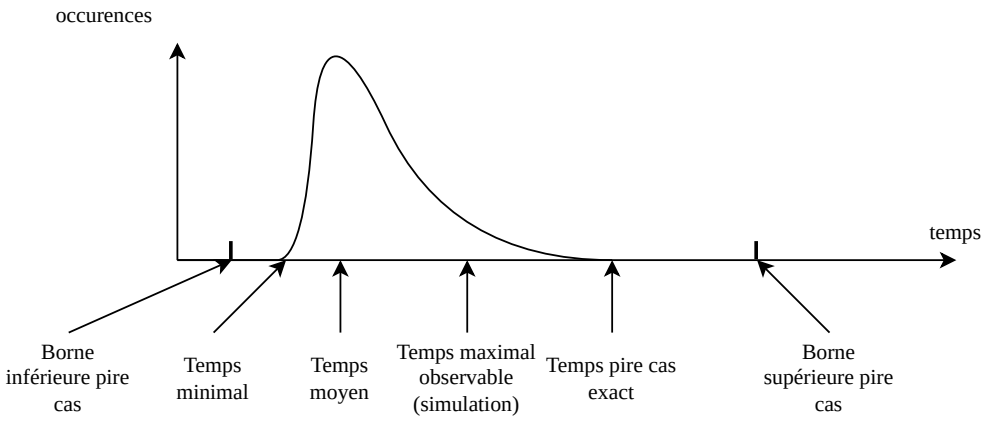

FIGURE 3.1 – Vocabulaire relatif aux différents temps de transmissions dans un réseau

— le temps minimal : le temps de traversée réseau dans les conditions optimales, sans congestion dans les commutateurs, lorsqu'il n'y a aucun trafic autre que celui étudié ;
— le temps moyen : déduit sur un grand nombre d'émissions de trames observées, généralement retenu lors de tests de performance ;
— le temps maximal observable : le temps maximal visible lors de simulations ;
— le temps pire cas : le temps de traversée réseau dans le pire des cas possibles, réalisable par le système ;
— une borne supérieure du pire cas : un temps garanti d'être supérieur ou égal au temps pire cas réalisable, mais ne correspondant pas nécessairement à celui-ci.

Pour prouver le déterminisme, il est possible de trouver le délai de transmission pire cas par la vérification de modèles (*model checking*) [29]. Le *model checking* permet de vérifier, à partir de la modélisation d'un système si une propriété est valide. Pour se faire, un parcours de toutes les possibilités du modèle est effectué en appliquant les contraintes à chaque étape. Cela conduit à un nombre de contraintes équivalent à n^x, où n correspond au nombre d'éléments étudiés et x au nombre de contraintes. Cette approche mène intrinsèquement à un problème d'explosion combinatoire [29][1], limitant son utilisation pour les réseaux de taille industrielle. Ce formalisme a été appliqué pour les réseaux AVB [59][65][6] et TSN [74] avec le model checker UPPAAL [18].

Une heuristique est une méthode de calcul qui assouplit des contraintes et recourt à des approximations pour obtenir des solutions plus rapidement que les méthodes exactes. Transformer la recherche d'un résultat exact en une approximation de la borne supérieure permet de réduire la complexité de la recherche pire cas. Dans le cadre de ce manuscrit, nous nous limiterons à identifier une borne supérieure pire cas pour les transmissions de bout en bout réseau.

Les sous-sections suivantes aborderont le cas de la simulation et de différentes heuristiques permettant de déterminer des bornes supérieures pires cas.

3.1.2 *Simulation*

La simulation consiste à exécuter un modèle du système étudié dans des conditions proches des conditions réelles. Le système simulé est lancé avec ses données au sein de la topologie du réseau, un grand nombre de fois avec des variations sur le temps d'envoi des messages, afin de couvrir le plus grand nombre de résultats possibles. La compilation des résultats permet d'obtenir le pire temps observable par simulation, qui n'est cependant pas nécessairement égal au pire temps de traversée réel. Le délai ainsi calculé n'est pas forcément garanti.

Des travaux concernant la simulation des réseaux TSN ont été proposés en utilisant les simulateurs NESTING [42], RTaW-Pegase [77] et OMNeT++ [115]. Des travaux initiaux de simulations, précurseurs par rapport aux normes TSN actuelles, ont aussi été conduits pour une combinaison AVB et *TTEthernet* [2] en 2012.

3.1.3 *Network Calculus*

La méthode Network Calculus [34][35] repose sur l'algèbre *min-plus* [98] et sur une analyse séquentielle des délais sur les différents commutateurs. Dans les explications suivantes, nous considérons le cas d'un commutateur utilisant exclusivement la politique de service FIFO.

À l'intérieur des commutateurs, tout ce qui a trait au trafic réseau est représenté sous la forme de courbes associées au temps t :

— la courbe d'arrivée $\gamma(t)$: représente les données admises en entrée $[0, t]$;
— la courbe de service $\beta(t)$: représente la capacité d'émission des trames par le commutateur réseau.

La courbe d'arrivée est égale, à l'instant 0, à la taille de la plus grande des trames passant dans le commutateur. Cette courbe croît linéairement en suivant le paradigme du *seau percé*. Les données arrivent continuellement dans le commutateur, qui les distribue également de manière continue. Il convient de noter que pour d'autres politiques de services, les courbes peuvent ne pas être linéaires.

La pente de la courbe de service doit être supérieure à celle de la courbe d'arrivée. Dans le cas contraire, la quantité de données entrant dans le commutateur excédera la capacité du port de sortie à les transmettre, ce qui conduira inévitablement à un débordement et à une suppression de données. La figure 3.2 représente les courbes d'arrivée γ et de service β. La courbe γ croît à un rythme de r, et b correspond à la taille de la plus grande des trames pouvant être émises par le commutateur.

La combinaison de ces deux courbes permet de déduire le délai de transmission pire cas ainsi que l'occupation pire cas en espace de stockage, exprimée en bits. La figure 3.3 illustre ce concept. Le calcul de la pire différence verticale entre les deux courbes ($v(\gamma, \beta)$) permet d'obtenir la pire occupation, en bits, de l'espace de stockage de la file d'attente du port de sortie du commutateur. De même, la pire différence horizontale ($h(\gamma, \beta)$) permet d'obtenir le pire délai de transmission.

Les délais déduits dans un commutateur sont propagés dans les suivants, permettant ainsi le calcul des délais de bout en bout du réseau.

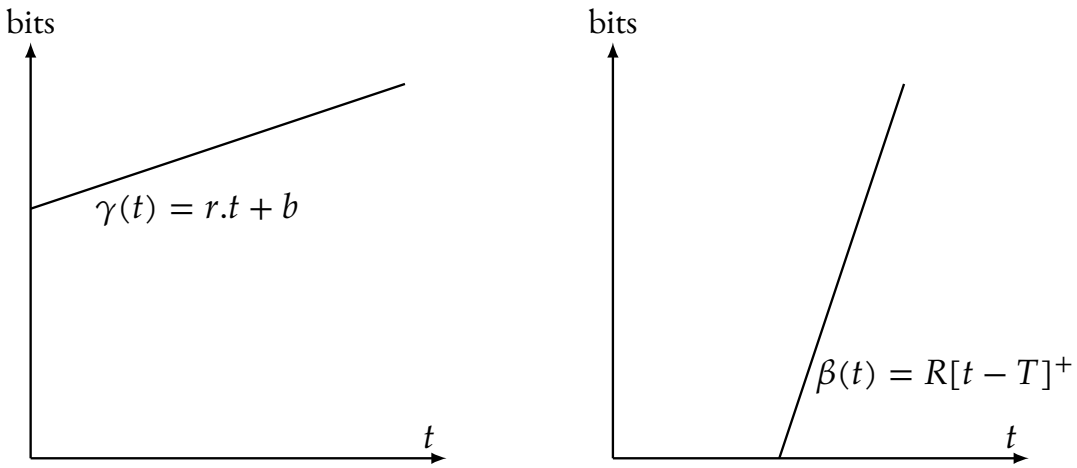

FIGURE 3.2 – Courbes d'arrivée γ et de service β

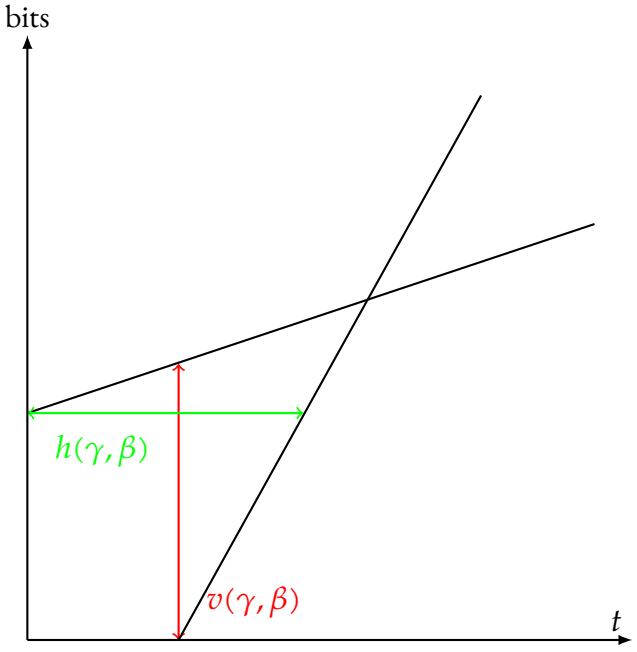

FIGURE 3.3 – Combinaison des courbes d'arrivées et de service

La méthode a été proposée pour les réseaux utilisant les politiques de service FIFO [69], FPFIFO [21] et dans les réseaux AFDX [49]. Une version adaptée au contexte des réseaux TSN avec le Time Aware Shaper (sous-section 2.5.3) a été proposée [112], ainsi que des versions adaptées aux flux CBS dans TSN [111] [114], [113]. Différents outils reposent sur le Network calculus, tels que : RTaW-Pegase [22], Disco [12], CONIC [20].

3.1.4 *Compositional Performance Analysis*

La méthode Compositional Performance Analysis (CPA) [91] [90], intégrée dans le framework *SymTA/S* [40] [50], est utilisée pour l'analyse des temps d'exécution des tâches dans les processeurs ou la communication en bus à l'intérieur d'un calculateur. Cette méthode couvre tous les aspects de l'exécution des tâches sur des calculateurs jusqu'à la transmission des résultats de ces tâches sur un réseau.

La méthode considère des événements qui parcourent un ensemble de composants. Ces événements déclenchent des tâches au sein des processeurs ou des transmissions dans le réseau. Dès qu'un événement entre dans un composant, celui-ci génère des délais qui se propagent de composant en composant. En fonction des délais calculés et des gigues déduites, une spécification formelle de l'impact de chaque composant sur l'évaluation globale des délais de bout en bout est établie. CPA a été formalisée pour les réseaux TSN avec différents modes de fonctionnement, préemptif [104] [86] et non préemptif [93] [103] [8] [105].

3.1.5 *La méthode des intervalles éligibles*

La méthode des intervalles éligibles [25] a été développée pour répondre aux besoins de détermination des délais de transmission dans un unique commutateur réseau configuré pour utiliser l'AVB (cf. 1.5.3) [26]. Cette méthode peut s'appliquer à des topologies réseau en étoile avec un seul commutateur central, ce qui peut être limitant pour des configurations plus complexes. Elle cherche à trouver des intervalles éligibles, c'est-à-dire des instants où une transmission du flux étudié peut commencer.

Cette méthode repose sur l'observation d'interférences. Une interférence est un événement qui empêche la transmission de la trame analysée. L'observation faite est que l'intervalle éligible apparaît toujours après la transmission de la trame qui devait précéder la transmission de la trame étudiée dans l'ordonnancement sans interférence. Il existe différents types d'interférences :

— les trames de plus haute priorité ;
— les trames de plus faible priorité ;
— les moments de recharge de crédit de la classe de trafic analysée.

La méthode calcule donc le délai de transmission pire cas sans interférence et additionne toutes les interférences possibles pour déterminer le temps pendant lequel la trame restera dans le commutateur.

La méthode a ensuite été étendue pour les réseaux TSN utilisant conjointement le *Time-Aware Shaper* et le *Credit Based Shaper* [75].

3.1.6 *Forward end-to-end delay Analysis (FA)*

La méthode *Forward end-to-end Analysis* (FA) est une méthode heuristique holistique détaillant un scénario pire cas de parcours d'une trame dans un réseau *Ethernet* commuté. Ce scénario décrit un cas où l'on charge un nœud au maximum de trames qui retardent la trame analysée. Nous pouvons ainsi déduire un délai pire cas dans le commutateur. Ce résultat est ensuite réutilisé pour calculer les délais pire cas dans les commutateurs suivants. Après avoir calculé tous les délais pire cas dans tous les commutateurs, nous pouvons déduire le délai de transmission pire cas d'un bout à l'autre du réseau.

FA a été proposée pour les réseaux AFDX fonctionnant avec les politiques de service FIFO [64] et FPFIFO [16]. La méthode a aussi été adaptée pour les réseaux AVB [15]. La version pour AVB est la plus polyvalente et permet d'analyser les réseaux utilisant les politiques de service FIFO et FPFIFO sans modification de la méthode.

Dans la suite de cette section, nous aborderons plus en détail les mécanismes sous-jacents à FA en mode FIFO, puis en ajoutant la notion de priorité et enfin avec l'ajout du trafic CBS.

Description de la méthode FA

La méthode FA définit, pour chaque entité du réseau, un scénario pire cas permettant d'obtenir une borne supérieure pire cas pour un flux donné. Il est considéré qu'un commutateur démarre son activité à $t = 0$ et l'analyse se poursuit jusqu'à la fin de la période d'activité générée par l'ensemble des flux concurrents à celui étudié. La fin de la période d'activité se conclut par la transmission de la trame analysée, ce qui donne la borne supérieure de transmission pire cas locale au commutateur étudié. Nous analysons donc le temps de transmission pour tous les instants $t \geq 0$.

Dans FA, pour l'analyse d'un flux donné, nous cherchons à maximiser le nombre de trames entrantes juste avant que la trame analysée n'arrive. Ce nombre peut être réduit en considérant la sérialisation des trames dans les liens physiques. La sérialisation est le principe selon lequel chaque bit de données arrive séquentiellement dans le commutateur. Il est donc physiquement impossible de recevoir toutes les trames en un seul bloc lors de l'initialisation du système, et la méthode FA en tient compte dans son scénario. Dans le contexte FIFO, la méthode maximise le nombre de trames entrant dans le commutateur juste avant l'instant d'arrivée de la trame analysée. Les trames arrivant après cet instant ne sont pas pertinentes dans ce contexte.

La méthode devient plus complexe concernant la politique de service FP/FIFO. Compte tenu des différents niveaux de priorité définis pour les trames, le scénario pire cas change. Les trames de même priorité se comportent comme dans le contexte FIFO, tandis qu'une seule trame de plus faible priorité et celles de plus haute priorité sont toujours plus prioritaires, tant que la trame analysée n'a pas commencé sa transmission. Il convient donc de toujours vérifier si de nouvelles trames de plus haute priorité sont arrivées dans le port de sortie pendant que d'autres trames étaient transmises. Cela conduit intrinsèquement à un calcul de point fixe permettant de récupérer le nombre de trames de plus haute priorité dans le commutateur après l'instant d'arrivée de la trame analysée.

Enfin, la méthode adaptée pour CBS ajoute une complexité supplémentaire avec son mécanisme de crédit [15]. CBS est une politique de service non conservative. Pour rappel, une politique de service non conservative ne garantit pas l'absence de temps creux. Il est possible, en CBS, qu'aucune des classes de trafic ne dispose de crédit, comme le montre la figure 2.8. La méthode doit pouvoir considérer ces temps d'inactivité pour les trames de même priorité et de priorité supérieure.

Des études comparatives de FA et du *Network Calculus* ont été conduites dans [110] et une explication concernant la possibilité que l'aspect de sérialisation de FA puisse induire un problème d'optimisme en suivant une modélisation différente de réseau par rapport à [15] a été donnée [109].

3.2 DIMENSIONNEMENT DES FILES D'ATTENTE DANS LES COMMUTATEURS RÉSEAUX

3.2.1 *Problématique*

Comme vu précédemment, les commutateurs ont plusieurs ports de sortie, et dans chacun de ces ports se trouvent plusieurs classes de trafic. Chaque classe de trafic dispose d'une file d'attente avec une taille de bits dédiée. Dans certains cas, la quantité de données admise dans une file d'attente peut être supérieure à l'espace disponible, entraînant la suppression de certaines données, ce qui constitue un comportement à risque lorsque les données sont critiques.

Afin de garantir l'absence de suppression de trames dans les files d'attente des ports de sortie, il est nécessaire de connaître la quantité maximale de données. Il existe deux approches pour quantifier l'occupation en espace des commutateurs : par nombre de bits et par nombre de trames. Cette quantification dépend de la gestion des trames par les normes réseaux. Par exemple, en avionique, les commutateurs sont configurés de manière statique et disposent d'emplacements mémoires physiques statiques permettant le stockage de trames. La figure 3.4 illustre cette différence. L'occupation des files d'attente peut être vue comme un écoulement fluide de bits entrants et sortants, ou comme une file d'attente composée de compartiments fixes se remplissant avec une seule trame. Ces compartiments ont une taille fixe adaptée à la plus grande taille de trames.

Pour garantir l'absence de suppression de trames dans les files d'attente des ports de sortie, des méthodes ont été développées afin de donner une borne maximale des buffers et ainsi dimensionner correctement les files d'attente. Nous allons les aborder dans la sous-section suivante.

3.2.2 *Méthodes*

Comme mentionné précédemment, il existe deux approches principales pour gérer les trames dans les files d'attente. Nous allons aborder le dimensionnement dans le cas de la gestion par bits et par trames.

Une approche pour dimensionner les files d'attente en termes de bits consiste à utiliser les résultats de calcul de délais de bout en bout obtenus par la méthode Network Calculus, FA ou TA, présentées dans la section précédente. En plus des délais de transmission pire cas, cette méthode permet également d'obtenir l'occupation pire cas des buffers. Sur la figure 3.3, la courbe horizontale

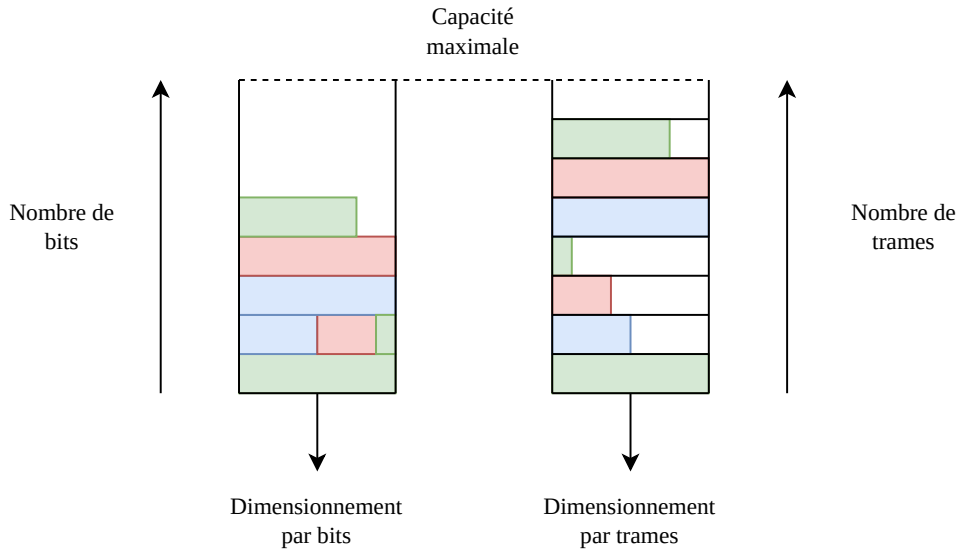

FIGURE 3.4 – Comparaison du dimensionnement des files d'attente en bits et en trames

maximale donne le temps maximal pour retransmettre la trame analysée, tandis que la courbe verticale indique son occupation maximale en bits.

Dans le cadre de l'approche par trames, une méthode a été développée [30] pour les réseaux AFDX avec différents niveaux de priorités (FP/FIFO) en utilisant la méthode des trajectoires. Cependant, une des limites de cette méthode est que toutes les trames du réseau ont une taille identique, quasi impossible dans les configurations réelles.

Des méthodes ont également été développées pour dimensionner les files d'attente en entrée des commutateurs AFDX, dans une version initiale [9] et dans une version probabiliste [10]. Néanmoins, ces travaux ne concernent pas les ports de sortie.

À notre connaissance, il n'existe pas de méthodes pour déterminer l'occupation pire cas des ports de sortie en nombre de trames avec des tailles de trames différentes.

3.3 GÉNÉRATION DE TABLES DE CONFIGURATION TAS

3.3.1 *Problématique*

Comme vu dans le paragraphe relatif à la description du *Time Aware Shaper* 2.5.3, le TAS repose sur l'exécution d'une table de configuration pour contrôler l'ouverture ou la fermeture de certaines classes de trafic à des instants donnés. Divers cas d'utilisation sont possibles, comme des créneaux de distribution de certains types de données à certains moments, ou encore la séparation de données.

Néanmoins, l'utilisation la plus commune du TAS est la retransmission des trames périodiques de haute priorité avec une garantie d'absence de congestion dans les *bridges*. Pour se faire, des créneaux temporels sont réservés à ces trames très prioritaires dans les *Gate Control lists* (GCL). Par exemple, une trame périodique arrivant dans un *bridge* ne subira aucune concurrence de la

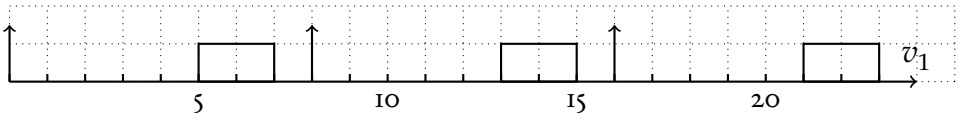

FIGURE 3.5 – Diagramme de Gantt représentant la relation entre l'arrivée d'une trame et sa transmission quand cette dernière dispose d'un offset

part d'autres trames et la garantie est donnée qu'aucune autre trame n'utilisera le lien de sortie à ce moment-là.

Prédire à quel instant une trame périodique arrive relève des problématiques de *model-checking* et est exponentiel en termes de complexité. À la place, le problème soulevé devient : comment organiser le trafic des trames périodiques prioritaires au niveau des *End-Stations* et comment configurer les *Gate Control Lists* sur les différents *bridges* ?

Nous allons d'abord étudier les solutions apportées à ce problème dans le cadre de la norme *TTethernet*, présentant des dispositions communes aux normes TSN concernant ce type de trafic. Ensuite nous explorerons les solutions pour les normes TSN. Les méthodes pour générer les tables de configuration se divisent en deux catégories : les méthodes optimales et les heuristiques.

3.3.2 *Méthodes optimales*

Dans cette section, nous allons aborder la génération des tables de configuration dans la norme *TTEthernet*, et ensuite dans les normes TSN. Le mode de fonctionnement concernant les trames périodiques de haute priorité pour *TTEthernet* et TSN est semblable. Par conséquent, nous ne détaillerons pas le mode de fonctionnement de *TTEthernet*.

La plupart des méthodes permettant d'obtenir des tables de calcul reposent sur les méthodes *Satisfiability Modulo Theories* (SMT) [11]. Elles relèvent du *model checking*. Ces méthodes de calcul formel se fondent sur l'expression de contraintes. Ces contraintes sont ensuite données à un solveur qui se charge de trouver la première solution respectant toutes les contraintes. Pour chaque trame contenue dans chaque flux du réseau, un ensemble de contraintes est défini.

Nous nous servons de [100], relatif à *TTEthernet*, et étendu avec [33] et [32] pour TSN, comme base de référence pour les explications. L'objectif de la méthode n'est pas de produire immédiatement une table de configuration, mais de produire des offsets pour chaque trame du réseau. Les offsets sont des temps qui imposent l'instant de départ d'une trame d'un commutateur, indépendamment de son instant d'arrivée.

Afin d'offrir une meilleure compréhension de cette notion, nous représentons sur la figure 3.5 un exemple illustrant ce mécanisme avec un diagramme de Gantt. Nous avons un flux qui transmet une trame de longueur 2 et de périodicité 8. Si la trame arrive dans un commutateur au temps 0 et que l'offset de la trame est fixé à 5, alors la trame partira à l'instant 5, sous condition qu'aucune autre trame ne soit en concurrence. À partir de ces offsets, il est possible de générer des tables de configuration.

La complexité de cette approche est exponentielle en temps pour trouver une solution et en espace pour stocker toutes les contraintes de trames. Elle est donc peu viable pour de larges configurations industrielles.

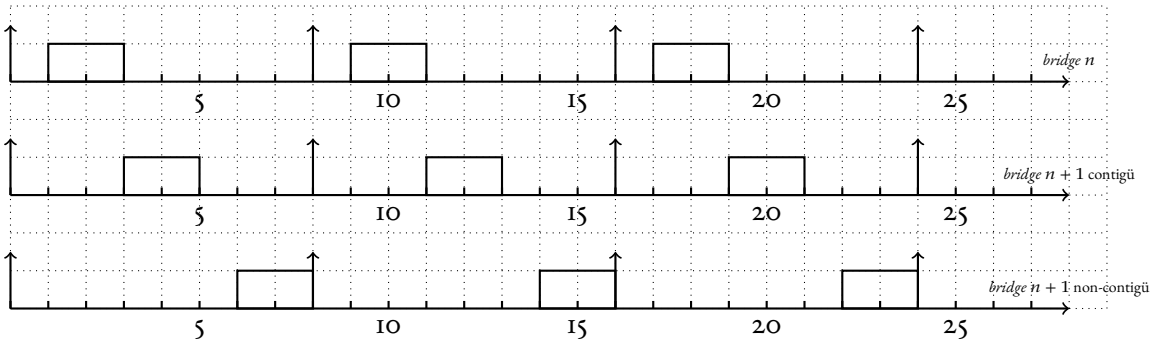

FIGURE 3.6 – Diagramme de Gantt montrant la différence entre un écoulement contigu et non-contigu

Les solutions trouvées sont optimales, mais ne sont pas nécessairement les meilleures concernant les délais de bout en bout. En effet, les contraintes formulées n'imposent pas que les différents offsets soient contigus d'un *bridge* à un autre. Nous le représentons visuellement sur la figure 3.6. Dans le *bridge n*, une trame est partie à l'instant 1. Dans le cadre d'une transmission contiguë, elle arrive entièrement à l'instant 3 et est immédiatement retransmise. Dans le cadre d'une transmission non contiguë, la trame peut partir du bridge suivant à n'importe quel instant après être arrivée dans le *bridge*.

Il est possible de réduire ces délais grâce à des méthodes OMT (*Optimization Modulo Theories*) [96]. Ces méthodes incluent, en plus des contraintes définies précédemment, une fonction objectif. Cette fonction indique quels sont les paramètres à minimiser lorsque des solutions sont trouvées. Néanmoins, cela ne change en rien l'aspect combinatoire de la recherche, et trouver une solution avec des délais de bout en bout minimaux peut prendre un temps exponentiel en fonction de la configuration.

Des méthodes basées sur la programmation linéaire sont également disponibles [95] [41]. Ces méthodes sont semblables aux méthodes OMT, mais ont la particularité d'utiliser des contraintes uniquement sous la forme de formules composées exclusivement d'entiers.

Des optimisations ont aussi été proposées pour accélérer la génération de solutions. Une approche par fenêtres a été proposée [89]. Elle isole un temps de trafic réservé aux trames prioritaires périodiques des autres. Cela permet de réduire l'intervalle de recherche de solutions valides.Cela, au coût de réduire la capacité de transmission des trames prioritaires périodique.

Un outil, permettant de générer simplement ces tables de configuration avec les méthodes SMT, a été proposé, TSNSCHED[94]. Il offre une solution clé en main pour pouvoir configurer les GCL sans avoir à s'occuper des offsets individuellement ou à manipuler le système de contraintes. Seule la topologie du réseau ainsi que les différents flux sont nécessaires.

3.3.3 *Méthodes heuristiques*

Les méthodes heuristiques développées pour résoudre la problématique de génération d'offsets dans ce contexte sont pour la plupart des méthodes métaheuristiques. L'objectif de ces méthodes est de guider les méthodes exactes à trouver une solution minimisant une fonction d'objectif, par exemple les délais de bout en bout. Pour ce faire, une métaheuristique peut rechercher des

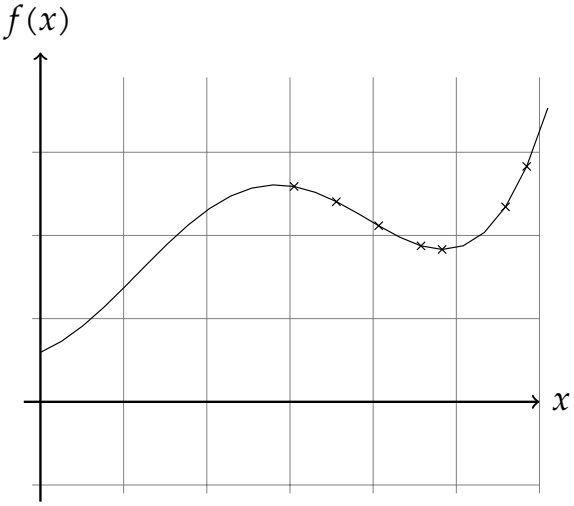

FIGURE 3.7 – Représentation du principe de fonctionnement d'une métaheuristique

solutions exactes proches d'une solution exacte trouvée, ou sinon explorer des solutions avec des paramètres aléatoires.

Nous représentons cela sur la figure 3.7. La courbe représente la fonction objectif à minimiser et les différents points représentent là où la méthode métaheuristique a cherché. Celle-ci a exploré autour d'un minimum local et a réussi à trouver une solution à plus faible coût. Néanmoins, la métaheuristique n'a pas pu trouver la solution minimale qui se trouvait à $x = 0$.

La méthode décrite dans [88], établie sur GRASP [43], décrit une façon d'essayer de trouver des solutions telles que les délais de transmission de bout en bout soient moins élevés. Cette méthode se déroule en deux étapes : trouver une solution respectant toutes les contraintes et ensuite rechercher des solutions valides avec des paramètres proches. La méthode inspecte aléatoirement les solutions proches. Une évaluation de différentes métaheuristiques permettant l'amélioration des calculs issus de model-checking a aussi été proposée [78].

Au meilleur de nos connaissances, il n'existe pas de méthode heuristique permettant la génération directe d'offsets ou de tables de configuration dans le cadre du *Time Aware Shaper*.

3.4 CONCLUSION

Dans cet état de l'art, nous avons abordé des méthodes permettant d'apporter des outils pour s'assurer du déterminisme des réseaux temps-réel sur les différents aspects vu dans l'introduction.

Nous récapitulons l'état du développement des méthodes d'évaluation des délais de bout en bout en fonction de la politique de service utilisée dans la table 3.1. Les méthodes adaptées pour TSN sont représentées par ✔️ et celles non-adaptées par ❌.

Concernant le dimensionnement des files d'attente en termes de trames, nous n'avons pas réussi à trouver des méthodes correspondantes. Nous en décrirons une dans ce manuscrit.

Enfin, dans le dernier chapitre des contributions, nous décrirons des problèmes qui peuvent survenir dans les méthodes développées pour construire des tables de configuration TAS, nous

TABLE 3.1 – Table récapitulative des différentes contributions dans les réseaux TSN

	FIFO	FPFIFO	CBS/AVB	TAS	CBS + TAS
Forward end-to-end delay Analysis	✓	✓	✓	✗	✗
Network Calculus	✓	✓	✓	✓	✓
Intervalles éligibles	✗	✗	✓	✗	✓
Compositional Performance Analysis	✓	✓	✓	✓	✓

proposerons aussi une heuristique permettant la génération de tables de configuration, non pas par trames mais par flux.

Deuxième partie

CONTRIBUTIONS

4

LE DIMENSIONNEMENT DES BUFFERS DE COMMUTATEURS EN NOMBRE DE TRAMES

Dans ce chapitre, nous allons étudier le dimensionnement des files d'attente en nombre de trames. Le contenu de ce chapitre est actuellement en soumission pour le journal *Real-Time Systems*.

4.1 INTRODUCTION

Les commutateurs réseaux sont les points centraux des normes réseaux commutés. Certaines de ces normes basées sur *Ethernet* sont conçues afin de répondre à des exigences de déterminisme, garantissant des bornes dans le pire cas sur la latence de bout en bout pour les messages et garantissant qu'il n'y a aucune perte de trame. Le calcul d'un délai de transmission exact dans le pire cas est un problème NP-difficile au sens fort [1] et plusieurs approches ont été proposées pour déterminer des bornes supérieures garanties à la place, telles que, le Network Calculus (cf. 3.2).

Pour certifier les réseaux temps réel, il faut garantir qu'aucune trame ne sera perdue durant le parcours des données en son sein. Avec l'utilisation de liens full-duplex (bidirectionnels) de la norme Ethernet [54], nous avons la garantie qu'aucune trame ne peut rentrer en collision avec une autre. Les endroits où des trames peuvent se perdre sont dans les points de multiplexage (c'est-à-dire les ports de sortie du commutateur ou des *End Stations*). Les caractéristiques des flux d'un réseau peuvent être très variables, ce qui peut avoir un impact sur l'occupation des trames dans les files d'attente. Lorsqu'une rafale de trames entrantes arrive dans le point de multiplexage, celui-ci peut ne pas être assez grand pour stocker toutes les trames et par conséquent des trames peuvent être supprimées des files d'attente.

Afin d'aider le concepteur de réseau à choisir des commutateurs et configurer les flux qui y circuleront, garantir que tous les flux qui circulent dans le réseau ne soient pas supprimés dans les ports de sortie des commutateurs est important. De plus, à des fins de certification, il peut être demandé de donner une preuve formelle qu'aucune trame ne soit supprimée pendant l'exécution du système.

Nous avons vu dans le chapitre de l'état de l'art (cf. section3.2) que l'occupation du buffer dans le pire cas peut être déterminée de deux manières : en nombre de bits ou en nombre de trames. Cette estimation permet de dimensionner les buffers afin d'éviter que ceux-ci débordent. Une façon naïve de déterminer l'occupation du pire cas en nombre de trames consiste à diviser l'occupation maximale en nombre de bits obtenue par une méthode d'évaluation des délais de bout en bout pire cas, par la taille minimale des trames. Néanmoins, cela conduit à une évaluation pessimiste de l'occupation des *buffers*.

Dans ce chapitre, nous proposons la méthode *Buffer Dimentionning per Frame* (BDF) dédiée spécifiquement à déterminer la taille maximale d'occupation des files d'attente en nombre de

trames. Notre approche peut être appliquée à tout type de réseau de paquets utilisant une politique de service conservative (c'est-à-dire qu'un port de sortie n'est jamais inactif tant que des trames sont présentes dans le buffer). La méthode BDF est basée sur :

— la topologie du réseau ;
— les contrats de trafic réseau ;
— une méthode de calcul du délai de bout en bout dans le pire cas.

Les contrats de trafic permettent de déterminer un scénario pire cas des trames produit par chaque flux v_i du nœud source duquel ils sont émis jusqu'au nœud destination. Chaque contrat est décrit par une taille de trame maximale F_i^{max} (en bits) et une période minimale entre deux générations successives T_i (sporadique).

Nous allons dans la prochaine section décrire les motivations qui conduisent à cette contribution. Les détails du modèle de réseau sont donnés dans la section 4.3. Une description détaillée du calcul du BDF est détaillée dans la section 4.4. Enfin, BDF est évaluée sur une étude de cas et sur une configuration industrielle.

4.2 MOTIVATIONS

La plupart des méthodes de calcul du délai de bout en bout, dans le pire cas proposent de dériver l'occupation maximale des files d'attente en nombre de bits directement à partir de l'analyse des délais de traversée d'une trame dans un port de sortie (cf. 3.2). Néanmoins, nous pouvons prouver que les occupations pire cas des files d'attente en nombre de bits et en nombre de trames ne coïncident pas toujours. Nous le prouvons à l'aide d'un contre-exemple.

Considérons la configuration représentée figure 4.1.a, avec un seul nœud h et trois flux v_1, v_2 et v_3, entrant dans h via des liens entrants séparés. Les flux sont caractérisés par leurs temps de transmission respectifs $C_1^h = C_2^h = 10\mu s$ et $C_3^h = 30\mu s$ ainsi que par leurs périodes $T_1 = T_2 = 30\mu s$ et $T_3 = 100\mu s$. Par simplicité et sans perte de généralité, nous considérons que le débit du lien de sortie de h est égal à 1 bit/μs. Ainsi, le retard accumulé en bits est équivalent au retard accumulé en μs.

Comme le montre la figure 4.1.b, au temps $t = 0\mu s$, une trame par flux arrive, une file d'attente composée de trois trames, et l'occupation en bits est égale à $C_1^h + C_2^h + C_3^h = 50$ bits. Si la trame 3 est servie en premier, elle quitte le nœud à $t = 30\mu s$. Néanmoins, selon leurs périodes, dans le pire cas, d'autres trames de v_1 et v_2 arrivent à $t = 30\mu s$. Le retard accumulé dans le port de sortie est alors composé de quatre trames, mais pour une taille totale de seulement 40 bits ($2 \times C_1^h + 2 \times C_2^h$). Ainsi, le nombre maximal de trames en attente est observé à $t = 30\mu s$ alors que l'occupation maximale en bits est obtenue à $t = 0\mu s$. Ainsi, le dimensionnement dans le pire cas en nombre de bits ou de trames sont deux problèmes distincts.

De plus, si nous considérons une approche naïve, consistant à diviser le retard accumulé maximal en bits (50 bits à $t = 0\mu s$) par la taille minimale des trames (10 bits), le dimensionnement des files d'attente résultant (5 trames) serait pessimiste, induisant un certain surdimensionnement des files d'attente dans les systèmes ne tolérant pas la perte de trames.

Par conséquent, nous proposons l'approche BDF pour calculer une borne supérieure garantie sur le dimensionnement pire cas en trames, lorsque l'allocation statique de la mémoire est utilisée.

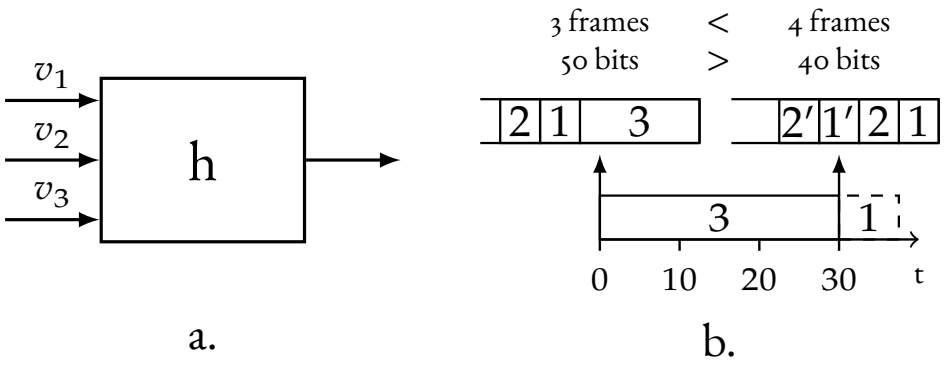

FIGURE 4.1 – Le retard accumulé maximal en nombre de bits et de trames ne coïncide pas toujours.

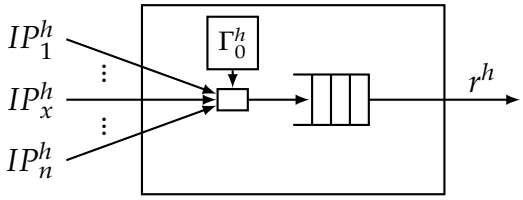

FIGURE 4.2 – Modèle d'un nœud h avec n liens entrants et un débit de sortie r_h.

4.3 MODÈLE RÉSEAU

Le réseau est composé d'un ensemble de nœuds (sans hypothèse de synchronisation) notés S et d'un ensemble de flux sporadiques notés Γ. Soit Γ_h l'ensemble des flux traversant un nœud donné h de S. Les nœuds représentent les points de multiplexage du réseau. Un nœud h est composé d'un ensemble de n liens entrant notées $IP_1^h, \ldots, IP_n^h$ alimentant une seule file d'attente, en sortie. Le nœud sert les trames à un débit constant r_h en utilisant une politique de service conservative (par exemple FIFO). Le modèle d'un nœud est illustré à la figure 4.2.

Un flux sporadique $v_i \in \Gamma_h$ est unidirectionnel et *unicast*. Il est généré dans un nœud source $first_i$ et traverse une séquence de nœuds h selon un chemin statique prédéfini jusqu'à ce qu'il atteigne son nœud destination $last_i$. L'ensemble des flux générés dans un nœud h est noté Γ_0^h et l'ensemble des flux entrants dans h via le port d'entrée IP_x^h est noté Γ_x^h. Ainsi, nous avons : $\Gamma^h = \bigcup_{x=0}^{n} \Gamma_x^h$.

Chaque flux v_i est contraint depuis son *End-Station* d'émission dans le réseau par un contrat de trafic défini par :

— une taille de trame maximale $Fmax_i$ et minimale $Fmin_i$;
— une période T_i entre deux générations successives.

Le temps de transmission maximal pour une trame de taille F_i^{max} dans un nœud h est $C_i^h = \frac{Fmax_i}{r_h}$. Nous désignons par $Smin_i^h$ (resp. $Smax_i^h$) le délai minimum (resp. maximum) subi par une trame de v_i pour atteindre un nœud h depuis son temps de génération dans $first_i$, l'ES émettrice. $Smin_i^h$ et $Smax_i^h$ peuvent être déterminés par toute méthode d'analyse de délai de bout en bout dans le pire cas, et sont considérés comme une entrée, alimentant notre approche de dimensionnement des files d'attente. Enfin, la gigue dans le pire cas de v_i dans un nœud h est

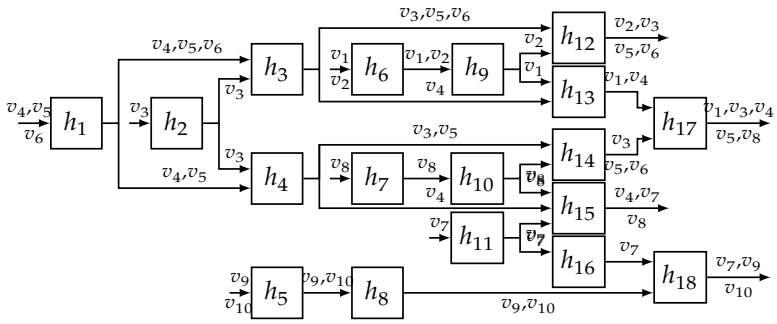

FIGURE 4.3 – Un exemple de configuration de réseau.

donnée par $J_i^h = Smax_i^h - Smin_i^h$. Un exemple de configuration composée de 18 nœuds et de 10 flux générant des trames est illustré à la figure 4.3.

4.4 DIMENSIONNEMENT DES FILES D'ATTENTE EN NOMBRE DE TRAMES

Dans cette section, nous détaillons notre approche BDF pour calculer une borne supérieure sur le nombre simultané maximal de trames dans un port de sortie au niveau de sa file d'attente. Cette approche repose sur la connaissance préalable de :

— une topologie de réseau ;
— le contrat de trafic de chaque flux à son point d'entrée dans le réseau (taille de trame et période sporadique) ;
— une borne supérieure de délai pire cas pour chaque flux, qui peut être fournie par toute méthode d'analyse réseau.

4.4.1 *Principe*

À notre connaissance, il n'existe aucun travail existant décrivant le scénario pire cas d'arrivée de trames maximisant son nombre dans une file d'attente, avec une politique de service non préemptive et conservative. Pour illustrer la complexité de la détermination d'un tel scénario, considérons à nouveau l'exemple de la figure 4.3 avec des durées $C_1^h = 20\mu s$, $C_2^h = 10\mu s$ et $C_3^h = 30\mu s$, des périodes toutes égales à $100\mu s$ et une politique de service FIFO.

Un scénario d'arrivée possible, avec une trame de chaque flux arrivant au plus tôt au temps $0\mu s$, est illustré à la figure 4.a. Le départage en cas d'arrivée simultanée est l'indice de trame le plus petit. Supposons qu'une nouvelle trame de v_3 (resp. v_1) arrive au plus tôt à $t = 30\mu s$ (resp. $40\mu s$), en raison de la gigue. Dans ce scénario, le retard accumulé maximal en nombre de trames, mesuré à la fois à $t = 0\mu s$ et $t = 40\mu s$, est d'au moins 3.

Mais, retarder l'arrivée de la trame 2 de $30\mu s$ (voir figure 4.b) conduit à un cas encore pire de 4 trames à $t = 40\mu s$. Cet exemple illustre qu'un instant critique à $t = 0\mu s$ ne maximise pas toujours le nombre de trames simultanément présentes dans un nœud. Cependant, analyser tous les scénarios d'arrivée possibles pour trouver le pire cas conduirait à une explosion combinatoire. Ainsi, nous proposons un scénario (éventuellement pessimiste) fournissant une borne supérieure

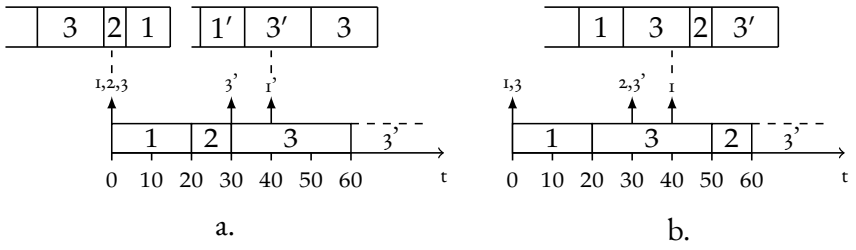

FIGURE 4.4 – Scénarios d'arrivée dans une fille d'attente FIFO.

sur l'occupation des files d'attente en nombre de trames. Le principe est le suivant : maximiser le nombre de trames entrantes au plus tôt, en fonction des contrats de trafic et des délais du réseau, tout en minimisant le nombre de trames transmises, en fonction de leur longueur et de leur ordre de transmission.

Nous allons détailler dans les sous-sections suivantes : le scénario pire cas pour les trames entrantes, le scénario pour minimiser le nombre de trames servies par un port de sortie, une solution pour réduire le pessimisme des deux scénarios précédents et enfin, nous fournirons l'algorithme détaillé de BDF.

4.4.2 *Trames entrantes*

Le nombre maximal de trames entrantes provenant d'un flux $v_i \in \Gamma_h$ dans un nœud h, durant un intervalle de temps de longueur t (avec $t \geq 0$), peut être obtenu en utilisant la *Request Bound Function* (RBF). Dans [17], la RBF est introduite dans un contexte FIFO, mais le principe peut être étendu à toute politique non préemptive et conservative.

Connaissant le contrat de trafic v_i, le nombre maximal de données générées sur son nœud source ($first_i$) pendant un intervalle de temps fermé $[t_0, t_1]$ est : $\left(1 + \left\lfloor \frac{t_1 - t_0}{T_i} \right\rfloor\right) C_i^{first_i}$.

Cependant, les trames arrivant dans tout nœud suivant h pendant un intervalle $[t_0, t_1]$ ont été générées dans l'intervalle étendu $[t_0 - Smax_i^h, t_1 - Smin_i^h]$ dans $first_i$. Ainsi, la durée générée par v_i entrant dans h pendant un intervalle $[0, t]$ est :

$$rbf_i^h(t) = \left(1 + \left\lfloor \frac{t + J_i^h}{T_i} \right\rfloor\right) C_i^h$$

$rbf_i^h(t)$ maximise l'arriéré des trames [17]. En raison de la gigue dans les nœuds précédents, au temps 0, une rafale de k_i^h trames provenant de v_i arrive dans h :

$$rbf_i^h(0) = \underbrace{\left(1 + \left\lfloor \frac{J_i^h}{T_i} \right\rfloor\right)}_{k_i^h \text{ frames}} C_i^h \tag{4.1}$$

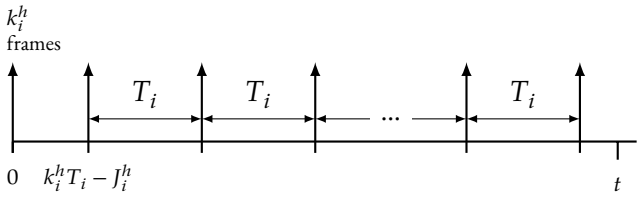

FIGURE 4.5 – Temps d'arrivée minimaux des trames de $v_i \in \Gamma_h$ dans une fille d'attente utilisant une politique de service conservative h.

où $k_i^h \in \mathbb{N}^*$ et est défini par $(k_i^h - 1)T_i \leq J_i^h < k_i^h T_i$. Si cette rafale se produit à l'instant t = 0µs, alors l'arrivée de la trame suivante ne peut pas être antérieure à l'instant $t = k_i^h T_i$ - J_i^h. Enfin, toutes les autres trames arrivent périodiquement tous les T_i. Ce scénario (voir figure 4.5) maximise au plus tôt, à partir de t = 0µs, le nombre de trames entrantes pour chaque flux $v_i \in \Gamma_h$.

Nous devons maintenant déterminer le nombre minimal de trames quittant h pendant la même période afin d'obtenir le nombre de trames pouvant être dans le port de sortie NbF_h.

4.4.3 *Trames sortantes*

Déterminer l'ordre exact de départ des trames qui maximisera le retard accumulé en nombre de trames, engendrerait une explosion combinatoire. Par conséquent, nous proposons d'utiliser comme borne inférieure pour toute politique non préemptive et conservative [l'algorithme Longest Processing Time First (LPT) [22]]. LPT à chaque instant de décision, transmet la trame ayant le temps de transmission le plus long parmi les trames en attente dans la file d'attente. Nous prouvons que LPT est optimal pour notre objectif dans le théorème suivant.

Théoréme 4.4.1. *Étant donné un ensemble de flux indépendants, générant des trames avec des temps d'arrivée arbitraires sur un nœud h, l'algorithme de service LPT est optimal parmi les politiques non préemptives et conservatives pour minimiser le nombre de trames transmises par h pendant tout intervalle [0, t] ($t \geq 0$).*

Démonstration. La preuve repose sur un argument d'échange, analogue à celui utilisé par Dertouzos [38] concernant l'optimalité de *Earliest Deadline First*, pour trouver un ordonnancement réalisable dans les systèmes monoprocesseurs.

Considérons qu'un nœud h sert les trames avec une politique de service non préemptive et conservative A, différente de LPT. À tout instant t, nous désignons par $\sigma(t)$ (resp. $lpt(t)$) la trame transmise par A (resp. LPT). De plus, $t_{lpt(t)}$ désigne le temps choisi par A pour démarrer la transmission de la trame $lpt(t)$. Les temps de transmission des trames dépendent uniquement du débit de sortie r_h et sont indépendants de la politique de service. Puisque A est différent de LPT, il existe un instant où la décision prise par A et LPT diffère. Nous notons par t_1 ($t_1 \geq 0$) le premier instant, tel que $\sigma(t_1) \neq lpt(t_1)$. Nous prouvons qu'échanger la transmission de $lpt(t_1)$ et $\sigma(t_1)$ ne peut pas augmenter le nombre de trames transmises par A.

L'échange entre la transmission de $lpt(t_1)$ et $\sigma(t_1)$ est illustré à la figure 4.6. Dans la figure 4.6.a, l'algorithme A sert la trame $\sigma(t_1)$ (trames hachurées) au temps t_1 et $lpt(t_1)$ (trame grise) au temps $t_{lpt(t_1)}$ alors que la transmission de ces trames est intervertie dans la figure 4.6.

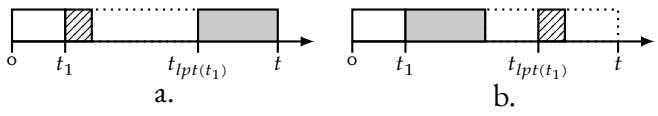

FIGURE 4.6 – Optimalité de LPT pour minimiser le nombre de trames transmises.

Pendant [o, t] (où t désigne la fin de la transmission de la trame $lpt(t_1)$ par A), la même durée (zone en pointillés) est disponible pour transmettre d'autres trames dans les deux cas. De plus, la zone en pointillés est découpée en deux dans le cas (b), ce qui peut conduire à transmettre encore moins de trames qu'avec A.

Par récurrence, à chaque instant t_1 ($t_1 \geq o$) tel que $\sigma(t_1) \neq lpt(t_1)$, nous échangeons la transmission de ces deux trames afin d'être conforme à LPT. À la fin de la transmission de chaque trame intervertie, le nombre de trames transmises en utilisant A est toujours supérieur ou égal à celui obtenu après échange. Par conséquent, LPT minimise le nombre de trames transmises.

□

Ainsi, les fonctions RBF présentées dans cette sous-section définissent, à tout instant t, une borne supérieure du nombre de trames à transmettre. LPT donne une borne inférieure des trames transmises pendant la même période, quelle que soit la stratégie de service. La différence entre le nombre de trames entrantes et sortantes donne une borne supérieure pour le nombre de trames en attente. Cependant, ce résultat peut être amélioré en considérant l'effet de sérialisation des flux.

4.4.4 *Effet de sérialisation*

Dans les réseaux de paquets, les trames partageant un même lien physique sont sérialisées : elles ne peuvent pas arriver simultanément au nœud suivant. Elles sont séparées par au moins le temps de transmission de la trame précédente. Cela implique que le débit de trames entrantes calculé avec les RBF dans la sous-section 4.4.2 est surestimé, car il ne tient pas compte de cet effet de sérialisation.

Selon notre modèle (voir figure 4.2), les flux $v_i \in \Gamma_0^h$ ne sont pas sérialisés dans le nœud h, car h correspond à leur nœud source. En revanche, tous les flux arrivants d'un nœud précédent h_x via un même lien entrant IP_x^h sont sérialisés. Comme il n'y a aucune garantie que deux trames, simultanément prêtes à être envoyées dans h_x, seront toujours transmises dans le même ordre, déterminer un scénario dans le pire cas nécessiterait une recherche exhaustive, conduisant à une explosion combinatoire.

Pour résoudre ce problème, notre solution est établie à partir du concept de période d'activité et de temps d'inactivité.

Définition 4.4.1. Dans un nœud h, une période d'activité [70] est un intervalle entre deux temps d'inactivité consécutifs. Un temps d'inactivité est un instant où toutes les trames précédemment arrivées ont été servies à cet instant (aucun retard accumulé restant). Un temps d'inactivité x fait référence à un temps d'inactivité sur le noeud h par rapport au lien entrant IP_x^h, et une période d'activité x à la période d'activité correspondante entre deux temps d'inactivité x.

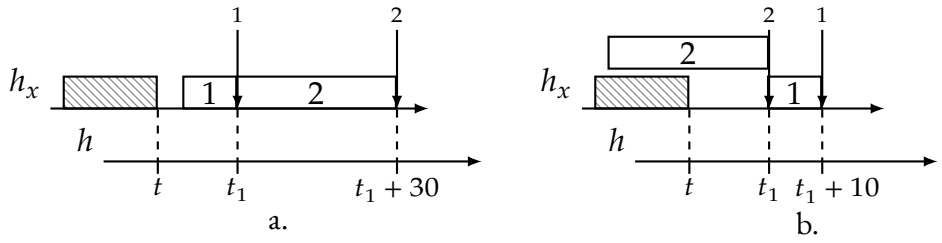

FIGURE 4.7 – Scénario d'arrivée maximisant le nombre de trames entrantes à un temps d'inactivité x.

Nous déterminons la trame à envoyer du nœud h par le lien x (h_x) au nœud h étudié (via IP_x^h) pendant une période d'activité x, afin de maximiser le nombre de trames transmises, en tenant compte de l'effet de sérialisation. Tout d'abord, nous considérons uniquement le temps de début d'inactivité x (Lem. 1), puis le reste de la période d'activité (Lem. 2).

Lemme 4.4.1. *Soit h et IP_x^h son lien entrant depuis le nœud h_x. Lorsqu'un temps d'inactivité x se produit, afin de maximiser le nombre de trames entrantes, la plus grande trame parmi celles prêtes à être transmises doit être sélectionnée*

Démonstration. Ce lemme peut être prouvé à l'aide d'un contre-exemple. Considérons la configuration représentée sur la figure 4.7 où deux flux v_1 et v_2 se rencontrent dans un nœud h. Les temps de transmission de ces flux sont, respectivement, dans h_x, C_1^h = 10µs et C_2^h = 30µs avec des périodes infinies. Soit $t \geq 0$ le début du temps d'inactivité x pour h. Par définition, au temps t, la charge de travail (zone hachurée sur la figure 7) a déjà été entièrement transmise de h_x à h, et le lien entrant IP_x^h est inactif. Soit $t_1 > t$ le prochain instant où la charge de travail augmentera à nouveau.

Supposons qu'au temps t_1, les trames 1 et 2 (générées respectivement par v_1 et v_2) peuvent arriver dans h via ce lien, selon leurs RBF. En raison de l'effet de sérialisation, l'une des trames doit être envoyée avant l'autre. Selon SPT, la plus petite trame 1 doit être sélectionnée en premier, tandis que la trame 2 pourrait arriver au temps t_1 + 30 (voir figure 7.a). Mais ce scénario n'est pas nécessairement le pire cas. En effet, lorsque la trame 2 est privilégiée, il est possible d'avancer l'arrivée de la trame suivante (celle de la trame 1, qui est plus courte), dès t_1 + 10µs (voir figure 7.b). Il convient de noter que la faisabilité de cet échange dépend de la durée de la période d'inactivité $[t, t_1]$. Dans notre exemple, le temps de transmission de la trame 2 est plus long que t_1 - t, le temps d'inactivité disponible. Même si ce scénario est irréaliste, il introduit du pessimisme, mais reste valide pour une analyse pire cas. □

Nous allons, dans le lemme suivant prouver que l'algorithme SPT (Shortest Processing Time First) [22] maximise le nombre de trames pouvant arriver dans h via un lien sérialisée depuis un nœud h_x pendant toute période d'activité $]t_1, t_2]$ $(t_1 < t_2)$.

Lemme 4.4.2. *La politique de service SPT maximise le nombre de trames transmises par un nœud pendant toute période d'activité $]t_1, t_2]$ $(t_1 < t_2)$ parmi toutes les politiques de service non préemptives et conservatives.*

Démonstration. La preuve est semblable à celle détaillée pour LPT dans Th 4.4.1 : avec un argument d'échange (Dertouzos [38]). Nous prouvons que tout algorithme A différent de SPT ne peut pas servir plus de trames dans le même intervalle de temps. □

Nous proposons dans le théorème 4.4.2 un scénario pire cas maximisant le nombre de trames entrant dans un nœud h établi à partir d'une variante de SPT, appelée SPT*, tenant compte de l'effet de sérialisation pour les trames arrivant via un même lien entrant.

Théoréme 4.4.2. *Soit h un nœud avec une politique d'ordonnancement non préemptive et conservative. Le nombre de trames entrant dans un nœud h, au travers d'un lien entrant* IP_x^h*, pendant tout intervalle de temps [o, t] (avec* $t > o$*), est maximisé si l'algorithme LPT est appliqué au début de l'intervalle et à chaque temps d'inactivité x, et appliquer SPT pour toutes les autres trames suivantes.*

Démonstration. La preuve de ce théorème est une conséquence directe des lemmes 4.4.1 et 4.4.2 □

Nous détaillons le calcul de l'occupation maximale des files d'attente en nombre de trames avec la méthode BDF dans la section suivante.

4.4.5 *Calcul de* NbF^h

Nous appelons NbF^h l'occupation de la file d'attente dans le pire cas pour un nœud h, comptée en nombre de trames par BDF (cf. l'algorithme 2), fondée sur SPT* (voir l'algorithme 1) pour les trames entrantes et LPT pour les trames sortantes. Nous commencerons à décrire le fonctionnement de SPT* et son intégration dans BDF dans un second temps.

L'algorithme 1 présente le tracé de la courbe du maximum de trames entrées dans un nœud h par un lien entrant IP_x^h pour un instant t donné. Il requiert en entrée une description des flux qui traverseront ce lien et en particulier des temps de transmission pire-cas.

- de la ligne 1 à 7 constituent l'initialisation des différents éléments nécessaires pour le bon fonctionnement de l'algorithme dont les trames qui ont fini de rentrer dans le commutateur (*ArrFrames*), le nombre de trames admises dans le nœud pour un instant t ($W_x^h(t)$), le temps de transmission qui passe dans le lien (C_i^h) ainsi que le calcul de la gigue pour tous les flux ;
- de la ligne 9 à 13, l'algorithme se charge de déterminer quelles sont les dernières trames admises et pas encore traitées dans le nœud h ;
- le bloc de code entre la ligne 14 et 18 permet de compter la première plus grande trame après un temps d'inactivité ;
- celui entre la ligne 19 et 21, permet de comptabiliser les trames analysées ;
- le bloc de code entre la ligne 22 et 26 permet de compter SPT dans le cadre nominal ;
- enfin, la dernière ligne permet de trouver l'instant suivant où une nouvelle trame arrive.

Input : Γ_x^h l'ensemble des flux arrivant au nœud h au travers du lien IP_x^h
pour chaque flux $v_i \in \Gamma_x^h : Fmax_i, T_i, Smin_i^h, Smax_i^h$
Result : La courbe $W_x^h(t)$

begin
  $ArrFrames \leftarrow \emptyset$
  $W_x^h(t) \leftarrow 0$
  $C^{h_x} \leftarrow 0$
  $t \leftarrow 0$
  foreach *flow* $v_i \in \Gamma_x^h$ **do**
    $J_i^h \leftarrow Smax_i^h - Smin_i^h$
  Loop
    $FrameSet \leftarrow \emptyset$
    foreach *flow* $v_i \in \Gamma_x^h$ **do**
      $k \leftarrow \left(1 + \left\lfloor \frac{t+J_i^h}{T_i} \right\rfloor\right)$
      $FrameSet \leftarrow FrameSet \bigcup \{f_{i,1} \ldots f_{i,k}\}$
    $FrameSet \leftarrow FrameSet \backslash ArrFrames$
    if $C^{h_x} = 0$ ***and*** $FrameSet \neq \emptyset$ **then**
      $f \leftarrow MaxFrameSize\ (FrameSet)$
      $C^{h_x} \leftarrow TransTime(f, h_x)$
      $ArrFrames \leftarrow ArrFrames \bigcup \{f\}$
      $FrameSet \leftarrow FrameSet \backslash \{f\}$
    if $C^{h_x} \neq 0$ **then**
      $W_x^h(t) \leftarrow W_x^h(t) + 1$
      $C^{h_x} \leftarrow 0$
    if $FrameSet \neq \emptyset$ **then**
      $f \leftarrow MinFrameSize\ (FrameSet)$
      $C^{h_x} \leftarrow TransTime(f, h_x)$
      $ArrFrames \leftarrow ArrFrames \bigcup \{f\}$
      $t \leftarrow t + C^{h_x}$
    else
      $t \leftarrow \min \left\{ t_i \leftarrow -J_i^h + nT_i | n \geq 0, t_i > t, v_i \in \Gamma_x^h \right\}$

Algorithme 1 : Description de l'algorithme SPT* pour un nœud h et un lien entrant IP_x^h.

Input : Pour chaque flux $v_i \in \Gamma : Fmax_i, T_i$
Pour chaque noeud $h : r^h$ avec $U^h < 1$
Pour chaque flux $v_i \in \Gamma^h : Smin_i^h, Smax_i^h$
Result : NbF^h pour chaque noeud h

1 **begin**
2 **foreach** *node* h **do**
3 **foreach** *flow* $v_i \in \Gamma^h$ **do**
4 $J_i^h \leftarrow Smax_i^h - Smin_i^h$
5 $NbF^h \leftarrow 0$
6 $t \leftarrow 0$
7 $InFrames \leftarrow 0$
8 **while** $InFrames \geq LPT^h(t)$ **do**
9 $InFrames \leftarrow \sum_{v_i \in \Gamma_0^h} \left(1 + \left\lfloor \frac{t+J_i^h}{T_i} \right\rfloor\right) + SPT^*(t)$
10 $NbF^h \leftarrow \max\left(NbF^h, InFrames - LPT(t)\right)$
11 $t \leftarrow t + 1$

Algorithme 2 : Calcul de NbF^h en utilisant BDF

Les entrées de l'algorithme 2 sont l'ensemble des flux appartenant à Γ_h traversant un nœud h en faisant attention à ce que la condition de charge soit bien respectée. De plus, des résultats de calcul pire cas sont nécessaires pour chaque flux $(Smin_i^h, Smax_i^h, J_i^h)$.

— pour chaque nœud où le nombre de trames est calculé, l'intégralité des gigues est déduite à partir des résultats des méthodes d'évaluation des délais pire cas (l.3-5). Il s'ensuit une initialisation des variables qui seront nécessaires dans la suite de l'algorithme (l.5-7);
— nous avons, le nombre de trames maximal que la file d'attente du nœud (l.5), le temps d'analyse (l.6) ainsi que le nombre de trames entrant dans le nœud à l'instant t (l.7);
— nous entrons ensuite dans la boucle de calcul qui vérifie à chaque fois si la quantité de trames en entrée est toujours supérieure à la quantité de trames émises (l.8). Pour $t = 0$, nous savons que cette condition est valide pour tous nœuds avec au moins un flux;
— au sein de la boucle est calculé le nombre de trames entrantes avec SPT* pour les liens entrants et avec l'expression de RBF pour les trames produites par le nœud. Ce résultat est stocké dans la variable $InFrames$ (l.10). Puis le nombre de trames maximal dans le nœud est actualisé en récupérant le maximum entre le nombre de trames maximal déjà calculé ou la différence entre les trames en entrée et en sortie (l.11).

La validité de la condition d'arrêt est prouvée dans 4.4.3

Théoréme 4.4.3. *Soit h un nœud avec une politique non préemptive et conservative. L'algorithme 2 atteindra nécessairement sa condition d'arrêt si la charge locale* $U_h = \sum_{v_i \in \Gamma_h} \frac{C_i^h}{T_i} < 1$.

Démonstration. Le nombre global de trames dans le port de sortie d'un nœud h est borné supérieurement par les RBF de chaque flux $v_i \in \Gamma_h$. Selon le scénario d'arrivée décrit à la figure 5, après une rafale initiale commençant au temps 0, le modèle d'arrivée des trames devient périodique

Flow	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
$Fmax_i$ (kb)	1	2	1	1	2	2.2	6.4	2.2	1	1
T_i (µs)	60	120	100	80	60	60	120	100	80	80

TABLE 4.1 – Caractéristiques des flux utilisés pour le cas d'étude.

(voir figure 4.5). Ainsi, si $U_h < 1$, il y aura au moins un temps d'inactivité pendant chaque cycle, comme la transmission d'une trame par un nœud est semblable à l'ordonnancement de tâches non préemptives dans un processeur. Par conséquent, si $U_h < 1$, toute rafale initiale sera finalement transmise et un temps d'inactivité se produira. D'où la condition d'arrêt. □

4.5 EXPÉRIMENTATIONS

Dans cette section, nous appliquons l'approche BDF sur deux configurations de réseau. Dans un premier, nous illustrons l'algorithme BDF, avec un calcul détaillé sur un exemple d'étude de cas simple. Par la suite, nous montrons l'avantage du BDF, par rapport à une approche naïve, pour le dimensionnement d'un réseau industriel.

4.5.1 *Étude de cas*

Nous considérons ici le réseau d'exemple, composé de 10 flux et 18 nœuds, avec des ports de sortie FIFO à 100 Mbps. Les données d'entrée nécessaires pour appliquer l'algorithme BDF sont données par :

— la topologie du réseau, décrite à la figure 4.3 ;
— les contrats de trafic du tableau 4.1 ;
— les bornes de délais pire cas sont calculées par la méthode *Forward end-to-end Analysis* [64] (FA).

Nous illustrons le principe de l'algorithme 2 par un calcul détaillé du retard accumulé dans le pire cas pour la file d'attente du nœud h_4.

Calcul détaillé

Le nœud h_4 est traversé par trois flux v_3, v_4 et v_5, arrivant via deux liens entrants. Les gigues dans le pire cas pour ces flux, lors de leur entrée dans h_4, peuvent être obtenues avec une méthode de calcul pire cas, telle que la méthode FA [64] : $J_3^{h_4} = 0$µs, $J_4^{h_4} = 42$µs et $J_5^{h_4} = 32$µs. Le nombre de trames arrivées et parties, en suivant l'algorithme 2, est tracé sur la figure 4.8. La courbe des trames arrivées (resp. des trames parties) est une enveloppe borne supérieure (resp. borne inférieure) calculée, valable pour toute politique de service non préemptive et conservative, parmi lesquels FIFO.

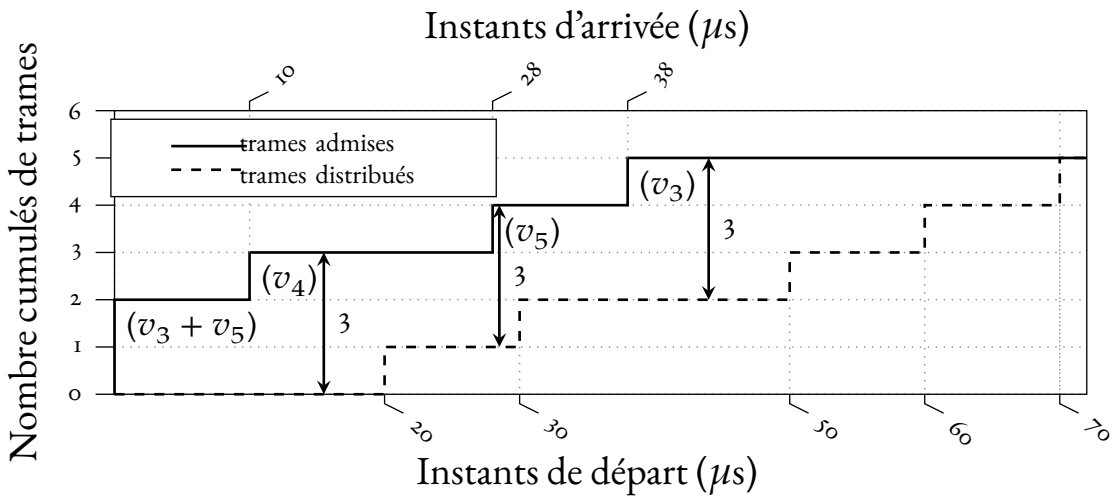

FIGURE 4.8 – Courbes d'arrivée et de service du nœud h_4.

Dans cet exemple, h_4 n'est pas un nœud produisant des données ($\Gamma_0^{h_4} = \emptyset$). Les trames arrivent via les liens entrants $IP_1^{h_4}$ et $IP_2^{h_4}$, respectivement de h_1 et h_4. Les ensembles de flux entrants par des liens entrant sont : $\Gamma_1^{h_4} = v_4, v_5$ et $\Gamma_2^{h_4} = v_3$, avec : $C_5^{h_4} = 20\mu s$ et $C_3^{h_4} = C_4^{h_4} = 10\mu s$.

En considérant $IP_1^{h_4}$, au temps 0, une seule trame de chaque flux v_4 et v_5 est éligible pour entrer dans h_4, selon leurs fonctions *rbf* respectives. En tenant compte de l'effet de sérialisation (théorème 4.4.2), une seule des deux peut arriver au temps 0. Et comme, $t = 0$ est un temps d'inactivité, SPT* choisit la trame de taille maximale (du flux v_5). Simultanément, une seule trame du flux v_3 peut arriver via $IP_2^{h_4}$. Au fur et à mesure que les trames arrivent, dès $t = 0$, une période d'activité commence, et h_4 transmet les trames via son port de sortie, en se basant sur LPT. Parmi toutes les trames qui auraient pu arriver, selon le RBF de tous les flux de Γ_{h_4}, la plus grande est choisie en premier (théorème 4.4.1), qui est, dans ce cas, une trame de v_5, avec un temps de transmission $C_5^{h_4} = 20\mu s$. Le prochain instant à considérer coïncide avec la prochaine marche de l'une des courbes. À $t = 10\mu s$, la deuxième trame entrant via $IP_1^{h_4}$ arrive. Ensuite, l'algorithme avance à $t = 20\mu s$, à la fin de la transmission de la première trame, où la prochaine plus grande trame disponible est sélectionnée par LPT. Le même processus se poursuit, jusqu'à un point où un temps d'inactivité se produit : ici, les deux courbes convergent à $t = 70\mu s$. Le retard accumulé dans le pire cas est la distance verticale maximale entre les courbes, c'est-à-dire 3 trames.

Résultats et comparaisons

Les résultats globaux du calcul pour cet exemple de configuration sont résumés dans le tableau 4.2. Le dimensionnement pire cas obtenu avec BDF est comparé à celle de l'approche plus naïve (cf. Introduction), qui consiste à diviser le retard accumulé maximal en bits par la taille minimale des trames parmi tous les flux entrants dans le nœud. Le retard accumulé maximal en bits peut être obtenu avec toute approche d'analyse de délai de bout en bout dans le pire cas, en multipliant le temps d'attente maximal dans un nœud par son débit de sortie. Ici, la méthode FA a été utilisée

autant pour calculer le retard accumulé maximal en bits avec l'approche naïve que pour fournir la gigue maximale en entrée pour le calcul du BDF.

Par exemple, dans le nœud h_1, la taille minimale des trames est de 1 kb (pour les flux v_3 et v_4), et le retard accumulé maximal calculé par FA est de 5,2 kb. Par conséquent, la fille d'attente au niveau du nœud peut théoriquement contenir simultanément jusqu'à 6 trames, tandis que BDF limite ce maximum à 3 trames.

La première partie de la table 4.2 (fournie par la méthode FA) est nécessaire pour calculer la borne supérieure naïve et pour calculer l'utilisation de la mémoire dans le pire cas. Les deux lignes suivantes donnent le dimensionnement maximal des files d'attente en trames pour l'approche naïve et BDF. Le dimensionnement est toujours plus petit avec BDF. Le bénéfice le plus important est observé lorsque des trames de tailles très différentes se produisent, comme dans les nœuds h_1, h_{15} et h_{18}. Par exemple, la taille des trames locales pour le nœud h_{15} varie de 1 à 6,4 kb.

Dans les deux dernières lignes du tableau 4.2, nous comparons l'utilisation de la mémoire induite par les deux approches et, en particulier, le surdimensionnement des files d'attente en nombre de bits. La quantité totale de mémoire allouée est calculée en multipliant le nombre d'emplacements alloués selon chaque approche par la taille de la plus grande trame entrant dans le nœud. L'utilisation de la mémoire est ensuite calculée par le rapport entre le dimensionnement maximal en bits (ici, obtenu par FA) et la mémoire totale allouée (pour chaque méthode). Par exemple, dans h_1, l'allocation de mémoire est de 3 × 2,2 kb = 6,6 kb. Mais, selon les résultats obtenus avec FA, un maximum de 5,2 kb est réellement utilisé dans h_1, ce qui correspond à une utilisation de la mémoire de 79 %. Cette utilisation tombe à 39 % lorsque la mémoire est allouée selon l'approche naïve (6 × 2,2 kb = 13,2 kb).

Les résultats pour les nœuds avec un seul flux entrant (par exemple, le nœud h_7) ou avec exactement deux flux ayant des caractéristiques identiques (par exemple, dans le nœud h_8) ont été ignorés dans la table, car le retard accumulé maximal en trames peut être obtenu par le retard accumulé maximal en bits. Pour de tels nœuds, le résultat obtenu par les deux méthodes est optimal et ne peut pas être amélioré. En moyenne, pour les colonnes présentes dans le tableau, où la stratégie d'allocation est importante, la mémoire moyenne allouée par BDF en nombre d'emplacements diminue d'un tiers par rapport à la méthode naïve. L'utilisation moyenne de la mémoire passe de 44 % pour l'approche naïve à 58 % pour BDF. En limitant le surdimensionnement des ressources allouées, BDF permet une meilleure efficacité et des coûts de conception réduits pour les commutateurs embarqués.

4.5.2 *Configuration industrielle*

Nous utilisons maintenant BDF pour le dimensionnement d'une configuration industrielle, établie à partir d'un réseau cœur AFDX d'un Airbus A380 [17]. Il est composé de huit commutateurs (SW) à 16 ports et de 96 ports d'entrée/sortie, ce qui fait 288 nœuds, ainsi que 983 flux. Nous alimentons le calcul du BDF avec les délais dans le pire cas calculé avec la méthode FA [17]. Pour cette configuration de grande taille, le temps de calcul d'une version simple cœur du BDF est de 117 ms sur un Intel Core i5-10500 à 3,1 GHz, et une empreinte mémoire de 27 Mo, avec Python comme langage de programmation.

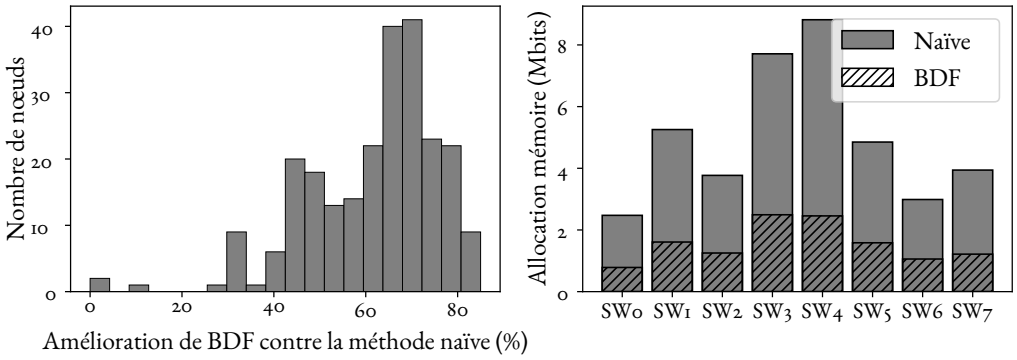

FIGURE 4.9 – Amélioration des performances du BDF par rapport à la méthode naïve dans un cas de test industriel.

Une présentation synthétique des résultats est donnée figure 4.9. Sur le côté gauche, l'amélioration relative de BDF par rapport à la méthode naïve est représentée. En moyenne, BDF offre une réduction de 76,3 % de la borne supérieure garantie pour l'allocation de mémoire en nombre de trames. BDF donne une borne plus serrée pour tous les flux, sauf pour deux, où les deux méthodes coïncident. Cela se produit sur les nœuds peu chargés, où toutes les trames ont la même taille.

Sur le côté droit de la figure 4.9, l'allocation totale de mémoire pour les 16 ports de sortie de chaque commutateur est représentée. En moyenne, lorsque BDF est utilisé pour le dimensionnement des files d'attente, l'allocation de mémoire est divisée par 3,16, par rapport à la méthode naïve. Sur le commutateur le plus chargé, le bénéfice est encore plus important, avec une borne divisée par 3,6, ce qui permet d'économiser 6,36 Mo de mémoire. La réduction de l'empreinte mémoire peut entraîner des améliorations de performance. Par exemple, toutes les trames peuvent être stockées dans le cache du CPU du commutateur et non dans la mémoire principale.

4.6 CONCLUSION

Dans ce chapitre, nous avons proposé une nouvelle approche dédiée au dimensionnement des buffers en nombre de trames dans le contexte de réseaux commuté temps-réel, où la certification est obligatoire et l'allocation dynamique de la mémoire n'est pas autorisée. La méthode *Buffer Dimentionning per Frame* (BDF) est applicable à tout réseau de paquets utilisant une politique de service conservative. Le calcul du pire cas ne nécessite que la connaissance de la topologie du réseau, des contrats de trafic et des délais de transmission de bout en bout pire cas et meilleur cas. Ces derniers peuvent être fournis par toute méthode d'évaluation de délais pire cas, telle que le Network Calculus ou la méthode FA.

La méthode BDF permet de s'affranchir de certaines des contraintes habituelles que l'on rencontre dans d'autres approches, telles que l'hypothèse de taille de trame constante ou l'hypothèse de trame la plus petite pour passer des bits aux trames dans le cas de la méthode naïve. Nous l'avons appliqué à un cas d'exemple, avec un calcul détaillé pour un nœud, ainsi sur une configuration industrielle réelle, où il a montré un bénéfice significatif par rapport l'approche naïve sur un cas

d'exemple industriel. Une preuve formelle concernant une amélioration de performances de BDF par rapport à la méthode naïve pourrait constituer un axe de perspectives à ces travaux.

Cet article est en cours de soumission depuis décembre 2022 et est actuellement en attente de révision par le journal *Real-Time Systems*.

	Nœud	h_1	h_3	h_4	h_6	h_9	h_{12}	h_{13}	h_{14}	h_{15}	h_{17}	h_{18}
Données produites par FA	Taille maximale de trame (kb)	2.2	2.2	2	2	2	2.2	1	2.2	6.4	2.2	6.4
	arrièré maximal (kb)	5.2	5.4	3	3	2	10.2	2	4.2	9.6	9.6	7.4
Emplacements réservés	Naïve	6	6	3	3	2	11	2	5	10	10	8
	BDF	3	5	3	2	2	8	2	4	4	9	3
Utilisation mémoire	Naïve (%)	39	41	50	50	50	42	100	38	15	44	14
	BDF (%)	79	49	50	75	50	58	100	48	38	48	39

TABLE 4.2 – Comparaison entre BDF et la méthode naïve sur un cas d'exemple

5

PROBLÈMES D'ORDONANÇABILITÉ DANS LES RÉSEAUX DIRIGÉS PAR LE TEMPS

Les contributions apportées dans ce chapitre sont basées sur l'analyse de la transmission de trames. Ces travaux sont basés sur l'analyse entre la transmission d'une trame par un nœud avec l'ordonnancement non préemptif de tâches sur un système monoprocesseur. Elles sont le fruit de collaborations avec un autre doctorant de l'équipe temps réel du LIAS [68].

Dans les réseaux temps réel, les trames émises par des flux périodiques subissent de grandes variations de délais (ref. 3.1) en fonction du trafic croisé en chemin. Un problème ouvert consiste à déterminer la possibilité d'attribuer des *offsets* aux flux, sur leurs nœuds sources, afin de minimiser les délais de bout en bout au maximum. Certaines normes existantes telles que le *TTEthernet* et TSN permettent de configurer des *offsets*. Ce type de système est désigné sous le nom de système à *offset free* dans la littérature. Néanmoins, ces normes ne donnent pas d'indications sur comment les trouver efficacement.

En ordonnancement temps réel *offset free*, l'un des problèmes est celui de la cyclicité. La cyclicité dans un ordonnancement réseau est le plus petit schéma de transmission de trames qui se répète. Des comportements cycliques ont été identifiés et l'article [72] indique que la durée maximale de simulation pour un système de tâches avec offsets est $O_{max} + 2H$, où H représente l'hyperpériode, donnée par le plus petit commun multiple (PPCM) des périodes des tâches, et O_{max} le plus grand offset. L'ordonnancement sur $[O_{max} + H, O_{max} + 2H)$ est garanti de se répéter infiniment avec un algorithme d'ordonnancement déterministe. Cependant dans plusieurs articles, nous avons remarqué que chaque méthode *offset-free* appliquée à l'ordonnancement de réseaux temps réel ne considère que l'intervalle d'étude $[0, H)$ pour valider les systèmes étudiés.

À notre connaissance, il n'existe aucune preuve formelle démontrant que l'étude peut être ainsi limitée à cet intervalle d'étude. Nous avons donc décidé de vérifier la validité et l'applicabilité de cette hypothèse. Les méthodologies adoptées et nos conclusions sont développées dans ce chapitre.

Nous démontrons formellement que forcer un point d'inactivité (un point où il n'y a pas de travail en attente) à se produire au temps H contraint l'intervalle d'étude analysé des tâches périodiques (ou des flux périodiques) avec offsets à $[0, H)$. De plus, nous prouvons une condition suffisante pour obtenir ce point d'inactivité et donc d'assurer deux contraintes :

1. *absence de congestion* (*contention-freedom*) : vérifier qu'à tout moment de l'ordonnancement, au plus une seule trame est prête à être envoyée ;
2. *contrainte sur trames* (*frame constraint*) : les offsets ne doivent pas être supérieurs à la période moins la durée du message.

La contrainte 1. est difficile à vérifier en raison de l'aspect combinatoire sous-jacent, voire irréalisable pour certains systèmes. Ainsi, nous proposons une heuristique capable de réduire cette complexité, même si la contrainte 1 ne peut pas être satisfaite. Cette heuristique ne respecte pas

systématiquement la contrainte *d'absence de congestion*, mais n'est pas affectée par le problème de cyclicité de l'ordonnancement remarqué dans d'autres méthodes.

Ces deux contraintes sont bien sûr très restrictives puisqu'à tout instant, au plus une trame est dirigée par le port de sortie étudié. Ces hypothèses sont toutefois le prix à payer pour déterminer correctement les *offsets* annulant la gigue des trames.

Les sections suivantes sont organisées comme suit : la section 5.1 fournit une revue complète des travaux publiés axés sur la génération d'un ordonnancement réalisable grâce à l'identification d'offsets appropriés pour les réseaux temps réel. La section 5.2 utilise des exemples simples pour expliquer pourquoi l'hypothèse de $[0, H)$ comme cycle d'un ordonnancement est incorrecte lorsque l'on remplit seulement une seule des deux conditions (*contrainte de non-congestion* ou *contrainte sur trames*). La section 5.3 explore l'essence du problème de l'identification du début de la partie cyclique d'un ordonnancement avec offsets, en montrant d'abord le résultat principal de ce chapitre – *l'absence de congestion* ET *la contrainte sur trames* est une condition suffisante pour garantir la cyclicité sur l'intervalle $[0, H)$ –, puis en explorant la façon dont le début de la partie cyclique d'un ordonnancement peut être calculé dans un contexte général. Dans la section 5.4, nous proposons une nouvelle méthode heuristique pour trouver un ordonnancement réalisable pour un réseau donné et un ensemble de flux périodiques même lorsque le cycle n'est pas la première hyperpériode. Enfin, la section 5.5 offre une description des performances de l'heuristique en comparaison avec la première méthode optimale SMT [100].

5.1 NOTATIONS ET ÉTAT DE L'ART

5.1.1 *Approches d'ordonnancement par offsets et notations*

Dans un réseau temps réel, la transmission d'un message par un nœud (et en particulier son port de sortie) est analogue à l'exécution d'une tâche en attente sur un processeur. Par conséquent, bien que la littérature sur le problème de la cyclicité soit vaste, nous nous limitons à l'ordonnancement monoprocesseur non préemptif. Le problème de la recherche d'un cycle dans un ordonnancement de tâches périodiques avec offsets a été initialement abordé dans [71, 72], où deux résultats fondamentaux concernant les systèmes de tâches avec offsets sur des systèmes monoprocesseurs sont présentés.

Premièrement, il a été démontré qu'un ordonnancement infini obtenu par une affectation de priorités (notons que les preuves sont toujours valables pour tout ordonnancement conservatif déterministe et sans mémoire qu'il soit préemptif ou non) pour un système de tâches périodiques avec offsets et des échéances implicites n'est faisable que s'il est valide sur l'intervalle $[0, O_{max} + 2H)$. L'ordonnancement donné sur l'intervalle de temps $[O_{max} + H, O_{max} + 2H)$ se répète infiniment. Nous appellerons cet intervalle qui se répète la *partie cyclique* d'un ordonnancement. La partie acyclique de l'ordonnancement est l'intervalle de temps entre l'initialisation du système et le début de la partie cyclique. Ce résultat a été étendu à des échéances arbitraires dans [47].

Deuxièmement, dans [71, 72], le problème pour déterminer la faisabilité d'un ordonnancement d'un système de tâches asynchrones sur un seul processeur est présenté comme étant coNP-Complet au sens fort. En d'autres termes, la simulation de l'ordonnancement, fonctionnant sur au

moins une hyperpériode, ne peut pas être utilisée pour prouver l'ordonnançabilité du système. En effet, l'hyperpériode, à moins que les périodes ne soient harmoniques, croit de façon exponentielle, car PPCM$\{1, 2, 3, ...n + 1\} \geq 2^n$ ([27]). Ce résultat conduit à un coût calculatoire très élevé.

Ensuite, un intervalle de faisabilité spécifique pour les algorithmes d'ordonnancement à priorité fixe préemptive est présenté dans [46]. L'intervalle proposé, en considérant les priorités, inclut le cycle si, et seulement si, le système est ordonnançable par l'affectation de priorité choisie.

Une manière exacte de calculer les parties acycliques et cycliques d'un ordonnancement, qui ne dépend que des offsets, des périodes et des durées, est présentée dans [28]. Chaque cycle de la partie cyclique a une longueur H et contient exactement le même nombre d'unités de temps où le système est inactif (temps creux). De par les offsets, la partie acyclique contient au moins un temps d'inactivité supplémentaire qui n'existe pas dans la partie cyclique. Il a été démontré dans [28] que "la partie cyclique" d'un ordonnancement valide produit par tout ordonnanceur conservatif et sans mémoire pour des tâches périodiques avec offsets est identifiée comme étant le premier intervalle de taille H suivant le dernier temps d'inactivité supplémentaire. Ce temps d'inactivité supplémentaire peut apparaître en début d'ordonnancement et peut être analogue aux perturbations qu'il peut y avoir lorsqu'un système informatique est sur sa phase de démarrage. Cela pourrait être conceptualisé comme le premier intervalle de longueur H sans aucun temps d'inactivité supplémentaire. Le dernier temps d'inactivité supplémentaire possible apparaît au plus tard à $O_{max} + H$. La détermination du dernier temps d'inactivité supplémentaire est un problème fortement combinatoire [28].

Notations

Un réseau commuté est représenté comme un ensemble de nœuds $h \in \Gamma$ et un ensemble de flux multicast v_i. Chaque nœud h, représente un port de sortie soit d'une *End-Station* (ES) soit d'un commutateur (SW). Chaque port de sortie d'un nœud h fonctionne à un débit de transmission constant de r^h.

Un flux v_i est défini par :

$$v_i = \{C_i, T_i, \{O_{i,h}\}, \{P_{i,m}\}\} \tag{5.1}$$

où C_i est la durée de transmission de chaque trame périodique de v_i (taille de trame divisée par r^h), T_i est la période du flux, $P_{i,m}$ est un ensemble de chemins pour v_i. Un chemin est un ensemble ordonné de nœuds. $O_{i,h}$ constitue l'instant de transmission par rapport à la période pour le flux v_i dans chaque nœud h de $P_{i,m}$. Par simplicité, nous considérons que pour chaque flux v_i, une seule trame est envoyée à chaque période T_i. Pour un flux donné v_i, chaque chemin $P_{i,m}$ (où m est le nœud de destination) commence par le même premier nœud, appelé nœud source, mais a un nœud de destination différent. Les chemins d'un même flux ne peuvent pas former de boucle (une trame ne repassera jamais dans le même nœud). La transmission d'une trame ne peut pas être interrompue. L'ordonnancement correspondant est donc non préemptif.

5.1.2 *Les travaux fondamentaux du domaine de l'ordonnancement offset-free réseau (Steiner, et al. [100])*

Le premier travail sur les calculs d'offsets dans les réseaux temps réel a été publié par W.Steiner [100]. La méthode de calcul proposé repose sur l'intervalle d'étude $[0, H]$ pour valider l'ordonnancement avec les offsets calculés par leur méthode. Ce travail a été réutilisé par de nombreux autres travaux dans le même cadre d'hypothèses [32, 33, 41, 95, 96]. Comme signalé dans l'introduction de ce chapitre, l'hypothèse d'intervalle d'étude $[0, H]$ pour valider une solution est incorrecte dans ce cadre d'hypothèses. Les auteurs qui ont réutilisé ce résultat ont par voie de conséquence définit des résultats erronés. Afin d'analyser et corriger ce problème, nous introduisons les travaux de W.Steiner[100].

W.Steiner utilise une approche *Satisfiability Modulo Theories*(SMT) [11] pour trouver des *offsets* en garantissant qu'aucune trame ne sera en congestion avec une autre. Les problèmes SMT sont des problèmes de décisions relevant de la logique du premier-ordre. Ils peuvent être vus comme une généralisation des problèmes SAT avec l'utilisation de nombres entiers, rationnels ou encore de chaînes de caractères. Dans le cas de l'article [100], un ensemble de propositions logiques sont données avec pour objectif de contraindre le domaine de recherche de solutions possible pour les *offsets*. Nous parlons dans ce cas-là de contraintes. Cet ensemble de contraintes est ensuite donné à un solveur comme Z3 [79] qui permet de déterminer, dans notre cas les *offsets*. Dans notre évaluation expérimentale, nous utiliserons le solveur Z3.

Dans l'article [100], les contraintes suivantes sont définies :

- contraintes de non-congestion (*contention-free constraint*) : cette classe de contraintes vérifie que les offsets produits garantissent que deux trames concurrentes ne soient pas disponibles simultanément;
- contraintes relatives aux sauts de commutateurs dans le chemin : cette classe de contraintes permet de s'assurer qu'un offset donné pour une trame dans un commutateur est plus petit que les offsets donnés aux commutateurs suivants pour la même trame;
- contraintes de mémoire (*Bounded Switch Memory Constraints*) : cette classe de contraintes vérifie que le nombre de trames pouvant être mises en attente dans un commutateur n'excède pas sa capacité de stockage;
- contraintes optionnelles de transmission simultanée (*Simultaneous Relay Constraints*) : cette classe de contraintes permet de donner les mêmes offsets à une trame qui se trouve dans un point de multiplexage (une trame qui est reconduite sur deux liens différents dans le même commutateur);
- contraintes de délai de transmission de bout en bout (*End-to-end transmission constraints*) : cette classe de contraintes vérifie qu'avec les offsets donnés, les trames arrivent dans un temps contraint;
- contraintes d'ordre applicatif (*Application-Level Constraints*) : certaines trames doivent être transmises avant d'autres afin de maintenir un ordre ou une cohérence dans les données;
- un ensemble de contraintes liées au fonctionnement des protocoles réseaux utilisés (*Other Constraints*);

La contrainte de non-congestion est celle qui sera la plus importante dans la suite de ce manuscrit est détaillée ici. Par l'activation périodique des flux de trames, le nombre de scénarios différents d'activation de trames de deux flux v_a et v_b traversant un nœud h est $PPCM(T_A, T_b)$. Dans l'intervalle $[0, PPCM(T_a, T_b)]$, le nombre maximal de réveils de flux v_a (respectivement v_b) est défini par $PPCM(T_a, T_b)/T_a$ (respectivement $PPCM(T_a, T_b)/T_b$). La contrainte de non-congestion vérifie à tout moment qu'au plus une trame est prête à être envoyée dans l'ordonnancement. Ainsi, la contrainte de non-congestion va alors vérifier que toute trame de v_a précède l'envoi d'une trame de v_b ou bien l'inverse, la trame de v_b précède l'envoi d'une trame de v_a. Dans la formulation SMT [33] [1], elle est définie de la manière suivante :

$$\begin{aligned}
&\forall h \in S, \forall v_a, v_b \in \Gamma^h, v_a \neq v_b \\
&\forall \alpha \in [0, \text{PPCM}\{T_a, T_b\}/T_a], \forall \beta \in [0, \text{PPCM}\{T_a, T_b\}/T_b] : \\
&(O_{a,h} + \alpha \times T_a \geq O_{b,h} + \beta \times T_b + C_b) \vee \\
&(O_{b,h} + \beta \times T_b \geq O_{a,h} + \alpha \times T_a + C_a)
\end{aligned} \tag{5.2}$$

où Γ^h est l'ensemble des flux traversant le nœud h.

Ultérieurement à la publication [100], une nouvelle contrainte a été ajoutée, la *contrainte sur trames* [33]. Elle impose qu'aucun *offset* ne doit être plus petit que 0 ou conduire la trame à terminer sa transmission au-delà de sa période. Dans la formulation SMT, elle est définie de la manière suivante :

$$\begin{aligned}
&\forall h \in S, \forall v_a \in \Gamma^h \\
&\forall \alpha \in [0, \text{H}/T_a] : \\
&(O_{a,h} + \alpha \times T_a \leq (\alpha + 1) \times T_a - C_a) \wedge \\
&(O_{a,h} + \alpha \times T_a \geq \alpha \times T_a)
\end{aligned} \tag{5.3}$$

Nous monterons dans la suite de ce chapitre pourquoi considérer un seul PPCM pour la contrainte de non-congestion n'est pas suffisant et peut conduire à penser qu'une solution produite garantie dans tous les cas l'absence de congestion. Dans un second temps, nous expliquerons pourquoi l'ajout de la *contrainte sur trames* permet de contraindre l'intervalle d'analyse sur un PPCM.

5.2 PROBLÈMES AVEC LES HYPERPÉRIODES

5.2.1 *Cyclicité et absence de congestion*

Dans les réseaux asynchrones, des trames provenant de plusieurs flux peuvent être en concurrence dans le même port de sortie de commutateur. Cette concurrence peut entraîner des variations de l'occupation des buffers et une grande variabilité pour les délais de transmission. Ces variations peuvent être atténuées par des stratégies d'ordonnancement au sein de réseaux synchrones.

1. Plusieurs écritures de cette contrainte existe. Nous choisissons ainsi celle-ci pour sa simplicité.

TABLE 5.1 – Paramètres pour les flux constituant les différents exemples de la figure 5.1 et 5.2

	Flux	Période	Temps de transmission	Offset
Cas d'exemple 1	1	12	8	0
	2	18	5	8
Cas d'exemple 2	1	12	8	5
	2	18	5	0
Cas d'exemple 3	1	7	2	0
	2	7	4	4

Au meilleur de nos connaissances, chaque méthode proposée pour ordonnancer des flux au sein de réseaux commutés implique d'affecter un offset à chaque flux dans chaque port de sortie. Ainsi, l'analyse porte pour chaque flux de l'ensemble de ses trames sur 1 hyperpériode. Cependant, en théorie de l'ordonnancement, cette analyse se fait en une hyperpériode de plus. Nous illustrons avec des exemples simples que limiter cette analyse sur une hyperpériode est insuffisant.

Nous introduisons un premier exemple comportant deux flux qui traversent un nœud, détaillé comme Cas 1 dans le Tableau 5.1. Dans ce cas, il est impossible pour les deux flux d'éviter totalement une congestion dans le port de sortie. Un chevauchement est inévitable car $C_1 + C_2 = 13 > \text{PGCD}\{T_1, T_2\} = 6$ (la méthode est expliquée dans le Lemme 5.4.2). La simulation de cet exemple est représentée dans la Figure 5.1 (Cas 1).

Dans cet exemple, on peut remarquer que la première hyperpériode, $[0, \text{PPCM}\{12, 18\}] = 36$, n'est pas un cycle qui se répète. L'ordonnancement nécessite une construction après le dernier temps d'inactivité supplémentaire, survenant au temps $[21, 22)$, débuté cycliquement. Comme l'utilisation du processeur du système est $U = \sum_i \frac{C_i}{T_i} = \frac{34}{36}$, dans un cycle de longueur égale à l'hyperpériode, nous devrions exactement trouver deux temps d'inactivité : $H(1 - U) = 2$.

Si l'on considère la première hyperpériode $[0, 36)$ (voir Figure 5.1 (Cas 1)), il n'y a pas deux mais trois unités de temps d'inactivité, formant un intervalle d'inactivité sur $[21, 24)$. Le premier de ces trois temps d'inactivité est considéré comme un temps d'inactivité supplémentaire. Ainsi, l'intervalle de taille $H = 36$ définie en t=22 ($[22, 58)$), est le premier intervalle de longueur de l'hyperpériode H avec précisément deux temps d'inactivité. En conséquence, pour étudier l'ensemble du comportement de l'ordonnancement, nous devons considérer l'intervalle$[0, 58]$ avec cinq trames pour le flux 1 et trois trames pour le flux 2. Tandis que la partie cyclique ne contient que trois et deux trames respectivement.

Une autre classe de problèmes auxquels sont confrontés les méthodes d'affectation d'offsets est qu'elles considèrent toutes les trames, alors que le nombre de trames à considérer dépend des offsets. Par conséquent, en général, et avec les méthodes SMT, la solution et les variables d'entrée s'influencent mutuellement. Pour illustrer cela, si l'on considère les mêmes flux périodiques que dans le Cas 1, mais en changeant les offsets en $O_1 = 5$ et $O_2 = 0$ (Cas 2), l'ordonnancement n'atteint pas son cycle au même instant que précédemment. Un nombre différent de trames pour chaque flux doit être considéré. Comme le montre la Figure 5.2 (Cas 2), nous voyons que le dernier temps d'inactivité supplémentaire se produit au temps $[14, 15)$ et l'ordonnancement se répète sur l'intervalle $[15, 51)$. Par conséquent, si une méthode nécessite de considérer chaque trame

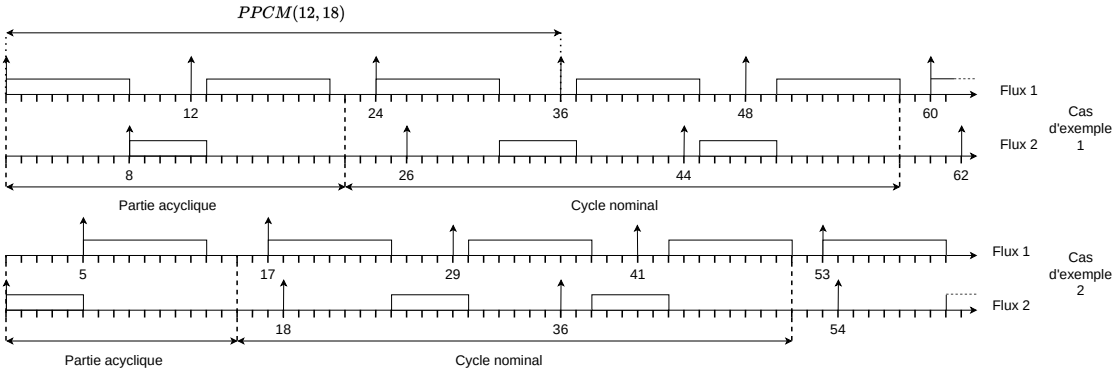

FIGURE 5.1 – Partie cyclique et acyclique sur les simulations des cas d'exemple 1 et 2 avec les paramètres de la table 5.1

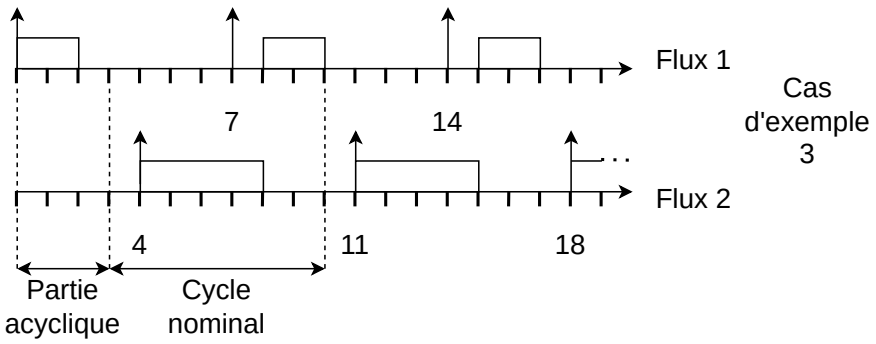

FIGURE 5.2 – Partie cyclique et acyclique sur les simulations des cas d'exemple 3 avec les paramètres de la table 5.1

dans un ordonnancement, elle doit considérer quatre trames pour le flux 1 (au lieu de cinq pour le Cas 1) et trois trames pour le flux 2. Par conséquent, tenir compte de la première hyperpériode de trames avec offsets n'est pas suffisant.

5.2.2 *Importance de la contrainte sur trames*

Nous montrons dans cette section que satisfaire la *contrainte de non-congestion* n'est pas suffisant pour garantir que l'ordonnancement $[0, H)$ se répète. Cette contrainte permet de s'assurer que les choix d'offsets ne conduisent pas à la concurrence entre deux trames pour une transmission simultanée. Nous utilisons la *contrainte de non-congestion* telle qu'elle a été proposée dans [33] (cf. 5.2). La *contrainte sur trames* est décrite par l'équation 5.3.

Avec cet ensemble de contraintes, le nombre de trames à considérer est limité par le nombre de trames que nous pouvons trouver dans la première hyperpériode de longueur PPCM$\{T_a, T_b\}$. Cette contrainte force une transmission séquentielle des trames dans une période d'activité : une trame ne peut être transmise que si aucune autre est en transmission. Cependant, si nous n'incluons pas la *contrainte sur trames*, l'ensemble de contraintes donné dans l'équation 5.2 n'est pas suffisant pour considérer toutes les trames qui apparaissent dans l'ordonnancement proposé.

Ce phénomène est illustré dans un exemple simple présenté figure 5.2, avec deux flux traversant un nœud (leurs caractéristiques sont décrites comme Cas 3 dans le Tableau 5.1). Dans ce cas, l'ordonnancement commence à se comporter cycliquement après le dernier intervalle d'inactivité supplémentaire qui se produit pendant $[2, 3)$. L'ordonnancement se comporte cycliquement sur $[3, 10)$ et le nombre de trames à considérer comme représentant l'ensemble de l'ordonnancement (parties acyclique et cyclique) inclut deux trames du flux 1 et une trame du flux 2. Cela implique que, si nous nous limitons à une trame par flux, conformément aux contraintes de l'équation 5.2, nous négligeons alors la seconde trame du flux 1 et le retard qu'elle subit par la première trame du flux 2. Nous pouvons conclure à tort, que dans ce système il n'y a aucune congestion. Cependant, cet exemple ne respecte pas la *contrainte sur trames*. Cette dernière imposant que $(O_{a,h} + \alpha \times T_a \leq \alpha \times T_b + C_a) \wedge (O_{a,h} + \alpha \times T_b \geq 0$, empêcherait la première trame du flux 2 de se terminer après $H = 7$. Cet exemple montre que lorsque la *contrainte de trame* ajoutée dans [13] et [33] n'est pas respectée, nous pouvons trouver des ordonnancements qui ne se comportent pas de façon cyclique dès le départ.

Un autre risque serait la génération d'un ordonnancement à partir de $[0, H)$ qui inclut des temps d'inactivité supplémentaires. En raison des temps d'inactivité supplémentaires présents dans l'ordonnancement, il n'y aurait pas suffisamment de temps dédié aux trames pour pouvoir les ordonnancer dans chaque hyperpériode de la partie cyclique. Les trames s'accumuleraient au niveau du port de sortie jusqu'à ce qu'après un certain nombre d'hyperpériodes, les files d'attente atteignent leur pleine capacité, ce qui entraînerait la perte de trames.

En conclusion de cette section, si les *contraintes sur trames* sont satisfaites, mais que certaines trames ont le potentiel d'interférer les unes avec les autres, le système peut ne pas se comporter cycliquement sur $[0, H)$. De plus, si nous ne considérons que les trames émises dans la première hyperpériode sans respecter la *contrainte sur trames*, alors nous pouvons être amenés à conclure à tort qu'aucune congestion ne se produit.

Dans la section suivante, nous démontrons que la conjonction de *contrainte de non-congestion* et de *contrainte sur trames* constitue une condition suffisante pour garantir que $[0, H)$ est la partie cyclique d'un ordonnancement. Pour cela, nous allons approfondir le problème de la cyclicité des ordonnancements de flux avec offsets dans la section suivante.

5.3 CYCLICITÉ ET ACYCLICITÉ DANS LES ORDONNANCEMENTS AVEC OFFSETS

La cyclicité de l'ordonnancement a été initialement établie dans la littérature sur l'ordonnancement monoprocesseur. Nous adaptons ici les termes employés pour les rendre conformes à notre modèle. Nous nous mettons dans le cadre de cette section à l'échelle des commutateurs réseaux. Par conséquent, nous retirons toutes références aux nœuds h aux notations proposées dans cette section.

5.3.1 *Résultats fondamentaux sur la cyclicité*

Pour analyser l'ordonnancement réseau, nous devons établir une définition complète d'un ordonnanceur par rapport à un système de flux qui en a besoin. Ces définitions sont dérivées de [48]. À tout instant $t > 0$ au sein d'un ordonnancement, nous employons $C_i(t)$ comme le travail restant à effectuer pour terminer la transmission d'une trame en cours de transmission généré par un flux v_i. Cette fonction prend la valeur zéro lorsque la transmission du flux est complètement terminée. Le travail restant cumulé pour la transmission de messages est défini par $C(t) \stackrel{\text{def}}{=} \sum_{i=0}^{n} C_i(t)$.

Définition 5.3.1 (Horloge locale d'un flux périodique)**.** L'horloge locale d'un flux v_i, notée $\Theta_i(t)$, est une fonction périodique continue, réelle et non-négative modulo T_i, telle qu'elle varie entre 0 et $T_i - \epsilon$ ($\epsilon > 0 \wedge \epsilon \to 0^+$). Elle est définie sur l'intervalle ouvert $(O_i - T_i, +\infty)$, et non définie avant. Si $O_i < T_i$, alors $\Theta_i(0) = (O_i - T_i) \mod T_i = T_i - O_i$, de sorte que l'horloge locale atteigne zéro (c'est-à-dire T_i modulo T_i) à O_i. Si $O_i \geq T_i$, alors elle n'est pas définie à et avant $O_i - T_i$, et prend la valeur $\epsilon > 0$ à $O_i - T_i + \epsilon$, pour qu'elle soit réinitialisée à zéro à O_i.

On peut observer que d'après l'algèbre de base [66], l'ensemble des horloges locales est périodique avec une période H, commençant à partir de zéro si aucun offset n'est supérieur ou égal à la période, et commençant à partir de la valeur maximale de $O_i - T_i + \epsilon$ s'il existe des flux i tels que $O_i \geq T_i$. Au fur et à mesure que de nouveaux messages sont produits, la quantité de travail à traiter augmente. Nous désignons $r_i(t)$ comme la fonction de libération caractéristique de v_i, définie comme nulle partout, sauf lorsque son horloge locale est à zéro, c'est-à-dire lorsque t satisfait $\Theta_i(t) = 0$. Dans ce cas, $r_i(t) = C_i$. Le travail total libéré au temps t est $r(t) = \sum_{i=1}^{n} r_i(t)$. À tout instant t, la charge de travail individuelle à traiter pour un flux est $W_i(t) = C_i(t) + r_i(t)$, et cette charge de travail diminue de δ chaque fois que v_i distribue ses trames dans l'intervalle $[t, t + \delta)$. La somme des charges de travail individuelles est notée $W(t)$. Toutes ces fonctions sont positives.

Définition 5.3.2 (État d'un système)**.** L'état d'un système au temps t est déterminé par les valeurs des charges de travail individuelles, $W_i(t)$ des flux et des horloges locales $\Theta_i(t)$.

Définition 5.3.3 (Point d'inactivité, temps d'inactivité, intervalle d'inactivité)**.** Un **point d'inactivité** est un moment dans le temps pendant lequel le port de sortie d'un commutateur n'est pas occupé. Avec notre notation, un point d'inactivité t est caractérisé par $\forall i, C_i(t) = 0$, ce qui est défini de manière équivalente par $C(t) = 0$, et en raison de la nature non négative des fonctions, encore équivalent à $W(t) = r(t)$. Pendant un point d'inactivité, la charge de travail est uniquement déterminée par la génération de nouvelles trames, le cas échéant. Le point d'inactivité initial suivant une période d'activité signifie la fin de ladite période d'activité.

Un **intervalle d'inactivité** est initié par un point d'inactivité et désigne une période continue d'inactivité s'étendant de ce point d'inactivité jusqu'à ce que qu'une trame soit prête. Il est défini comme un intervalle de taille $\delta > 0$ qui suit un point d'inactivité à t, satisfaisant $\forall i, \forall \theta \in [t...t+\delta), r_i(\theta) = 0$.

Nous définissons le concept de **temps d'inactivité**, représentant le plus petit intervalle d'inactivité possible. Dans les cas où les paramètres des flux soient des entiers, un temps d'inactivité prend la forme d'un intervalle d'inactivité confiné à une unité de temps. Pour les paramètres rationnels, la durée minimale d'un intervalle d'inactivité, appelée temps d'inactivité, peut être exprimée comme $\frac{1}{\text{PPCM}_d}$, où PPCM_d désigne le plus petit commun multiple de tous les dénominateurs apparaissant dans les valeurs des offsets, des durées et des périodes des flux.

Le théorème de cyclicité général, tel qu'il a été présenté dans [28], qui est démontré pour des paramètres entiers, peut être généralisé aux nombres rationnels. Il peut s'exprimer de la façon suivante :

Théoréme 5.3.1 (CYCLICITÉ)**.** *[28] Lorsque les échéances relatives sont inférieures ou égales aux périodes, tout ordonnancement produit par un ordonnanceur conservatif, déterministe et sans mémoire se comporte cycliquement avec une période H après le dernier temps d'inactivité supplémentaire. Ce temps d'inactivité supplémentaire se produit dans l'intervalle de temps $[-1, O_{max} + H)$ où $O_{max} = \max_i\{O_i\}$ et $H = PPCM_i\{T_i\}$.*

Ce théorème peut être illustré dans l'exemple de la Figure 5.2. Avec $H = 7$ et une utilisation du processeur de $U = \frac{6}{7}$, un ordonnancement réalisable doit précisément comporter $H \cdot (1-U) = 1$ temps d'inactivité par cycle. Sans cela, l'ordonnancement entraînerait des retards croissants d'une hyperpériode à l'autre. L'intervalle $[0, 7)$ comporte deux occurrences de temps d'inactivité. Par conséquent, l'un d'entre eux est un temps d'inactivité supplémentaire.

Le premier intervalle, avec exactement un temps d'inactivité est $[3, 10)$, constitue la partie cyclique de l'ordonnancement. L'intervalle $[0, 3)$ représente le segment acyclique de l'ordonnancement, se terminant avec le dernier temps d'inactivité supplémentaire $[2, 3)$. Cet intervalle est qualifié d'acyclique, car il n'est exécuté qu'une seule fois au début de l'ordonnancement.

Il convient de noter que le cas particulier où le dernier temps d'inactivité supplémentaire se produit sur $[-1, 0)$, ce qui correspond au cas où le cycle commencerait au temps 0, conduit à ce que l'intervalle $[0, H)$ soit répétée. De plus, lorsque le dernier temps d'inactivité supplémentaire se termine à $O_{max} + H$, qui est, d'après le théorème 5.3.1, sa dernière position possible dans un ordonnancement, nous trouvons la borne supérieure de la cyclicité $O_{max} + 2H$, présentée dans [72].

Théoréme 5.3.2. *Les parties acyclique et cyclique d'un ordonnancement réalisable commencent et se terminent par un point d'inactivité.*

Démonstration. En se référant à la Définition 5.3.3, au temps 0, $W(0) = r(0)$, est un point d'inactivité. Maintenant, considérons deux scénarios :

1. l'absence de temps acyclique dans l'ordonnancement. Dans ce cas, il est défini sur l'intervalle $[-1, 0)$ où la charge de travail est nulle. Ainsi, il commence et se termine par un point d'inactivité, formant un temps d'inactivité du début à la fin. La partie cyclique de l'ordonnancement opère sur $[0, H)$: le temps 0 est un point d'inactivité, et conformément au Théorème 5.3.1, le même état est atteint à H, représentant également un point d'inactivité;
2. la présence d'un temps d'inactivité acyclique, avec t_x marquant sa fin. Le segment acyclique de l'ordonnancement est $[0, t_x)$, commençant à 0 comme un point d'inactivité et se terminant à t_x. Puisqu'il reste inactif et que l'ordonnanceur est conservateur, $C(t_x) = 0$, confirmant qu'il s'agit d'un point d'inactivité. La partie cyclique de l'ordonnancement opère sur $[t_x, t_x + H)$, les deux instances représentant le même état selon le Théorème 5.3.1. Notamment, puisque t_x est un point d'inactivité, $t_x + H$ l'est également.

□

Corolaire 5.3.1. *Il y a un nombre entier de trames dans la partie acyclique d'un ordonnancement et un nombre entier de trames dans la partie cyclique d'un ordonnancement.*

Démonstration. Par le théorème 5.3.2, la partie cyclique commence et se termine par un point d'inactivité. Aucun flux n'est en attente à la fin du cycle, et chaque flux qui est transmis dans cet intervalle est également terminé dans cet intervalle. □

Le Corollaire 5.3.1 présente des implications positives pour les méthodes traitant des flux avec *offsets*. En effet, il garantit que les trames ne se chevauchent pas entre les parties acycliques et cycliques d'un ordonnancement. Cependant, ce corollaire ne suffit pas pour les réseaux temps réel où les méthodes limitent l'intervalle à $[0, H)$. Ainsi, le théorème suivant et son corollaire dérivé sont nécessaires.

Théoréme 5.3.3. *Étant donné l'hypothèse qu'aucun offset n'est supérieur ou égal à la période, un ordonnancement se comporte cycliquement sur* $[0, H)$ *si, et seulement si, il y a un point d'inactivité à* H.

Démonstration. Ordonnancement cyclique sur $[0, H) \Rightarrow$ point d'inactivité à H : s'il y a un point d'inactivité à H, puisque $\forall i, O_i < T_i$, d'après la définition 5.3.1, les horloges locales se comportent cycliquement avec une période H à partir du temps 0. Il y a un point d'inactivité à 0. Par conséquent, l'état du système (voir définition 5.3.2), donné par les valeurs des charges de travail individuelles et des horloges locales, est le même au temps 0 et H. Puisque l'ordonnanceur est déterministe et sans mémoire, il prend la même décision et les mêmes états seront atteints. Ainsi, le système répète à partir du temps H ce qui s'est passé de 0 à H.

Ordonnancement cyclique sur $[0, H) \Leftarrow$ point d'inactivité à H : s'il n'y a pas de point d'inactivité à H, alors les états à 0 et H sont différents, et le système ne peut pas se comporter cycliquement sur $[0, H)$. □

Le théorème 5.3.3 joue un rôle central dans les réseaux temps réel. En effet, nous pouvons en déduire le résultat principal permettant à certaines méthodes capables de garantir l'*absence de congestion* d'une part, et qu'aucune trame ne déborde après H. Cela permet de générer des ordonnancements qui se comportent de façon cyclique sur $[0, H)$:

Corolaire 5.3.2. *Si un ordonnancement de flux avec offsets garantit :*

1. *la contrainte de non-congestion sur* $[0, H)$
2. *la contrainte sur trames, c'est-à-dire* $0 \leq O_i \leq T_i - C_i$

alors l'ordonnancement sur $[0, H)$ *est répété infiniment.*

Démonstration. La *contrainte de non-congestion* dans un ordonnancement conservateur signifie que la charge de travail à traiter au niveau de la sortie n'est donnée que par un seul flux, qui se termine au plus tard à la libération suivante. Cela signifie que pour tout t tel que $\exists i, r_i(t) > 0$, la prochaine fois qu'il y aura un j tel que $r_j(t + \delta) > 0$ est tel que $\delta \geq C_i$. Par conséquent, chaque période d'activité correspond à une seule requête, qui est exécutée dès qu'elle est libérée, et immédiatement terminée par un point d'inactivité. Par la contrainte sur trames, $O_i \leq T_i - C_i$ pour chaque flux, la dernière libération possible dans $[0, H)$ peut se produire à $H - C_i$, et sera transmise seule (la *contrainte de non-congestion*) et se terminera au plus tard à H, définissant ainsi un point d'inactivité. Comme 0 et H sont deux points d'inactivité, le corollaire est obtenu en appliquant le théorème 5.3.3. □

Le Corollaire 5.3.2 est d'une importance capitale pour la justesse de nombreux articles concernant l'affectation d'offsets dans les réseaux commutés. La *contrainte sur trames* a été initialement introduite en 2014 pour *TTEthernet* [13], puis pour TSN en 2016 [33]. Malgré cela, plusieurs articles n'utilisant pas la *contrainte sur trames* affirment que considérer les trames envoyées sur $[0, H)$ est suffisant. Cependant, nos résultats de la section 5.2.1 indiquent que garantir l'absence de congestion sans la *contrainte sur trames* ne garantit pas un cycle sur $[0, H)$. Par conséquent, il est important pour les personnes travaillant dans le domaine soient conscientes que certains articles peuvent par inadvertance négliger cet aspect crucial, ce qui peut affecter la justesse de leurs méthodes.

Dans la section suivante, nous présentons une méthode capable de gérer les temps d'inactivité acycliques et de fournir des ordonnancements même lorsque les deux contraintes du corollaire 5.3.2 ne sont pas respectées. À notre connaissance, il s'agit de la seule méthode le permettant, car ses calculs sont basés sur des flux au lieu de trames individuelles.

5.4 UNE HEURISTIQUE SUPPORTANT LA CONGESTION : GCD#

Dans l'état de l'art, nous avons souligné que chaque méthode abordant le problème de l'affectation d'offsets, qu'elle soit exacte ou heuristique, nécessite une solution qui garantit l'absence de congestion. Cependant, certains systèmes ne permettent pas de telles solutions (comme les réseaux TSN), posant des défis insolubles aux méthodes actuelles.

Dans cette section, nous introduisons une nouvelle approche heuristique adaptée aux réseaux temps réel, même lorsque la congestion de trame dans un port de sortie est inévitable. En opérant

au niveau du flux, cette stratégie se concentre sur l'interaction directe avec les flux au lieu des trames qui le composent, nous libérant ainsi des contraintes de l'intervalle temporel $[0, H)$.

Par conséquent, cette approche peut régir des systèmes avec des flux qui n'interdisent pas les solutions avec interférence. Là, où les méthodes basées sur SMT et les heuristiques qui nécessitent une absence de congestion ne parviennent pas à fournir une solution réalisable. Cette heuristique, adaptée de GCD+ [67], est notée GCD#. Nous commençons par un aperçu des principaux concepts, qui sont très proches de ceux de GCD+, en nous concentrant sur un seul nœud avant d'étendre la discussion au niveau du réseau. Tous les paramètres décrits dans les sections suivantes sont des entiers positifs. Bien que nous puissions considérer des nombres rationnels, nous choisissons d'exprimer chaque valeur sous forme d'entier par simplicité.

5.4.1 *Ordonnancement d'un seul nœud*

GCD+ [67] développé dans le Laboratoire d'Informatique d'Automatique et des Systèmes (LIAS) est conçu pour générer des offsets pour les systèmes monoprocesseurs *offset-free*. Nous adaptons ici, son expression à notre modèle.

La méthode accepte en entrée un ensemble de flux v_i où nous utiliserons simplement : T_i désignant la période des flux et C_i la durée de transmission. GCD+ s'efforce d'atteindre (mais ne garantit pas) l'*absence de congestion*, tout en visant également à ce que le délai d'attente de chaque flux soit nulle. Pour ce faire, l'algorithme tire parti de la semi-harmonicité des systèmes *offset-free*, c'est-à-dire les systèmes où toutes les périodes T_i sont un multiple d'un facteur commun. La semi-harmonicité est recherchée pour réduire l'espace d'exploration des solutions potentielles dans de nombreux travaux [14, 23, 84, 92, 108]. Cette approche est basée sur une technique utilisée à l'origine pour analyser deux flux dans un système monoprocesseur. Nous l'adaptons au domaine des réseaux temps-réel et utilisons l'opérateur modulo.

Lemme 5.4.1 ([87], Lemme 2). *Étant donné deux flux v_i et v_j, la distance temporelle minimale entre un instant de libération du flux i et l'instant de libération consécutive d'un flux v_j est égale à :*

$$\Delta_{i,j} = (O_j - O_i) \mod PGCD\{T_i, T_j\} \tag{5.4}$$

Dans un système composé uniquement de deux flux v_1 et v_2, il est possible de représenter la transmission des flux autour du cercle PGCD$\{T_1, T_2\}$, comme dans la Figure 5.3. Dans la figure, s'il n'y a pas de chevauchement dans l'exécution, alors les conditions de l'équation 5.5 sont satisfaites :

$$\begin{aligned} \Delta_{i,j} &\geq C_i \\ \Delta_{j,i} &\geq C_j \end{aligned} \tag{5.5}$$

Par conséquent, v_1 et v_2 n'interféreront jamais l'un avec l'autre, et le temps de réponse de chaque flux sera égal à son temps d'exécution. Ainsi, nous pouvons affirmer que si le système de

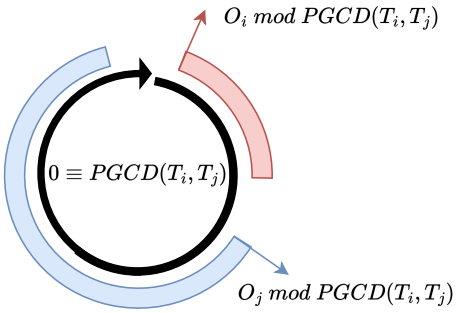

FIGURE 5.3 – Cercle PGCDT_i, T_j, avec une représentation de v_i et v_j

deux flux respecte l'ensemble d'inégalités précédent où $i = 1$ et $j = 2$, nous pouvons garantir l'absence de congestion sur le port de sortie.

Nous pouvons étendre cette technique pour évaluer plus de deux flux simultanément. Ceci est réalisé avec les concepts de sections et de cycles.

Lemme 5.4.2 (Non-interférence deux à deux)**.** *Si chaque paire de flux d'un système respecte l'équation 5.5, aucune congestion ne pourra jamais se produire dans l'exécution du système, et le temps de réponse de chaque flux est égal à sa durée de transmission.*

Il convient de noter que le Lemme 5.4.2 peut efficacement remplacer la *contrainte de non-congestion* telle qu'elle a été exprimée dans l'équation 5.2, car ils sont équivalents. Néanmoins, le premier peut être calculé en temps polynomial, tandis que le second nécessite un temps exponentiel.

Définition 5.4.1 (Le PGCD global, Ω)**.** Le PGCD global, également appelé Ω, est le plus grand commun diviseur de toutes les périodes de tous les flux (équation 5.6).

$$\Omega = \text{PGCD}\{T_i \forall i\} \tag{5.6}$$

Notre méthode fonctionne mieux sur des ensembles de flux dont les durées de transmissions ne sont pas supérieures à Ω. Sinon, elle fournit quand même des résultats, mais ils réduiront la congestion de manière heuristique, sans garantir l'absence de congestion. Dans les réseaux, un Ω grand est courant : par exemple, dans le réseau commuté *Ethernet* ARINC 664 partie 7 [4] utilisé dans l'avionique civile, chaque période autorisée pour les flux du réseau est une puissance de deux millisecondes, et tous les flux ont des périodes harmoniques.

Définition 5.4.2 (Cycle)**.** Le cycle O_C (pour $O_C \in \mathbb{N}$) est l'intervalle temporel tel que $O_C\Omega \leq t < (O_C + 1)\Omega$.

Définition 5.4.3 (Sous-période, T_{v_i})**.** La sous-période T_{v_i} d'un flux v_i est le rapport entre sa période T_i et Ω (équation 5.7).

$$T_{v_i} = \frac{T_i}{\Omega} \tag{5.7}$$

L'offset O_i attribué à un flux v_i peut être associé à un cycle spécifique O_{C_i} selon l'équation 5.8.

$$O_{C_i} = \left\lfloor \frac{O_i}{\Omega} \right\rfloor \tag{5.8}$$

D'après la définition de Ω, nous savons que $T_{v_i} \in N \forall i$. De plus, comme v_i a été affecté au cycle O_{C_i}, il est trivial de montrer que chaque instant de disponibilité de v_i se produira dans des cycles k tels que $k \equiv O_{C_i} \mod T_{v_i}$.

Lemme 5.4.3. *Pour toute paire de flux v_1 et v_2 donné, si PGCD$\{T_{v_1}, T_{v_2}\} = 1$, ils doivent être transmis dans le même cycle dans l'hyperpériode du système, et si PGCD$\{T_{v_1}, T_{v_2}\} > 1$, ils peuvent être organisés pour ne jamais être libérées dans le même cycle.*

Démonstration. Cette preuve est liée au théorème des restes chinois généralisé. Soit $T_i \in \mathbb{N} | i \in [1, n]$ et P le plus petit commun multiple de tous les T_i. Soit $a, O_1, O_2, ..., O_n$ des entiers quelconques. Si $O_i \equiv O_j \mod \text{PGCD}\{T_i, T_j\}$ pour $1 \leq i < j \leq n$, le théorème des restes chinois généralisé stipule qu'il existe exactement un entier k qui satisfait les conditions :

$$a \leq k < (a + P), k \equiv O_i \mod T_i \; \forall i \in [1, n]$$

Sinon, un tel entier n'existe pas. Si PGCD$\{T_{v_1}, T_{v_2}\} = 1$, quels que soient les premiers cycles (O_{C_1} et O_{C_2}) où elles sont disponibles, $O_{C_1} \equiv O_{C_2} \equiv 0 \mod 1$, et il doit exister un entier $k \equiv O_{C_1} \mod T_{v_1} \equiv O_{C_2} \mod T_{v_2}$. Sinon, si PGCD$\{T_{v_1}, T_{v_2}\} > 1$, O_1 et O_2 peuvent être choisis pour que $O_1 \not\equiv O_2 \mod \text{PGCD}\{T_1, T_2\}$, ce qui rend impossible l'existence d'un tel k. □

Avec le lemme 5.4.3, le concept de section peut être défini. Les flux qui ont des sous-périodes premières entre elles doivent être ordonnancés en considérant qu'elles finiront par être disponibles dans le même cycle, et donc une certaine partie du cycle doit être réservée à chaque flux si l'on recherche une affectation d'offset *sans congestion*. Cependant, les sous-périodes non premières entre elles permettront aux flux d'utiliser la même partie réservée du cycle, puisqu'elles peuvent être affectées à des cycles alternés. Ces parties réservées sont les sections auxquelles les flux sont affectés.

Définition 5.4.4 (Section, Ψ)**.** Une section Ψ est un sous-ensemble de flux tel que leurs sous-périodes sont soit toutes égales à un, soit elles doivent partager un facteur premier commun.

Ainsi, chaque section est liée à un nombre, le facteur commun dans la factorisation des sous-périodes des membres de la section (ou 1, au cas où les sous-périodes seraient unitaires), noté ici $p \in \mathbb{P} \cup \{1\}$, où $\mathbb{P}$ est l'ensemble de tous les nombres premiers. Afin de créer une partition

réservée dans chaque cycle pour chaque section, chaque section Ψ_p se voit attribuer un offset O_S par rapport au début du cycle et a une taille finie δ, donnée par l'équation 5.9.

$$\Psi_p.\delta = \max_{\forall i | v_i \in \Psi_p} ((O_i + C_i) \mod \Omega) - \min_{\forall i | v_i \in \Psi_p} (O_i \mod \Omega) \tag{5.9}$$

Ensuite, pour chaque cycle $k \in N$, l'intervalle $[k\Omega + \Psi_p.O_S, k\Omega + \Psi_p.O_v + \Psi_p.\delta)$ est réservé à la transmission des flux de la section Ψ_p. En d'autres termes, chaque trame d'un flux de Ψ_p n'est disponible qu'à un instant $r_{i,j}$ tel que $r_{i,j} \geq k\Omega + \Psi_p.O_v$ et $r_{i,j} + C_i \leq k\Omega + \Psi_p.O_v + \Psi_p.\delta$. Dans chaque section, O_v est affecté comme un offset partiel à chaque flux de la section. Cependant, à l'intérieur de chaque section, certains flux peuvent avoir besoin d'occuper le même cycle. Dans ces cas, l'un des flux doit être configuré pour n'être disponible qu'après la fin de l'exécution de l'autre. Pour cela, un offset interne O_{I_i} est ajouté à l'offset final. Par conséquent, l'offset final peut être obtenu à partir de l'équation 5.10.

$$O_i = \Omega O_{C_i} + O_{v_i} + O_{I_i} \tag{5.10}$$

De plus, les sections doivent commencer après la fin des sections précédentes afin de garantir l'isolation entre les sections. De plus, pour que chaque section s'intègre de manière optimale dans le PGCD global, une section doit commencer à l'instant où la section précédente se termine, évitant ainsi la création de petits espaces où aucune section ne pourrait s'intégrer. Cela signifie que, si $\Psi_1.O_S = 0$, alors $\Psi_2.O_S = \Psi_1.\delta$, $\Psi_3.O_S = \Psi_1.\delta + \Psi_2.\delta$, et ainsi de suite.

Pour mieux comprendre les concepts abordés, nous allons décrire un exemple. Nous considérons les flux suivants :

$$v_1 = (T_1 = 24, C_1 = 2)$$
$$v_2 = (T_2 = 16, C_2 = 1)$$
$$v_3 = (T_3 = 16, C_3 = 3)$$
$$v_4 = (T_4 = 16, C_4 = 3)$$

Dans ce système, $\Omega = 8$. Par conséquent, $T_{S_1} = 3$ et $T_{S_2} = T_{S_3} = T_{S_4} = 2$. Nous travaillons avec deux sections : une pour $p = 2$ (Ψ_2) et une pour $p = 3$ (Ψ_3). Ψ_3 ne contient que v_1, et les autres flux sont contenus dans Ψ_2. En choisissant le cycle 0 pour v_1 et v_2, nous choisissons le cycle 1 pour v_4 et à nouveau le cycle 0 pour v_3. Puisque v_2 et v_3 partagent le même cycle dans la même section, v_3 doit avoir un offset interne $O_{I_3} = C_2$. Enfin, nous pouvons calculer les tailles des sections. La taille de Ψ_3 est de 2, et la taille de Ψ_2 est de 4. Nous pouvons arbitrairement choisir que $\Psi_3.O_S$ soit égal à 0, et donc $\Psi_2.O_v = \Psi_3.\delta = 2$. Par conséquent, chaque offset est calculé selon l'équation 5.10 : $O_1 = 0, O_2 = 2, O_3 = 3$ et $O_4 = 10$. La représentation visuelle de ces calculs peut être observée Figure 5.4. De plus, on peut observer à partir de la Figure 5.5, la transmission de trames divisée en cycles issues de données de la Figure 5.4. Le flux v_1 devra partager un cycle avec tous les autres flux, mais v_2 et v_4, par exemple, ne sont jamais dans le même cycle.

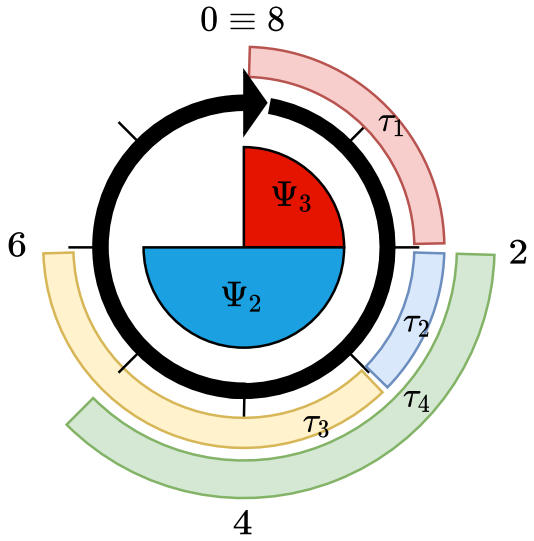

FIGURE 5.4 – Cercle modulaire de taille Ω avec les flux du cas d'exemple

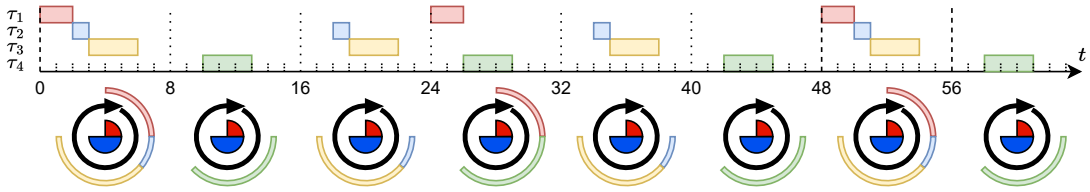

FIGURE 5.5 – Distribution des trames divisées en cycles

Lemme 5.4.4. *Si* $C_i \leq \Omega \forall i$ *et que les flux sont affectés à des sections telles que* $\sum \Psi_p.\delta \leq \Omega$ *pour toutes les sections (en considérant* $\Psi_p.\delta = 0$ *pour les sections vides), alors les flux sont sans congestion, et le temps d'attente dans le nœud est nul.*

Démonstration. La preuve repose sur l'analyse de chaque paire de flux possible dans un système, en prouvant qu'elles ne peuvent jamais provoquer de retard de leurs transmissions mutuelles. Par conséquent, par conception, et en tenant compte du lemme 5.4.2, l'ensemble du système composé de n flux ne présentera aucun délai d'attente. En considérant un flux v_i quelconque, il y a trois possibilités d'analyser un second flux v_j :

1. v_j est affecté à une section différente de v_i ;
2. v_j est affecté à la même section mais dans un cycle non congruent à v_i ;
3. v_j est affecté à la même section et à un cycle congruent à v_i.

Dans le premier cas, d'après la définition des sections, la transmission des trames appartenant aux flux à l'intérieur d'une section est entièrement contenue dans cette section. En considérant que la somme des tailles des sections est au plus Ω, sachant que $\text{PGCD}T_i, T_j$ est au moins égal à Ω, alors nous remarquons que les conditions de l'équation 5.5 seront respectées.

Pour le second cas, sachant que $\sum \Psi_p.\delta \leq \Omega$, que la transmission de trame est contenue dans les limites de sa section, et que v_i et v_j ne s'exécutent pas dans le même cycle, alors chaque trame

appartenant à un flux sera toujours transmise à l'intérieur d'un cycle différent. Par conséquent, chaque flux de v_i aura terminé son exécution avant que v_j ne commence, et vice versa.

Pour le troisième cas, grâce à l'offset interne O_I et au fait que la taille d'une section n'est pas supérieure à Ω, alors chaque trame de v_i aura terminé sa transmission avant que v_j ne commence, et vice versa. □

Si les conditions du Lemme 5.4.4 ne sont pas satisfaites, GCD+ ne peut pas garantir l'absence de congestion. Cependant, GCD+ peut quand même fournir une affectation d'offset. Par exemple, si l'on ajoute v_5 à l'exemple tel que $T_5 = 40$ et $C_5 = 3$, GCD+ devrait affecter v_5 à une nouvelle section, avec $p = 5$. Par conséquent, Ψ_5 commencerait après Ψ_2 (donc $\Psi_5.O_v = 6$) et aurait une taille de 3 unités de temps, créant un chevauchement avec le début de Ψ_3. À la lumière du Lemme 5.4.4, il devient clair que les choix de l'algorithme doivent viser à maintenir les tailles des sections aussi petites que possible.

5.4.2 *GCD#*

Avec les concepts pour un seul nœud issus de GCD+, GCD# adapte la méthode afin d'ordonnancer les flux dans un réseau de nœuds. Pour cette adaptation, nous avons considéré certaines propriétés, qui sont couramment utilisées dans le contexte d'un réseau temps réel, afin de définir un contexte dans lequel GCD# devrait bien fonctionner. Il ne s'agit pas de contraintes fortes, car GCD# peut être exécuté sans que ces propriétés soient vraies.

1. Le temps de transmission maximal doit être inférieur à Ω, le PGCD de toutes les périodes des flux.
2. Tous les nœuds fonctionnent à la même vitesse, ce qui rend la valeur de C_i constante sur l'ensemble du réseau.
3. Comme nous nous concentrons uniquement sur les points d'ordonnancement, seuls les ports de sortie des commutateurs et des stations terminales sont considérés, et seront désignés comme des nœuds du réseau. Les flux traversant le même nœud peuvent interférer les uns avec les autres, tandis que les flux qui ne partagent jamais un nœud commun ne peuvent pas interférer directement les uns avec les autres (même s'ils traversent le même commutateur).
4. Le temps nécessaire à un nœud pour recevoir et renvoyer une trame est constant (coût "Store & Forward", ou S&F). Ce temps dépend du temps nécessaire au traitement de la plus grande trame par un nœud et est constant sur l'ensemble du réseau.
5. Le temps nécessaire au transfert d'une trame entre deux nœuds est nul, mais il est considéré dans le délai "Store & Forward".
6. Le délai de bout en bout, c'est-à-dire le temps nécessaire à une trame pour arriver à destination, par rapport au temps où elle a été envoyée depuis son nœud source, est inférieur à Ω.

Les hypothèses 2, 3, 5 et 6 sont courantes dans la littérature. L'hypothèse 1, couplée à l'hypothèse 4, est due au fait que GCD# travaille globalement sur les flux en attribuant un offset unique à

chaque flux au niveau de son nœud source. Cet offset sera incrémenté du délai S&F à chaque saut du réseau. Comme il y a un seul cercle de PGCD utilisé pour calculer les offsets, si une trame ne peut pas tenir dans ce cercle, alors la congestion ne peut pas être évitée par GCD#. Dans ce cas, comme le montre la section 5.5.1, GCD# fournit quand même des offsets, mais l'existence d'une congestion est hautement probable. Notons que si un tel événement se produit localement sur un port de sortie d'un nœud, il est impossible d'éviter la congestion, même pour des méthodes optimales comme SMT. On peut noter que dans les réseaux temps réels, l'hypothèse 1 est généralement satisfaite.

L'algorithme se compose de cinq phases : choisir la section pour chaque flux ; choisir un cycle pour chaque flux dans une section donnée ; choisir un offset interne pour chaque flux qui partage une section et un cycle avec un autre ; calculer les tailles des sections ; et calculer l'offset final.

Choisir la section

La phase de choix de sections peut être caractérisée comme une variante du problème de *bin-packing* [31]. Le but est de déterminer le *bin* approprié pour chaque flux pour que la taille cumulée de tous les *bins* soit minimisée. Il est important de noter que chaque flux aura un poids spécifique, en fonction de son placement dans la section. Pour y parvenir, GCD# utilise un algorithme *Best Fit Decreasing*, comme expliqué dans les paragraphes suivants.

Chaque flux avec $T_v = 1$ doit être affecté à Ψ_1, et chaque flux pour lequel T_v est composé d'un seul nombre premier p doit être placé dans la section correspondante Ψ_p. Parmi les autres flux, celui qui a la plus grande valeur pour C est le premier à être analysé. Si aucune des sections potentielles ne comporte de flux qui lui sont affectés, la préférence est donnée au plus petit premier p_L, car il a la plus grande chance de composer également d'autres sous-périodes. Ce faisant, nous augmentons la probabilité de partager la section avec d'autres flux. Cette stratégie permet d'éviter la création de sections superflues, minimisant ainsi les tailles globales des sections (Lemme 5.4.4).

Si une section potentielle est déjà occupée par un flux, la préférence est donnée à la section occupée. Cela pourrait réduire la nécessité d'augmenter les tailles des sections. Ainsi, l'ajout du flux à une nouvelle section augmenterait certainement la taille de la section de 0 à C_i, alors que seul le pire des cas nécessiterait que la taille de la section déjà occupée soit augmentée de C_i. Cependant, dans des cas plus favorables, la possibilité d'alterner les cycles pourrait entraîner une augmentation plus modeste (voire inexistante) de la taille de la section.

Parmi les sections occupées, la préférence est donnée en fonction d'une fonction de score. Le score représente une estimation du nombre de cycles occupés dans la section (puisqu'aucun cycle n'a été défini) par rapport à un flux candidat v_i, et est donné par l'équation 5.11. Cette équation est basée sur le principe que chaque flux $v_j \in \Psi_p$ aura une chance de $\frac{1}{\text{PGCD}\{T_{v_i}, T_{v_j}\}}$ d'occuper un cycle congruent au cycle affecté à v_i. Cependant, si un score est supérieur à la valeur de 1, nous considérons qu'il perd son sens. En effet, il existe une chance que chaque cycle soit déjà occupé, et les meilleurs choix pour affecter v_i dans ce cas nécessiteraient des informations que nous n'avons pas encore. Par conséquent, le score est limité à 1.

$$\Psi_p.score(v_i) = \min\left(1, \sum_{\forall i | S_j \in \Psi_p} \frac{1}{\text{PGCD}\{T_{S_i}, T_{S_j}\}}\right) \tag{5.11}$$

Choisir un cycle

Une fois que chaque flux a été affecté à une section, l'étape suivante consiste à choisir les cycles pour chaque flux. Pour chaque section, ses flux sont analysés selon un ordre décroissant de la taille de leurs messages, car les messages volumineux ont un impact plus important. Comme nous considérons que le délai de bout en bout est inférieur à Ω, chaque flux v_i n'occupera que des cycles congruents à une seule valeur $k \mod T_{v_i}$.

Pour se faire, un vecteur de possibilités est construit, de taille Tv_i, rempli de zéros. Ensuite, en analysant tous les autres flux v_j de la même section qui ont déjà été affectés, et en vérifiant qu'ils croisent bien le flux analysé S_i (partageant le même nœud à un moment donné), C_j est ajouté à chaque position du vecteur de possibilités qui sont congruentes à $O_{C_j} \mod \text{PGCD}\{T_{v_i}, T_{v_j}\}$. Affecter S_i à l'une de ces positions garantit que v_i et v_j partageront éventuellement le même cycle (Lemme 5.4.3). Par conséquent, le meilleur choix est la position dans le vecteur dont la somme est la plus faible (idéalement, 0).

Choisir un offset interne

Une fois que chaque flux a un cycle, les offsets internes sont choisis pour chaque flux. Pour attribuer un offset interne à un flux v_i, il faut considérer les flux v_j qui croisent v_i, qui sont dans la même section, ont un cycle congruent ($\mod \text{PGCD}\{T_{v_i}, T_{v_j}\}$) et se sont déjà vu attribuer un offset interne O_I. S'il n'existe pas de tel v_j, alors $O_{I_i} = 0$. La constante "Store & Forward" (S&F) doit être considérée. Si v_j partage un certain nœud N_k avec v_i, et que N_k est le n-ième nœud pour les deux chemins respectifs, alors les deux flux subiront le même délai S&F. Dans ce cas, les flux doivent être analysés comme s'il n'y avait pas de S&F, et l'offset interne est choisi de telle sorte que les conditions suivantes soient vraies $\forall j$:

$$\begin{aligned} O_{I_i} \leq O_{I_j} &\Rightarrow O_{I_i} + C_i \leq O_{I_j} \\ O_{I_i} > O_{I_j} &\Rightarrow O_{I_i} \geq O_{I_j} + C_j \end{aligned} \tag{5.12}$$

Cependant, si N_k est le n-ième nœud pour un chemin de v_i mais le $(n+1)$-ième pour v_j, cela signifie que, lorsque leurs trames arrivent à N_k, v_j sera en retard d'un délai S&F par rapport à v_i. Par conséquent, pour chaque flux v_j qui satisfait aux conditions susmentionnées, il faut apporter une correction. Si v_j est en avance par rapport à v_i, nous devons soustraire le nombre approprié de délais S&F de O_{I_j} dans les conditions de l'équation 5.12. S'il est en retard, nous devons ajouter le nombre de délais S&F à O_{I_j}.

Calculer les tailles des sections et l'offset final

Pour calculer la taille de chaque section, l'ordre d'exécution entre les sections doit être défini. GCD# choisit arbitrairement l'ordre en fonction du nombre p lié à la section : si $p_1 < p_2$, alors Ψ_{p_1} précède Ψ_{p_2}. La taille d'une section Ψ_{p_i} peut être calculée en deux parties, illustrées dans l'équation 5.13 :

1. Pour chaque flux de la section, les temps de transmission sont ajoutés à l'offset interne respectif. Parmi les sommes pour chaque flux, la valeur maximale est la première partie de la taille de la section (équivalent à l'équation 5.9).

2. Une marge est calculée pour tenir compte des différences possibles de temps S&F. Si nous analysons Ψ_{p_i} qui précède Ψ_{p_j}, nous devons tenir compte du délai S&F maximal possible qu'un flux $v_i \in \Psi_{p_i}$ aura par rapport à tout autre flux $v_j \in \Psi_{p_j}$. Ensuite, ajouter cette valeur à la taille de la section, pour chaque nœud, afin qu'aucun flux de la section suivante ne soit transmis avant un flux de la section précédente.

$$\Psi.\delta = \max_{\forall i | v_i \in \Psi} (O_{I_i} + C_i) + marge_L \tag{5.13}$$

Après avoir effectué cette opération pour chaque section, y compris la dernière (qui doit être considérée comme précédant la première section en raison de la cyclicité), chaque offset O_S peut être calculé. Pour chaque section, O_S est la somme des tailles des sections précédentes (sans tenir compte de la cyclicité). En d'autres termes, le O_S de la première section est 0 ; pour la deuxième, il est égal à la taille de la première section ; pour la troisième, il est égal à la somme des tailles des deux premières sections ; et ainsi de suite. Par conséquent, les offsets pour chaque flux peuvent enfin être calculés selon l'équation 5.10.

5.5 EXPÉRIMENTATIONS

Dans cette section, nous allons d'abord travailler sur un réseau synthétique afin d'avoir un meilleur aperçu de GCD# dans le cas de commutateurs fortement chargés ou de configurations qui ne respectent pas les propriétés mises en évidence dans la section 5.4.2. Ensuite, nous comparerons les performances de GCD# à la méthode formelle SMT décrite dans [33] (avec le solveur Z3). Nous avons spécifiquement sélectionné cette méthode, car à notre connaissance, il n'existe aucune méthode heuristique disponible et la méthode SMT est utilisée dans plusieurs heuristiques comme base pour explorer les solutions. Toutes les expériences sont codées en Python 3.11 fonctionnant sous Fedora Linux 38 sur un processeur AMD Ryzen 9 5900HS (jusqu'à 4,6 Ghz sur un cœur) avec 24 Go de RAM. Tous les réseaux utilisés ci-après utiliseront la spécification suivante : des liens de 1 Gb/s, ce qui donne un temps unitaire de 1 ns.

5.5.1 *Pousser GCD# à ses limites*

Nous introduisons ci-après deux cas de test pour démontrer les performances de GCD# dans des conditions extrêmes. Les objectifs sont d'étudier les résultats de GCD# lorsque la charge atteint 100% dans un nœud et de comprendre son comportement lorsqu'un temps de transmission dépasse le PGCD de toutes les périodes. Ces cas de test utilisent la topologie représentée dans la Figure 5.2, avec trois flux suivant un chemin partiellement partagé. Leurs paramètres sont détaillés dans le Tableau 5.2. Dans chaque cas de test, la politique d'ordonnancement du trafic de chaque nœud est non préemptive et basée sur le *rate monotonic*. Nous supposons aussi un délai S&F égal au temps de transmission de la plus grande trame (3 unités de temps dans les deux cas de test). En raison du choix des paramètres, un solveur SMT ne serait pas en mesure de produire des résultats.

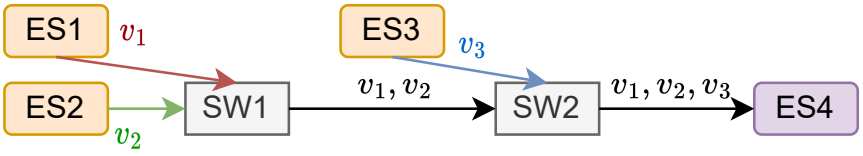

FIGURE 5.6 – Topologie utilisée dans le cas synthétique pour évaluer les collisions de GCD#

Commutateur fortement chargé

L'ordonnancement se complique avec des charges de trafic élevées. Comme GCD# ne peut pas toujours produire des solutions optimales, il produit parfois une solution qui n'est pas sans collision. Le résultat de la simulation pour le cas de test 1 est présenté dans la Figure 5.7. Les délais de bout en bout suivants peuvent être déduits : 10 pour v_1, 7 pour v_2 et 6 pour v_3. Le cycle qui se répétera dans chaque nœud est mis en évidence avec un fond gris. Une partie blanche précédant une section grise correspond à un temps d'inactivité acyclique.

Soit n le nombre de sauts. Le délai de bout en bout dans un système sans collision est $(n-1)$ fois la durée de S&F, plus le temps de transmission de la trame analysée. Pour notre scénario, cela aurait été respectivement 8, 7, 6 unités de temps pour v_1, v_2 et v_3. La collision entraîne une augmentation de 25% des délais de bout en bout pour v_1 s'il n'y avait eu aucune collision. À partir de cette simulation, nous pouvons déduire les intervalles d'ordonnancement telles qu'elles sont anticipées dans un réseau TSN IEEE 802.1Q [57] (Vu dans le chapitre 2).

Taille de la trame supérieure au PGCD des périodes

Dans ce second cas de test, nous souhaitons observer comment GCD# répond aux configurations où la valeur de Ω est inférieure au temps de transmission de la plus grande trame. La simulation correspondante est représentée dans la Figure 5.8. D'après les paramètres fournis dans le Tableau 5.2, le PGCD est déterminé à 2. En revanche, le temps de transmission pour la plus grande trame (de v_3) est de 3. Pour cette simulation, nous avons choisi d'afficher uniquement le port de sortie de SW2. D'après le diagramme, on peut voir que la plus grande trame v_3, entre en collision avec v_1 et v_2. Ces collisions sont inévitables, car l'espace maximal disponible dans chaque

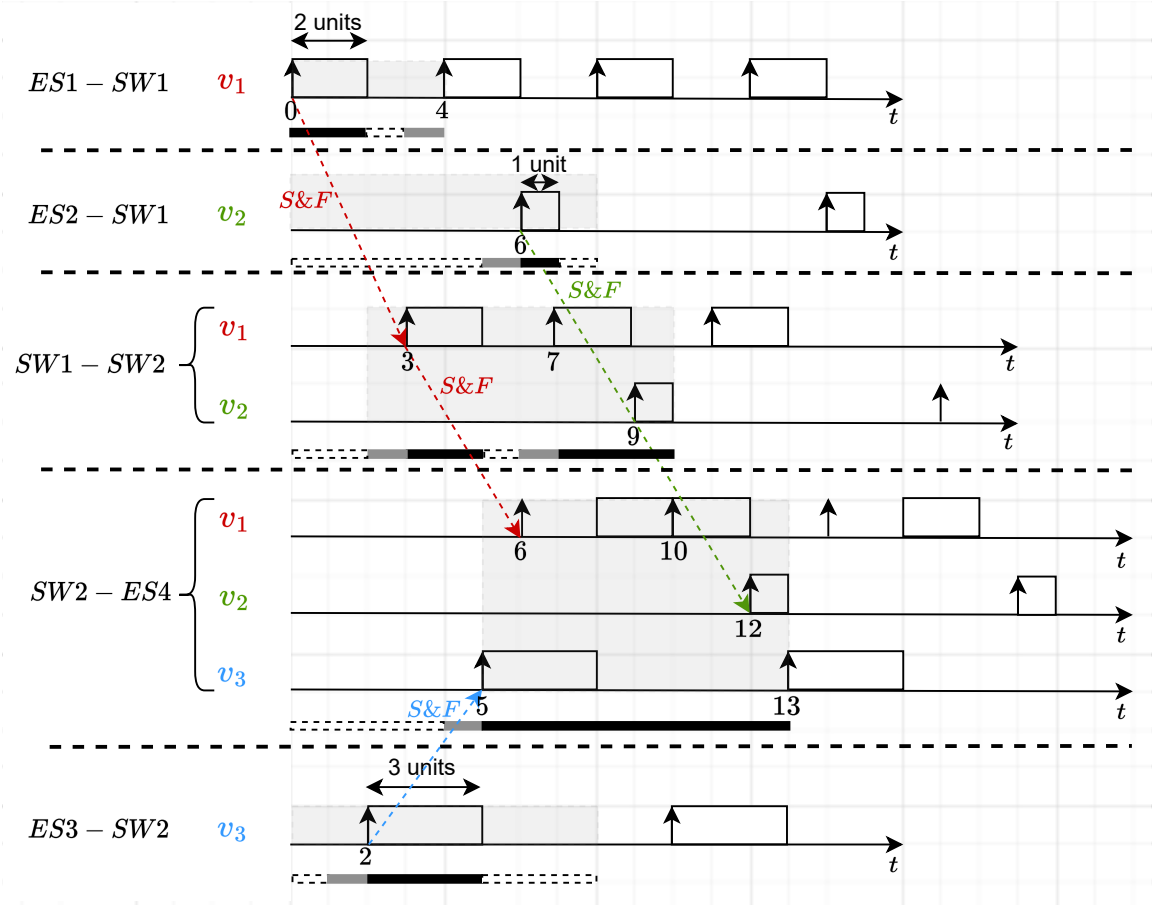

FIGURE 5.7 – Simulation du cas de test 1 décrit dans le Tableau 5.2

cercle de PGCD est de 2. Dans ce cas, les délais de bout en bout dans le pire des cas pour v_1 et v_2 sont affectés et leurs délais respectifs sont tous deux de 10. Cela signifie que l'augmentation des délais de bout en bout dans le pire des cas est égale à 42% par rapport au cas sans collisions.

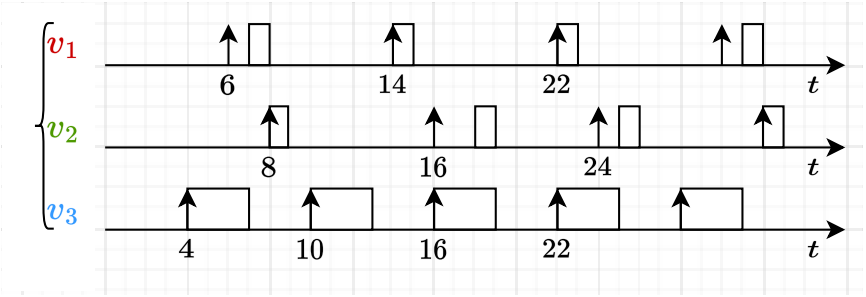

FIGURE 5.8 – Simulation du cas de test 2 décrit dans le Tableau 5.2

TABLE 5.2 – Paramètres des différents flux dans les cas de test spécifiques à GCD#

Cas d'exemple	Flux	Temps de transmission	Période	Offsets GCD#
Cas d'exemple 1	1	2	4	0
	2	1	8	6
	3	3	8	2
Cas d'exemple 2	1	1	8	0
	2	1	8	2
	3	3	6	1

5.5.2 *GCD# comparé à SMT*

Configuration synthétique

L'objectif de cette étude de cas est de comparer GCD# à SMT lorsqu'il est appliqué à une configuration synthétique compacte. Pour cette étude, nous choisissons d'utiliser une configuration provenant de [113] (topologie et flux). Tous les flux de ce réseau proviennent des systèmes terminaux (ES) $[1, 2, 3]$, transitent par deux commutateurs et atteignent finalement les ES $[4, 5, 6]$. Chaque flux est expédié à partir des ES $[1, 2, 3]$ et peut arriver à un ou plusieurs d'autres systèmes terminaux.

La sortie du solveur SMT peut fluctuer en fonction de l'identifiant attribué aux contraintes (c'est-à-dire la séquence dans laquelle les contraintes sont ajoutées au solveur ou l'effet de la suppression et de la réinsertion des contraintes), alors que les résultats de GCD# restent constants. Comme la même configuration peut engendrer plusieurs résultats, chaque expérimentation sur SMT est répétée 10 000 fois avec des identifiants de contrainte différents. En moyenne, GCD# est meilleur de 14,63% que SMT pour les délais de bout en bout, s'exécutant en 53 ms en moyenne pour SMT, contre moins de 1 ms pour GCD#. Les deux parviennent à garantir l'*absence de congestion*, mais GCD# est plus performant que SMT. En effet, SMT s'arrête à la première affectation d'offset réalisable, sans essayer de minimiser les délais de bout en bout. Pourtant, GCD# fonctionne avec un handicap puisqu'il considère le délai S&F maximal pour chaque trame.

Cas de test Orion

Orion est un vaisseau spatial développé par la NASA (vu dans la section 1.6). La topologie de son réseau est décrite dans [85]. Des configurations de flux ont été proposées dans [113] pour l'analyse des délais de bout en bout à l'aide du *Network Calculus*. Nous avons pris les chemins fournis puis nous avons générés des flux avec les paramètres aléatoires suivants : taille des trames (72 à 1542 octets, conformément aux normes *Ethernet* et IEEE 802.1Q), période, délai pire cas (supérieur à la période) et chemin (dans un ensemble de chemins prédéfinis). La taille des flux suit une loi de distribution uniforme, tandis que les périodes suivent trois lois normales pondérées. Ainsi, nous avons essayé de nous conformer à la répartition des caractéristiques fournies dans l'article [113].

La topologie du réseau est représentée dans la Figure 5.9. Les périodes ont été choisies comme multiples de 7500, suivant une distribution aléatoire, afin de réduire considérablement le temps de calcul de SMT. En conséquence, nous avons obtenu 625 µs comme PGCD. Le délai S&F a été plafonné à 12,336 µs (trame *Ethernet* maximale avec intervalle entre les trames), respectant ainsi la condition de temps d'exécution de GCD# (aucune trame ne doit dépasser le PGCD). Le réseau est chargé de 100 flux, ayant un expéditeur et une ou plusieurs destinations.

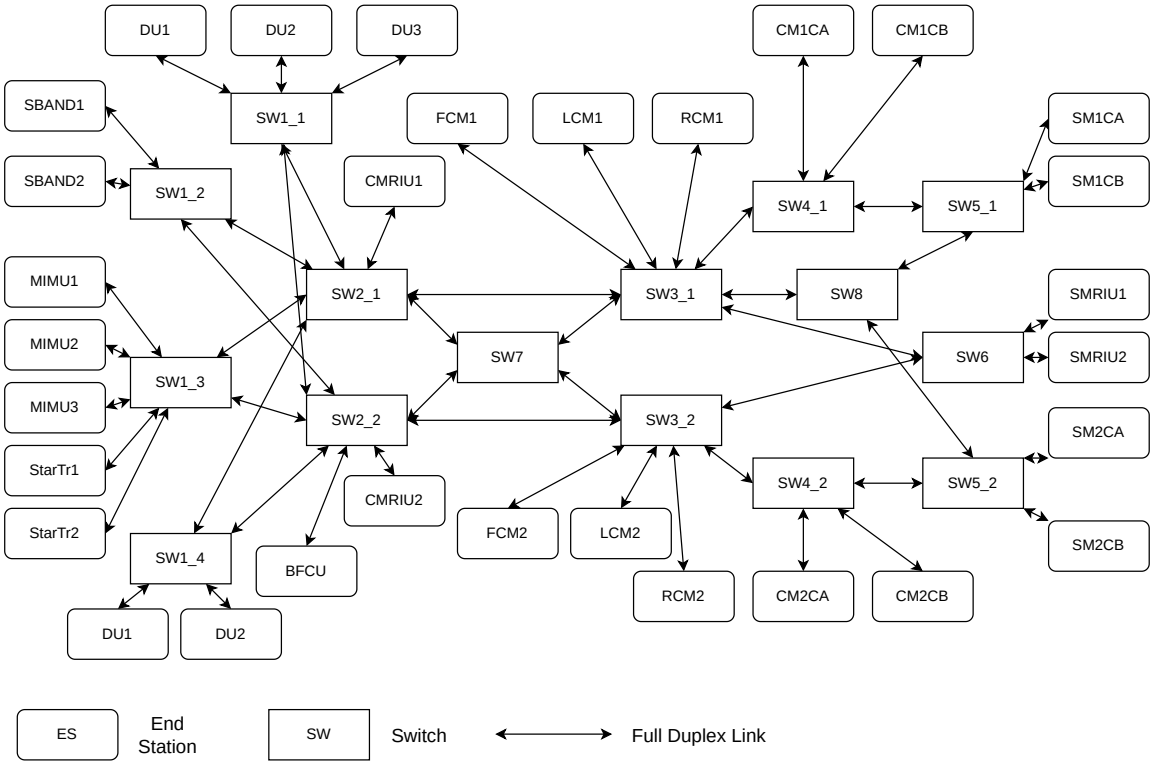

FIGURE 5.9 – Topologie d'Orion d'après [113]

La réalisation de 100 évaluations a conduit aux résultats suivants :

— Les délais de bout en bout sont en moyenne 72,82% meilleurs avec GCD# par rapport à SMT ;
— En moyenne, les délais de bout en bout de SMT atteignent 61,32% de leur délai, contre 0,19% pour GCD# ;
— Le temps de calcul moyen de GCD# est d'environ 0,02 s, alors que SMT prend 188 s pour construire et 47 s pour résoudre.

Dans la Figure 5.10, nous présentons les principaux résultats de la comparaison entre SMT et GCD# pour un cas de test représentatif. Sur le graphique de gauche, pour la plupart des destinations (plutôt que des flux, car un flux peut avoir plusieurs destinations), GCD# a de meilleurs résultats en termes de latences de bout en bout, alors que SMT n'est pas efficace pour la plupart des destinations. Sur la droite, les délais de bout en bout sont tracés par rapport à leurs délais. On peut voir que les délais de GCD# sont empilés en dessous de 5% du délai, tandis que les délais de SMT peuvent aller jusqu'à 90% du délai.

La principale différence entre SMT et GCD# concernant le délai de bout en bout est due au fait que SMT attribue des offsets arbitraires à chaque nœud, tandis que GCD# attribue des offsets

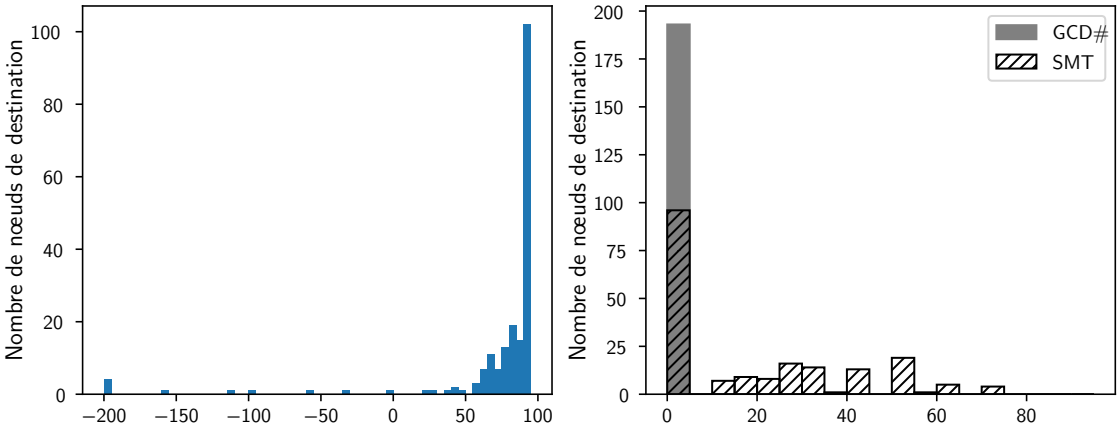

FIGURE 5.10 – Comparaison des performances entre GCD# et SMT sur une execution pour 100 flux

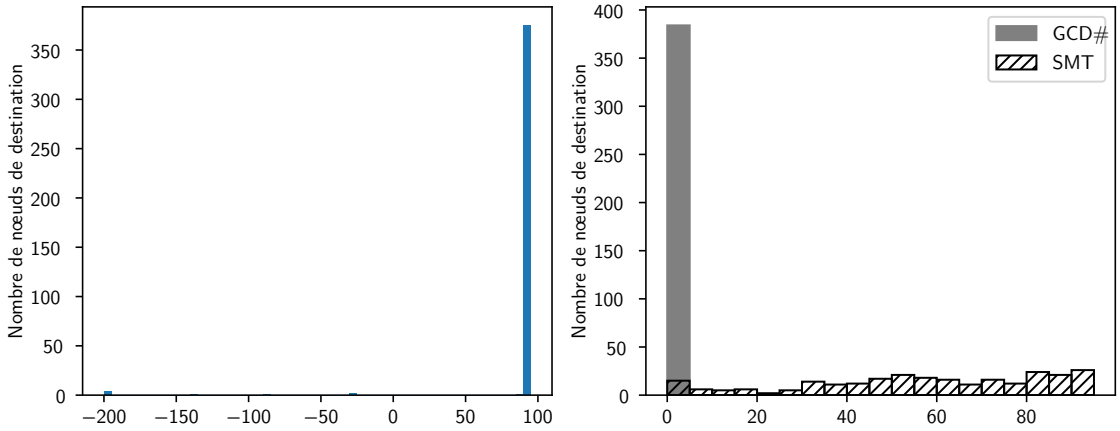

FIGURE 5.11 – Comparaison des performances entre GCD# et SMT sur une execution pour 200 flux

congruents pour chaque nœud suivant dans le réseau. Après avoir effectué 10 000 tests, nous avons constaté que GCD# fournit des solutions avec congestion dans 1,34% des cas. Par construction, la méthode SMT ne donne pas de résultats avec congestion. Nous avons mené une autre expérience impliquant 200 flux et 10 exécutions. L'augmentation du nombre de flux dans SMT pose un défi considérable, car la *contrainte sur trames* entraîne une augmentation exponentielle du nombre de contraintes à formuler. En moyenne, il a fallu 680 s pour générer les contraintes et 348 s pour les résoudre dans le cas de SMT. En revanche, GCD# n'a eu besoin que de 63 ms pour identifier une solution, tout en offrant une réduction du délai de bout en bout de 90%. Les valeurs de la Figure 5.11 sont encore plus favorables à GCD# que dans le cas précédent. Sur le côté gauche, seule une poignée de délais ont tendance à être meilleurs avec SMT qu'avec GCD#, alors que la grande majorité des destinations affichent des performances supérieures avec GCD#. Par rapport aux délais, GCD# positionne systématiquement toutes les latences de bout en bout dans une plage de 0 à 5%, alors que les résultats de SMT sont largement dispersés sur le spectre des pourcentages.

5.6 CONCLUSION ET PERSPECTIVES

Dans ce chapitre, nous avons abordé la problématique de cyclicité dans l'ordonnancement réseau temps réel avec *offsets*. Nous avons prouvé pourquoi l'hypothèse considérant l'intervalle $[0, H)$ comme étant toujours cyclique dans un ordonnancement avec *offsets* ne peut pas toujours être vérifiée. Néanmoins, nous avons démontré que par l'application conjointe de la *contrainte de non-congestion* et la *contrainte sur trames* permettent de garantir que l'intervalle cyclique $[0, H)$ est suffisant.

Par ailleurs, nous avons proposé une nouvelle heuristique GCD# qui est capable de générer des ordonnancements même en présence de congestions, ce qui la distingue par rapport aux autres méthodes existantes. Les expérimentations conduites sur des réseaux synthétiques et réels ont démontré l'efficacité de GCD#, notamment en termes de réduction des délais de bout en bout et du temps de calcul par rapport à la méthode SMT.

Différentes perspectives peuvent être étudiées concernant GCD# :

— adapter GCD# pour des réseaux plus complexes, telles que des ports de sortie avec des vitesses de transmission différentes ou avec des flux pouvant boucler ;
— optimiser la méthode avec un S&F de durée variable et adaptée à la taille de chaque trame ;
— rendre la méthode perméable à des pertes de paquets ;
— permettre d'affecter des priorités sur les flux pour minimiser les délais de bout en bout pour des flux très critiques

Les résultats de ce chapitre ont été présentés à la conférence RTSS [44].

CONCLUSION ET PERSPECTIVES

Dans ce manuscrit, nous avons abordé des problématiques liées à la distribution des systèmes embarqués et au déterminisme des réseaux. Nous avons étudié la perte de trames dans un réseau, de façon agnostique. De plus, nous avons examiné un problème de cyclicité dans les ordonnancements produits par des méthodes développées pour les réseaux *offset-free*, en particulier les réseaux conformes aux normes *Time Sensitive Networking* (TSN).

RÉSULTATS

Après un état de l'art sur le dimensionnement des files d'attente des ports de sortie et les méthodes d'ordonnancement dans les réseaux temps réel à *offset-free*, nous avons obtenu les résultats suivants :

— méthode *Buffer Dimensioning per Frame* (BDF) : Nous avons proposé la méthode BDF pour déterminer la quantité maximale de trames pouvant se trouver dans un port de sortie d'un commutateur réseau. Cette méthode repose sur l'élaboration d'un scénario pessimiste permettant de modéliser, pour tout instant t, le nombre de trames entrées dans le port de sortie et distribuées. La plus grande différence entre ces deux résultats donne la quantité maximale de trames. Nous avons comparé les résultats de cette méthode avec ceux d'une méthode naïve (occupation pire cas en bits des ports de sortie divisée par la taille de trame la plus petite) sur la configuration de l'Airbus A380. Nous avons observé une réduction de l'estimation du nombre de trames maximal de 76,3 % en moyenne ;

— problème de génération d'ordonnancement : Nous avons identifié un problème dans certaines méthodes de génération d'ordonnancement pour les réseaux temps réel à *offset-free*. La plage d'étude choisie, une hyperpériode H sur l'intervalle $[0, H)$, est généralement insuffisante et peut entraîner des problèmes dues à des trames non observées. Nous avons prouvé que, pour limiter l'analyse de la recherche de solutions à l'intervalle $[0, H)$, il est nécessaire d'utiliser deux contraintes conjointes : la contrainte de non-congestion (empêchant la transmission simultanée de deux trames sur le même médium physique) et la contrainte sur trames (interdisant un *offset* supérieur à la période de la trame). Nous avons également proposé une méthode heuristique, GCD#, qui n'a pas besoin de respecter ces contraintes. Nous avons évalué une méthode de génération d'ordonnancement basée sur SMT contre GCD# sur des réseaux synthétiques et sur la configuration de la navette Orion. Nous avons constaté une amélioration considérable des temps de calcul et des délais de bout en bout pour GCD#. Cependant, GCD# peut produire des configurations avec congestion, alors que la méthode SMT, par nature, cherche toujours à fournir une solution sans congestion.

PERSPECTIVES

Plusieurs perspectives restent ouvertes concernant les contributions de cette thèse :

— Méthode BDF : Le scénario fourni est pessimiste et ne peut pas se produire en réalité. Il serait possible de chercher un moyen d'exclure des cas de figure irréalistes. Bien que la méthode BDF soit agnostique vis-à-vis des normes réseau employées, sa précision pourrait

être améliorée en prenant en compte les spécificités des normes utilisées, telles que les normes TSN ou AFDX.

— Méthode GCD# : La méthode pourrait être améliorée en adaptant la valeur de *Store & Forward* aux trames étudiées, plutôt que d'utiliser une valeur fixe. Cela réduirait les délais d'attente des plus petites trames dans les ports de sortie et améliorerait les délais de transmission de bout en bout. Il serait également possible d'optimiser la méthode pour supporter des flux qui bouclent dans le réseau ou qui se séparent dans un commutateur pour se rejoindre plus loin.

BIBLIOGRAPHIE

[1] Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont et Christian Fraboul. « Model for worst case delay analysis of an AFDX network using timed automata ». In : *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. 2010, p. 1-4. DOI : 10.1109/ETFA.2010.5641124 (cf. p. 36, 49).

[2] Giuliana Alderisi, Alfio Caltabiano, Giancarlo Vasta, Giancarlo Iannizzotto, Till Steinbach et Lucia Lo 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)*. 2012, p. 187-194. DOI : 10.1109/VNC.2012.6407430 (cf. p. 37).

[3] ARINC. *664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*. Rapp. tech. 2009 (cf. p. 19).

[4] *ARINC 664, Aircraft Data Network, Parts 1,2 7. Technical report, ARINC specification 664*. 2002-2005 (cf. p. 5, 80).

[5] *AS6802A : Time-Triggered Ethernet*. Rapp. tech. SAE, 2023 (cf. p. 19).

[6] Mohammad Ashjaei, Gaetano Patti, Moris Behnam, Thomas Nolte, Giuliana Alderisi et Lucia Lo Bello. « Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support ». In : *Real-Time Systems* 53 (2017), p. 526-577 (cf. p. 36).

[7] Bell System Practices AT&T. *Registration Interface—Selection and General Information*. Rapp. tech. 1976 (cf. p. 17).

[8] Philip Axer, Daniel Thiele, Rolf Ernst et Jonas Diemer. « Exploiting shaper context to improve performance bounds of Ethernet AVB networks ». In : *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2014, p. 1-6. DOI : 10.1145/2593069.2593136 (cf. p. 39).

[9] Yohan Baga, Fakhreddine Ghaffari, Etienne Zante, Michael Nahmiyace et David Declercq. « Worst frame backlog estimation in an avionics full-duplex switched ethernet end-system ». In : *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. 2016, p. 1-10. DOI : 10.1109/DASC.2016.7777990 (cf. p. 42).

[10] Yohan Baga, Morgane Richaud, Fakhreddine Ghaffari, Etienne Zante, David Declercq et Michael Nahmiyace. « Probabilistic model of AFDX frames reception for end system backlog assessment ». In : *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*. 2017, p. 1-6. DOI : 10.1109/SIES.2017.7993400 (cf. p. 42).

[11] Clark Barrett et Cesare Tinelli. « Satisfiability modulo theories ». In : *Handbook of model checking* (2018), p. 305-343 (cf. p. 43, 70).

[12] Michael A. Beck et Jens Schmitt. « The DISCO stochastic network calculator version 1.0 : when waiting comes to an end ». In : *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*. ValueTools '13. Torino, Italy : ICST (Institute for Computer Sciences, Social-Informatics et Telecommunications Engineering), 2013, p. 282-285. ISBN : 9781936968480. DOI : 10.4108/icst.valuetools.2013.254378. URL : https://doi.org/10.4108/icst.valuetools.2013.254378 (cf. p. 39).

[13] Sofiene Beji, Sardaouna Hamadou, Abdelouahed Gherbi et John Mullins. « SMT-based cost optimization approach for the integration of avionic functions in IMA and TTEthernet architectures ». In : *2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*. IEEE. 2014, p. 165-174. DOI : 10.1109/DS-RT.2014.28 (cf. p. 74, 78).

[14] Chaitanya Belwal et Albert MK Cheng. « Generating bounded task periods for experimental schedulability analysis ». In : *2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing*. IEEE. 2011, p. 249-254. DOI : 10.1109/EUC.2011.39 (cf. p. 79).

[15] Nassima Benammar, Henri Bauer, Frédéric Ridouard et Pascal Richard. « Timing analysis of AVB Ethernet network using the Forward end-to-end Delay Analysis ». In : *Proceedings of the 26th International Conference on Real-Time Networks and Systems*. RTNS '18. Chasseneuil-du-Poitou, France : Association for Computing Machinery, 2018, p. 223-233. ISBN : 9781450364638. DOI : 10.1145/3273905.3273922. URL : https://doi.org/10.1145/3273905.3273922 (cf. p. 40, 41).

[16] Nassima Benammar, Frédéric Ridouard, Henri Bauer et Pascal Richard. « Forward end-to-end delay analysis extension for FP/FIFO policy in AFDX networks ». In : *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2017, p. 1-8. DOI : 10.1109/ETFA.2017.8247606 (cf. p. 40).

[17] Nassima Benammar, Frédéric Ridouard, Henri Bauer et Pascal Richard. « Forward end-to-end delay for AFDX networks ». In : *IEEE Transactions on Industrial Informatics* 14.3 (2017), p. 858-865 (cf. p. 53, 62).

[18] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson et Wang Yi. *UPPAAL—a tool suite for automatic verification of real-time systems*. Springer, 1996 (cf. p. 36).

[19] S. Biyabani, J. Stankovic et K. Ramamritham. « The integration of deadline and criticalness in hard real-time scheduling ». In : *Proceedings. Real-Time Systems Symposium*. Los Alamitos, CA, USA : IEEE Computer Society, déc. 1988, p. 152, 153, 154, 155, 156, 157, 158, 159, 160. DOI : 10.1109/REAL.1988.51111. URL : https://doi.ieeecomputersociety.org/10.1109/REAL.1988.51111 (cf. p. 13).

[20] Anne Bouillard, Bertrand Cottenceau, Bruno Gaujal, Laurent Hardouin, Sébastien Lagrange et Mehdi Lhommeau. « COINC library : a toolbox for the network calculus : invited presentation, extended abstract ». In : ICST, mai 2010. DOI : 10.4108/ICST.VALUETOOLS2009.8045 (cf. p. 39).

[21] Anne Bouillard, Laurent Jouhet et Eric Thierry. « Service curves in Network Calculus : dos and don'ts ». Thèse de doct. INRIA, 2009 (cf. p. 39).

[22] Marc Boyer, Nicolas Navet, Xavier Olive et Eric Thierry. « The PEGASE Project : Precise and Scalable Temporal Analysis for Aerospace Communication Systems with Network Calculus ». In : *Leveraging Applications of Formal Methods, Verification, and Validation*. Sous la dir. de Tiziana Margaria et Bernhard Steffen. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 122-136. ISBN : 978-3-642-16558-0 (cf. p. 39).

[23] Vicent Brocal, Patricia Balbastre, Rafael Ballester et Ismael Ripoll. « Task period selection to minimize hyperperiod ». In : *ETFA2011*. IEEE. 2011, p. 1-4. DOI : 10.1109/ETFA.2011.6059178 (cf. p. 79).

[24] Giorgio C Buttazzo. *Hard real-time computing systems : predictable scheduling algorithms and applications*. T. 24. Springer Science & Business Media, 2011 (cf. p. 13).

[25] Jingyue Cao, Pieter J. L. Cuijpers, Reinder J. Bril et Johan J. Lukkien. « Tight worst-case response-time analysis for ethernet AVB using eligible intervals ». In : *2016 IEEE World Conference on Factory Communication Systems (WFCS)*. 2016, p. 1-8. DOI : 10.1109/WFCS.2016.7496507 (cf. p. 39).

[26] Jingyue Cao, Pieter J.L. Cuijpers, Reinder J. Bril et Johan J. Lukkien. « Independent yet Tight WCRT Analysis for Individual Priority Classes in Ethernet AVB ». In : *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS '16. Brest, France : Association for Computing Machinery, 2016, p. 55-64. ISBN : 9781450347877. DOI : 10.1145/2997465.2997493. URL : https://doi.org/10.1145/2997465.2997493 (cf. p. 39).

[27] Hing-Lun Chan et Michael Norrish. « Proof Pearl : Bounding Least Common Multiples with Triangles ». In : *Interactive Theorem Proving*. Sous la dir. de Jasmin Christian Blanchette et Stephan Merz. Cham : Springer International Publishing, 2016, p. 140-150. ISBN : 978-3-319-43144-4 (cf. p. 69).

[28] Annie Choquet-Geniet et Emmanuel Grolleau. « Minimal schedulability interval for real-time systems of periodic tasks with offsets ». In : *Theoretical computer science* 310.1-3 (2004), p. 117-134. DOI : 10.1016/S0304-3975(03)00362-1 (cf. p. 69, 76).

[29] Edmund M Clarke et E Allen Emerson. « Design and synthesis of synchronization skeletons using branching time temporal logic ». In : *Workshop on logic of programs*. Springer. 1981, p. 52-71 (cf. p. 36).

[30] Rodrigo Coelho, Gerhard Fohler et Jean-Luc Scharbarg. « Dimensioning buffers for AFDX networks with multiple priorities virtual links ». In : *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. 2015, 10A5-1-10A5-12. DOI : 10.1109/DASC.2015.7311495 (cf. p. 42).

[31] Edward G Coffman Jr, Michael R Garey et David S Johnson. « Approximation algorithms for bin-packing—an updated survey ». In : *Algorithm design for computer system design*. Springer, 1984, p. 49-106 (cf. p. 85).

[32] Silviu S Craciunas, Ramon Serna Oliver et Wilfried Steiner. « Formal scheduling constraints for time-sensitive networks ». In : *arXiv preprint arXiv:1712.02246* (2017) (cf. p. 43, 70).

[33] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík et 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. Brest, France : Association for Computing Machinery, 2016, p. 183-192. ISBN : 9781450347877. DOI : 10.1145/2997465.2997470 (cf. p. 19, 43, 70, 71, 74, 78, 87).

[34] Rene L Cruz. « A calculus for network delay. I. Network elements in isolation ». In : *IEEE Transactions on information theory* 37.1 (1991), p. 114-131 (cf. p. 37).

[35] Rene L Cruz. « A calculus for network delay. II. Network analysis ». In : *IEEE Transactions on information theory* 37.1 (1991), p. 132-141 (cf. p. 37).

[36] Rodney Cummings, Kai Richter, Rolf Ernst, Jonas Diemer et Arkadeb Ghosal. « Exploring Use of Ethernet for In-Vehicle Control Applications : AFDX, TTEthernet, EtherCAT, and AVB ». In : *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 5.1 (avr. 2012), p. 72-88. ISSN : 1946-4614. DOI : https://doi.org/10.4271/2012-01-0196. URL : https://doi.org/10.4271/2012-01-0196 (cf. p. 19, 20).

[37] Libing Deng, Guoqi Xie, Hong Liu, Yunbo Han, Renfa Li et Keqin Li. « A Survey of Real-Time Ethernet Modeling and Design Methodologies : From AVB to TSN ». In : *ACM Comput. Surv.* 55.2 (jan. 2022). ISSN : 0360-0300. DOI : 10.1145/3487330. URL : https://doi.org/10.1145/3487330 (cf. p. 20).

[38] Michael L Dertouzos. « Control robotics : The procedural control of physical processes ». In : *Proceedings IF IP Congress, 1974*. 1974 (cf. p. 54, 57).

[39] Jan Dvořák, Martin Heller et Zdeněk Hanzálek. « Makespan minimization of Time-Triggered traffic on a TTEthernet network ». In : *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE. 2017, p. 1-10. DOI : 10.1109/WFCS.2017.7991955 (cf. p. 19).

[40] R. Ernst. « System level performance analysis – the SymTA/S approach ». English. In : *IEE Proceedings - Computers and Digital Techniques* 152 (2 mars 2005), 148-166(18). ISSN : 1350-2387. URL : https://digital-library.theiet.org/content/journals/10.1049/ip-cdt_20045088 (cf. p. 39).

[41] Jonathan Falk, Frank Dürr et Kurt Rothermel. « Exploring Practical Limitations of Joint Routing and Scheduling for TSN with ILP ». In : *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2018, p. 136-146. DOI : 10.1109/RTCSA.2018.00025 (cf. p. 44, 70).

[42] Jonathan FALK, David HELLMANNS, Ben CARABELLI, Naresh NAYAK, Frank DÜRR, Stephan KEHRER et Kurt ROTHERMEL. « NeSTiNg : Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++ ». In : *Proceedings of the 2019 International Conference on Networked Systems (NetSys)*. Garching b. München, Germany, mars 2019 (cf. p. 37).

[43] Thomas A FEO et Mauricio GC RESENDE. « A probabilistic heuristic for a computationally difficult set covering problem ». In : *Operations research letters* 8.2 (1989), p. 67-71 (cf. p. 45).

[44] Richard GARREAU, Matheus LADEIRA, Emmanuel GROLLEAU, Henri BAUER, Frédéric RIDOUARD et Pascal RICHARD. « Link Between Real-Time Scheduling and Time-Triggered Networks ». In : *2023 IEEE Real-Time Systems Symposium (RTSS)*. 2023, p. 397-410. DOI : 10.1109/RTSS59052.2023.00041 (cf. p. 93).

[45] M. GERLA et L. KLEINROCK. « On the Topological Design of Distributed Computer Networks ». In : *IEEE Transactions on Communications* 25.1 (1977), p. 48-60. DOI : 10.1109/TCOM.1977.1093709 (cf. p. 15).

[46] Joël GOOSSENS et Raymond DEVILLERS. « The non-optimality of the Monotonic Assignments for Hard Real-Time Offset Free Systems ». In : *Real-Time Systems : The International Journal of Time-Critical Computing* 13.2 (1997), p. 107-126. DOI : 10.1023/A:1007980022314 (cf. p. 69).

[47] Joël GOOSSENS et Raymond DEVILLERS. « Feasibility intervals for the deadline driven scheduler with arbitrary deadlines ». In : *Proceedings of the 6th IEEE International Conference on Real-time Computing Systems and Applications*. 1999, p. 54-61. DOI : 10.1109/RTCSA.1999.811193 (cf. p. 68).

[48] Joël GOOSSENS, Emmanuel GROLLEAU et Liliana CUCU-GROSJEAN. « Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms ». In : *Real-time systems* 52 (2016), p. 808-832. DOI : 10.1007/s11241-016-9256-1 (cf. p. 75).

[49] Jérôme GRIEU. « Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques ». Thèse de doct. 2004 (cf. p. 39).

[50] Arne HAMANN, Rafik HENIA, Marek JERSAK, Razvan RACU, Kai RICHTER et Rolf ERNST. *Symta/s-symbolic timing analysis for systems*. 2004 (cf. p. 39).

[51] Jun HUANG, Mingli ZHAO, Yide ZHOU et Cong-Cong XING. « In-Vehicle Networking : Protocols, Challenges, and Solutions ». In : *IEEE Network* 33.1 (2019), p. 92-98. DOI : 10.1109/MNET.2018.1700448 (cf. p. 20).

[52] IEEE. *Time-Sensitive Networking (TSN) Task Group*. 2024. URL : https://1.ieee802.org/tsn/ (cf. p. 23).

[53] IEEE. *TSN Summary*. 2024. URL : https://www.ieee802.org/1/files/public/docs2024/admin-tsn-summary-0324-v01.pdf (cf. p. 24).

[54] « IEEE Standard for Ethernet ». In : *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)* (2022), p. 1-7025. DOI : 10.1109/IEEESTD.2022.9844436 (cf. p. 5, 17, 49).

[55] « IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25 : Enhancements for Scheduled Traffic ». In : *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)* (2016), p. 1-57. DOI : 10.1109/IEEESTD.2016.8613095 (cf. p. 31).

[56] « IEEE Standard for Local and metropolitan area networks–Audio Video Bridging (AVB) Systems ». In : *IEEE Std 802.1BA-2011* (2011), p. 1-45. DOI : 10.1109/IEEESTD.2011.6032690 (cf. p. 19, 29).

[57] « IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks ». In : *IEEE Std 802.1Q-2022 (Revision of IEEE Std 802.1Q-2018)* (2022), p. 1-2163. DOI : 10.1109/IEEESTD.2022.10004498 (cf. p. 28, 31-33, 88).

[58] « IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31 : Stream Reservation Protocol (SRP) Enhancements and Performance Improvements ». In : *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)* (2018), p. 1-208. DOI : 10.1109/IEEESTD.2018.8514112 (cf. p. 29).

[59] Jahanzaib IMTIAZ, Jürgen JASPERNEITE et Lixue HAN. « A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication ». In : *2009 IEEE Conference on Emerging Technologies & Factory Automation*. 2009, p. 1-8. DOI : 10.1109/ETFA.2009.5347126 (cf. p. 36).

[60] *OSI 11898-1:2015 : Véhicules routiers – Gestionnaire de réseau de communication (CAN) – Partie 1 : Couche liaison de données et signalisation physique*. Standard. Genève, CH : Organisation internationale de normalisation, déc. 2015 (cf. p. 18).

[61] *OSI 17987-8:2019 : Véhicules routiers – Réseau Internet local (LIN) – Partie 8 : Spécification de couche physique électrique (EPL) : LIN sur ligne d'alimentation en courant continu (DC-LIN)*. Standard. Genève, CH : Organisation internationale de normalisation, oct. 2019 (cf. p. 18).

[62] *ISO 21806-1:2020 : Véhicules routiers – Système de transport axé sur les médias – Partie 1 : Information générale et définitions*. Standard. Genève, CH : Organisation internationale de normalisation, 2020 (cf. p. 18).

[63] *ISO/IEC 7498-1:1994 : Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base : Le modèle de base*. Standard. Genève, CH : Organisation internationale de normalisation, nov. 1994 (cf. p. 15).

[64] Georges KEMAYO, Frédéric RIDOUARD, Henri BAUER et Pascal RICHARD. « A Forward end-to-end delays Analysis for packet switched networks ». In : *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*. RTNS '14. Versaille, France : Association for Computing Machinery, 2014, p. 65-74. ISBN : 9781450327275. DOI : 10.1145/2659787.2659801. URL : https://doi.org/10.1145/2659787.2659801 (cf. p. 40, 60).

[65] Nima KHALILZAD, Mohammad ASHJAEI, Luis ALMEIDA, Moris BEHNAM et Thomas NOLTE. « Towards adaptive resource reservations for component-based distributed real-time systems ». In : *SIGBED Rev.* 12.3 (août 2015), p. 24-27. DOI : 10.1145/2815482.2815486. URL : https://doi.org/10.1145/2815482.2815486 (cf. p. 36).

[66] Donald E KNUTH. *Art of computer programming, volume 2 : Seminumerical algorithms*. Addison-Wesley Professional, 2014 (cf. p. 75).

[67] Matheus LADEIRA, Emmanuel GROLLEAU, Fabien BONNEVAL, Gautier HATTENBERGER, Yassine OUHAMMOU et Yuri HÉROUARD. « Scheduling Offset-Free Systems Under FIFO Priority Protocol ». In : *34th Euromicro Conference on Real-Time Systems (ECRTS 2022)*. T. 231. 2022. DOI : 10.4230/LIPIcs.ECRTS.2022.11 (cf. p. 79).

[68] Matheus LADEIRA BOECHAT LEMOS. « Towards a Safe and Modular Architecture for Autonomous Drone Autopilots ». 2023ESMA0011. Thèse de doct. 2023. URL : http://www.theses.fr/2023ESMA0011/document (cf. p. 67).

[69] « Network Calculus ». In : *Network Calculus : A Theory of Deterministic Queuing Systems for the Internet*. Sous la dir. de Jean-Yves LE BOUDEC et Patrick THIRAN. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001, p. 3-81. ISBN : 978-3-540-45318-5. DOI : 10.1007/3-540-45318-0_1. URL : https://doi.org/10.1007/3-540-45318-0_1 (cf. p. 39).

[70] John P LEHOCZKY. « Fixed priority scheduling of periodic task sets with arbitrary deadlines ». In : *[1990] Proceedings 11th Real-Time Systems Symposium*. IEEE. 1990, p. 201-209 (cf. p. 55).

[71] Joseph Y.-T. LEUNG et M.L. MERRILL. « A note on preemptive scheduling of periodic, real-time tasks ». In : *Information Processing Letters* 11.3 (1980), p. 115-118. ISSN : 0020-0190. DOI : 10.1016/0020-0190(80)90123-4 (cf. p. 68).

[72] Joseph Y.-T. LEUNG et Jennifer WHITEHEAD. « On the complexity of fixed-priority scheduling of periodic, real-time tasks ». In : *Performance Evaluation* 2.4 (1982), p. 237-250. ISSN : 0166-5316. DOI : 10.1016/0166-5316(82)90024-4 (cf. p. 67, 68, 76).

[73] Hyung-Taek LIM, Lars VÖLKER et Daniel HERRSCHER. « Challenges in a future IP/Ethernet-based in-car network for real-time applications ». In : *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2011, p. 7-12 (cf. p. 20).

[74] Jin LV, Yongxin ZHAO, Xi WU, Yongjian LI et Qiang WANG. « Formal Analysis of TSN Scheduler for Real-Time Communications ». In : *IEEE Transactions on Reliability* 70.3 (2021), p. 1286-1294. DOI : 10.1109/TR.2020.3026689 (cf. p. 36).

[75] Dorin MAXIM et 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. Grenoble, France : Association for Computing Machinery, 2017, p. 18-27. ISBN : 9781450352864. DOI : 10.1145/3139258.3139283. URL : https://doi.org/10.1145/3139258.3139283 (cf. p. 39).

[76] M. McCabe, C. Baggerman et D. Verma. « Avionics architecture interface considerations between constellation vehicles ». In : *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*. 2009, 1.E.2-1-1.E.2-10. DOI : 10.1109/DASC.2009.5347562 (cf. p. 21).

[77] Jörn Migge, Josetxo Villanueva, Nicolas Navet et 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)*. Toulouse, France, jan. 2018. URL : https://hal.science/hal-01746132 (cf. p. 37).

[78] Junhong Min, MyoungJin Oh, Woongsoo Kim, Hyewon Seo et Jeongyeup Paek. « Evaluation of metaheuristic algorithms for TAS scheduling in time-sensitive networking ». In : *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2022, p. 809-812 (cf. p. 45).

[79] Leonardo de Moura et Nikolaj Bjørner. « Z3 : An Efficient SMT Solver ». In : *Tools and Algorithms for the Construction and Analysis of Systems*. Sous la dir. de C. R. Ramakrishnan et Jakob Rehof. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 337-340. ISBN : 978-3-540-78800-3 (cf. p. 70).

[80] *Mouser Electronics France - Distributeur de composants électroniques*. URL : https://www.mouser.fr/ (cf. p. 20).

[81] Sape Mullender. *Distributed systems*. ACM, 1990 (cf. p. 14).

[82] NASA. *Artemis I Flight Day 13 : Orion, Earth, and Moon [Distributed under the terms of : CC BY-NC-ND 2.0 DEED (https://creativecommons.org/licenses/by-nc-nd/2.0/)]*. Nov. 2022. URL : https://www.flickr.com/photos/nasaorion/52530331286/ (cf. p. 21).

[83] NASA. *Orion Reference Guide*. dernier accès au 20/04/2024 (cf. p. 20).

[84] Mitra Nasri et Gerhard Fohler. « An efficient method for assigning harmonic periods to hard real-time tasks with period ranges ». In : *2015 27th Euromicro Conference on Real-Time Systems*. IEEE. 2015, p. 149-159. DOI : 10.1109/ECRTS.2015.21 (cf. p. 79).

[85] Roman Obermaisser. *Time-triggered communication*. CRC Press, 2012 (cf. p. 20, 21, 90).

[86] Mubarak Adetunji Ojewale, Patrick Meumeu Yomsi et Borislav Nikolić. « Worst-case traversal time analysis of TSN with multi-level preemption ». In : *Journal of Systems Architecture* 116 (2021), p. 102079. ISSN : 1383-7621. DOI : https://doi.org/10.1016/j.sysarc.2021.102079. URL : https://www.sciencedirect.com/science/article/pii/S1383762121000667 (cf. p. 39).

[87] Rodolfo Pellizzoni et Giuseppe Lipari. « Feasibility analysis of real-time periodic tasks with offsets ». In : *Real-Time Systems* 30.1-2 (2005), p. 105-128. ISSN : 09226443. DOI : 10.1007/s11241-005-0506-x (cf. p. 79).

[88] Paul POP, Michael Lander RAAGAARD, Silviu S. CRACIUNAS et Wilfried STEINER. « Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks ». In : *IET Cyber-Physical Systems : Theory & Applications* 1.1 (2016), p. 86-94. DOI : https://doi.org/10.1049/iet-cps.2016.0021. eprint : https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cps.2016.0021. URL : https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cps.2016.0021 (cf. p. 45).

[89] Niklas REUSCH, Luxi ZHAO, Silviu S. CRACIUNAS et Paul POP. « Window-Based Schedule Synthesis for Industrial IEEE 802.1Qbv TSN Networks ». In : *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*. 2020, p. 1-4. DOI : 10.1109/WFCS47810.2020.9114414 (cf. p. 44).

[90] K. RICHTER, M. JERSAK et R. ERNST. « A formal approach to MpSoC performance verification ». In : *Computer* 36.4 (2003), p. 60-67. DOI : 10.1109/MC.2003.1193230 (cf. p. 39).

[91] K. RICHTER, R. RACU et R. ERNST. « Scheduling analysis integration for heterogeneous multiprocessor SoC ». In : *RTSS 2003. 24th IEEE Real-Time Systems Symposium, 2003*. 2003, p. 236-245. DOI : 10.1109/REAL.2003.1253270 (cf. p. 39).

[92] Ismael RIPOLL et Rafael BALLESTER-RIPOLL. « Period selection for minimal hyperperiod in periodic task systems ». In : *IEEE Transactions on Computers* 62.9 (2012), p. 1813-1822. DOI : 10.1109/TC.2012.243 (cf. p. 79).

[93] Jonas ROX et Rolf ERNST. *Formal timing analysis of full duplex switched based ethernet network architectures*. Rapp. tech. SAE Technical Paper, 2010 (cf. p. 39).

[94] Aellison Cassimiro T. dos SANTOS, Ben SCHNEIDER et Vivek NIGAM. « TSNSCHED : Automated Schedule Generation for Time Sensitive Networking ». In : *2019 Formal Methods in Computer Aided Design (FMCAD)*. 2019, p. 69-77. DOI : 10.23919/FMCAD.2019.8894249 (cf. p. 44).

[95] Eike SCHWEISSGUTH, Peter DANIELIS, Dirk TIMMERMANN, Helge PARZYJEGLA et Gero MÜHL. « ILP-based joint routing and scheduling for time-triggered networks ». In : *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS '17. Grenoble, France : Association for Computing Machinery, 2017, p. 8-17. ISBN : 9781450352864. DOI : 10.1145/3139258.3139289. URL : https://doi.org/10.1145/3139258.3139289 (cf. p. 44, 70).

[96] Ramon SERNA OLIVER, Silviu S. CRACIUNAS et Wilfried STEINER. « IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding ». In : *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2018, p. 13-24. DOI : 10.1109/RTAS.2018.00008 (cf. p. 44, 70).

[97] Luis SILVA, Paulo PEDREIRAS, Pedro FONSECA et Luis ALMEIDA. « On the adequacy of SDN and TSN for Industry 4.0 ». In : *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. 2019, p. 43-51. DOI : 10.1109/ISORC.2019.00017 (cf. p. 20).

[98] Anne Elizabeth SPALDING. *Min-plus algebra and graph domination*. University of Colorado at Denver, 1998 (cf. p. 37).

[99] J.A. STANKOVIC. « Misconceptions about real-time computing : a serious problem for next-generation systems ». In : *Computer* 21.10 (1988), p. 10-19. DOI : 10.1109/2.7053 (cf. p. 13).

[100] Wilfried STEINER. « An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks ». In : *2010 31st IEEE Real-Time Systems Symposium*. 2010, p. 375-384. DOI : 10.1109/RTSS.2010.25 (cf. p. 19, 43, 68, 70, 71).

[101] Domitian TAMAS-SELICEAN, Paul POP et Wilfried STEINER. « Synthesis of communication schedules for TTEthernet-based mixed-criticality systems ». In : *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. 2012, p. 473-482. DOI : {10.1145/2380445.2380518} (cf. p. 19).

[102] Domiţian TĂMAŞ-SELICEAN et Paul POP. « Optimization of TTEthernet networks to support best-effort traffic ». In : *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE. 2014, p. 1-4. DOI : 10.1109/ETFA.2014.7005256 (cf. p. 19).

[103] Daniel THIELE. « Formal Timing Analysis of Ethernet TSN Extensions for Automotive Applications ». Thèse de doct. Technische Universität Carolo-Wilhelmina zu Braunschweig, 2022 (cf. p. 39).

[104] Daniel THIELE et 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)*. 2016, p. 1-9. DOI : 10.1109/ETFA.2016.7733740 (cf. p. 39).

[105] Daniel THIELE, Rolf ERNST et Jonas DIEMER. « Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers ». In : *2015 IEEE Vehicular Networking Conference (VNC)*. 2015, p. 251-258. DOI : 10.1109/VNC.2015.7385584 (cf. p. 39).

[106] *Time-Sensitive Networking (TSN) Task Group*. URL : https://1.ieee802.org/tsn/ (cf. p. 5, 19).

[107] Martin WOLLSCHLAEGER, Thilo SAUTER et Juergen JASPERNEITE. « The Future of Industrial Communication : Automation Networks in the Era of the Internet of Things and Industry 4.0 ». In : *IEEE Industrial Electronics Magazine* 11.1 (2017), p. 17-27. DOI : 10.1109/MIE.2017.2649104 (cf. p. 19, 20).

[108] Jia XU. « A method for adjusting the periods of periodic processes to reduce the least common multiple of the period lengths in real-time embedded systems ». In : *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*. IEEE. 2010, p. 288-294. DOI : 10.1109/MESA.2010.5552058 (cf. p. 79).

[109] Q. XU et X. YANG. « Analysis of forward approach for upper bounding end-to-end transmission delays over distributed real-time avionics networks ». In : *The Aeronautical Journal* 124.1279 (2020), p. 1399-1435. DOI : 10.1017/aer.2020.33 (cf. p. 41).

[110] Qingfei XU et Xinyu YANG. « Performance Evaluation on Packet Transmission for Distributed Real-time Avionics Networks Using Forward End-to-End Delay Analysis ». In : *TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES* 64.1 (2021), p. 1-12. DOI : 10.2322/tjsass.64.1 (cf. p. 41).

[111] Jiayi ZHANG, Lihao CHEN, Tongtong WANG et Xinyuan WANG. « Analysis of TSN for Industrial Automation based on Network Calculus ». In : *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, p. 240-247. DOI : 10.1109/ETFA.2019.8869053 (cf. p. 39).

[112] Luxi ZHAO, Paul POP et Silviu S. CRACIUNAS. « Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus ». In : *IEEE Access* 6 (2018), p. 41803-41815. DOI : 10.1109/ACCESS.2018.2858767 (cf. p. 39).

[113] Luxi ZHAO, Paul POP, Zhong ZHENG, Hugo DAIGMORTE et Marc BOYER. « Latency Analysis of Multiple Classes of AVB Traffic in TSN With Standard Credit Behavior Using Network Calculus ». In : *IEEE Transactions on Industrial Electronics* 68.10 (2021), p. 10291-10302. DOI : 10.1109/TIE.2020.3021638 (cf. p. 39, 90, 91).

[114] Luxi ZHAO, Paul POP, Zhong ZHENG et Qiao LI. « Timing Analysis of AVB Traffic in TSN Networks Using Network Calculus ». In : *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2018, p. 25-36. DOI : 10.1109/RTAS.2018.00009 (cf. p. 39).

[115] Zifan ZHOU, Juho LEE, Michael Stübert BERGER, Sungkwon PARK et Ying YAN. « Simulating TSN traffic scheduling and shaping for future automotive Ethernet ». In : *Journal of Communications and Networks* 23.1 (2021), p. 53-62 (cf. p. 37).

RÉSUMÉ

Ce manuscrit se penche sur deux problématiques des réseaux temps-réel embarqués. La première est le dimensionnement des buffers de commutateurs en nombre de trames. Une nouvelle méthode nommée *Buffer Dimentionning per Frame* (BDF), est proposée pour calculer une borne supérieure garantie sur le dimensionnement. Cette méthode permet d'éviter le sous-dimensionnement des buffers, contrairement aux approches existantes. La deuxième problématique concerne l'ordonnancement des trames dans les réseaux à *offset-free*. L'hypothèse courante selon laquelle l'intervalle [0, H) est toujours cyclique est démontrée incorrecte en l'absence de deux contraintes. Une condition suffisante pour garantir cette cyclicité est prouvée, basée sur la conjonction de la contrainte de non-congestion et de la contrainte sur trames. De plus, une nouvelle heuristique, GCD#, est proposée pour générer des ordonnancements réalisables même en présence de congestion, offrant ainsi une alternative aux méthodes existantes limitées par la contrainte de non-congestion.

ABSTRACT

This document addresses two challenges in embedded real-time networks. The first one is the dimentionning of switch buffers in terms of frames. A new method, *Buffer Dimentionning per Frame* (BDF), is proposed to compute a guaranteed upper bound on the dimensioning. This method helps avoid under-dimensionned buffers, unlike existing approaches. The second challenge concerns the scheduling of frames in *offset-free* networks. The commonly held assumption that the interval [0, H) is always cyclic is proven incorrect without two constraints. A sufficient condition to guarantee this cyclicity is established, based on the conjunction of the non-congestion constraint and the constraint on frames. Furthermore, a novel heuristic, GCD#, is introduced to generate feasible schedules even in the presence of congestion, providing an alternative to existing methods limited by the non-congestion constraint.