Analyse de Flux de Trames AFDX en Réception et Méthode d'Optimisation Mémoire

Yohan Baga

Thèse de Doctorat

En vue de l'obtention du grade de

Docteur de l'Université de Cergy-Pontoise

Analyse de Flux de Trames AFDX en Réception et Méthode d'Optimisation Mémoire

---

Présentée et soutenue par

Yohan Baga

Ecole doctorale : Sciences et Ingénierie (SI)

Unité de recherche : Équipes Traitement de l'Information et Systèmes (ETIS), équipe ASTRE
Partenaire industriel : Zodiac Aero Electric (ZEL), Cockpit & Lighting Systems (ZCLS), R&T

Soutenance le -- Mars 2018

Devant le jury composé de

Dr. Claire Pagetti	Ingénieure de recherche à l'IRIT	Rapporteuse
Pr. Jean-Luc Scharbarg	Professeur des universités à l'INPT/ENSEEIHT	Rapporteur
Dr. Claire Maiza	Maître de conférence à Grenoble INP	Examinatrice
Dr. Catherine Dezan	Maître de conférence à l'UBO	Examinatrice
Pr. David Declercq	Professeur des universités à l'ENSEA	Examinateur
Pr. Raoul Velazco	Directeur de recherche à l'UGA	Président du jury
Pr. Oliver Romain	Professeur des universités à l'UCP	Directeur de thèse
Dr. Fakhreddine Ghaffari	Maître de conférences à l'UCP	Encadrant de thèse
M. Michael Nahmiyace	Manager Service R&T à Zodiac Aerospace	Invité

Je dédie ce travail à mes parents, Thierry et Evelyne Baga, pour leur soutien et leur amour.

Table des matières

III Étude probabiliste de l'occurrence de SBFs dans le flux entrant 91

Liste des figures

Liste des tableaux

Glossaire

AEEC : Avionic Electronic Engineering Committee
AFDX : Avionic Full-DupleX switched Ethernet
AMP : Asymmetric Multi Processing
APEX : Application / EXecutive
API : Application Programming Interface
ARINC : Aeronautical Radio, INCorporated
BAG : Bandwidth Allocation Gap
CAAC : Civil Aviation Administration of China
CAN : Controller Area Network
CCM : Cache Coherency Module
COTS : Component Off The Shelf
CPU : Central Processing Unit
CRC : Cyclic Redundancy Code
CTRES : Configuration Table of the Reception ES
DAL : Design Assurance Level
DO : DOcument
DS : Data Set
EASA : European Aviation Safety Agency
ED : Eurocae Documents
ES : End-System
ESA : Agence Spatiale Européenne
EUROCAE : EURopean Organisation for Civil Aviation Equipment
FAA : Federal Aviation Administration
FDS : Functional Data Set
FIFO : First In First Out
FPGA : Field Programmable Gate Array
FSS : Functional Status Set
IC : Integrity Checking
ICAO : International Civil Aviation Organization
IEEE : Institute of Electrical and Electronics Engineers
IFG : Inter Frame Gap
IHL : Internet Header Length
IMA : Integrated Modular Avionics

IO : Input/Output
IP : Internet Protocol
IP : Intellectual Property
OEM : Original Equipment Manufacturer
OS : Operating System
OSI : Open Systems Interconnection
PP : Periodical Pattern
PPCM : Plus Petit Commun Multiple
QoS : Quality of Service
LAN : Local Area Network
LPS : Longest Periodical Sequence
LRU : Line Replaceable Unit
LSBF : Longest Sequence of Back-to-back Frames
LZW : Lempel Ziv Welch
MAC : Media Access Control
MMU : Memory Management Unit
MSP : Multi Softcore Processors
RM : Redundancy Manager
RTCA, Inc : Radio Technical Commission for Aeronautics
RTOS : Real Time Operating System
SAP : Service Access Points
SBF : Sequence of Back-to-back Frames
SD : Standard Deviation
SDF : Start Frame Delimiter
SN : Sequence Number
SMP : Symmetric Multi Processing
SoC : System on-Chip
SWaP : Size, Weight and Power
TDMA : Time Division Multiple Access
TTL : Time To Live
UDP : User Data Protocol
VL : Virtual Link
WCETs : Worst Case Execution Time
WFB : Worst Frame Backlog

Introduction générale

Proposer un avion « tout électrique » est une motivation forte portée par les principaux acteurs de l'industrie aéronautique. Elle consiste à remplacer progressivement les chaînes de puissance hydrauliques, pneumatiques et mécaniques par des systèmes entièrement électriques, mais aussi à faire face à l'émergence de nouveaux besoins fonctionnels tels qu'un service wifi cabine ou un système de taxiage électrique. Cela induit une augmentation significative du nombre de systèmes électroniques embarqués à bord, et ainsi l'apparition de problématiques liées à la fiabilité et aux performances opérationnelles de l'ensemble de ces systèmes.

Dans cette perspective, l'émergence des architectures multicœurs séduit les équipementiers qui cherchent à bénéficier des avancées technologiques tirées par le marché grand public pour concevoir des architectures matérielles à moindre coût et à haut niveau de performance. En revanche, la dualité, d'une part, des fortes contraintes de l'industrie aéronautique (déterminisme, fiabilité, etc…) et d'autre part, de la rigidité et des contraintes de protection intellectuelle des architectures multicœurs, rend difficile l'emploi de ces architectures en aéronautique. Pour permettre leur utilisation en milieu critique, des techniques architecturales sont mises en œuvre comme le partitionnement robuste ou l'IMA. Cependant, les ressources partagées entre les différents cœurs constituent une faille pouvant remettre en cause le partitionnement robuste et limitant la constitution d'IMA. Ces ressources partagées sont les mémoires caches pour la sauvegarde de données d'exécution et les **IOs (Input/Outputs)** pour les échanges de données entre les partitions logicielles hébergées sur la plateforme et les autres équipements de bord. Dans cette étude, nous nous concentrons sur l'IO AFDX, appelée **ES (End-System)**.

D'un bout à l'autre de la chaîne fonctionnelle, des équipements aéronautiques de nature différente échangent des informations sous la forme de messages numériques (trames) transitant sur des architectures de communication telles que des bus ou des réseaux. En réponse à l'évolution des besoins en matière de fiabilité et de bande passante, Airbus a proposé le réseau **AFDX (Avionic Full-DupleX switched Ethernet)**, standardisé par la suite sous le nom ARINC 664 (1). Le réseau AFDX offre aux équipements connectés un service de communication fiable, temps-réel et déterministe. Ce réseau est organisé autour de commutateurs et de terminaux d'émission et de réception de trames, les ESs. Les ESs comprennent des couches de traitement pour permettre l'émission et la réception de trames simultanées suivant un protocole de communication spécifique à l'AFDX, lui-même basé sur le protocole Ethernet. Le rôle premier du protocole AFDX est d'assurer la robustesse du réseau face aux fautes et aux cas de pannes, mais aussi le déterminisme temporel pour le temps de transfert des trames d'un ES du réseau à l'autre. En réception, les délais de traitement des différentes couches de l'ES imposent le stockage des trames reçues afin d'éviter la perte ou la corruption de trames interdite par le standard ARINC 664.

L'objectif de cette thèse est de caractériser le backlog de la mémoire de réception pour n'importe quelle configuration, et ainsi d'optimiser l'espace mémoire nécessaire au stockage des trames AFDX lors de leur réception.

Position du problème : Dimensionnement de la mémoire de réception

Dans cette thèse, nous traitons le problème du dimensionnement mémoire en milieu critique. Plus spécifiquement, il s'agit de dimensionner de façon optimale la mémoire de l'ES de réception en prenant en compte les différents paramètres du réseau AFDX qui influencent la réception des trames. Par « dimensionner de façon optimale », nous entendons déterminer la taille mémoire la plus réduite pour une table de configuration d'un ES de réception (en anglais : **CTRES (Configuration Table of the Reception End-System)**), tout en garantissant la non-perte et l'intégrité des trames reçues.

Commençons par justifier la nécessité de stocker les trames reçues au niveau de l'ES de réception. En réception, un ES extrait les données contenues dans les trames AFDX et fournit ces données aux partitions logicielles hébergées sur l'architecture matérielle. Au sein d'un ES de réception de conception mixte, des composants logiciels et des composants matériels traitent les trames au fur et à mesure de leur arrivée. Les composants matériels traitent les trames au fil de l'eau et ils ne ralentissent pas le flux entrant de façon significative. Il n'en est pas de même pour les composants logiciels qui peuvent causer d'importants retards. Il en résulte le placement d'une mémoire de réception à l'interface des composants matériels et des composants logiciels. Cette mémoire de type FIFO est nommée dans cette étude la *mémoire de réception*.

À cela s'ajoute l'absence de synchronisation globale du réseau AFDX et la variabilité des flux de trames causée par les latences technologiques du réseau. De plus, la configuration générale du réseau implique qu'un ES de réception reçoit généralement des trames de plusieurs ESs sources ce qui peut conduire à un important trafic en réception. Par conséquent, les trames sont fréquemment en concurrence pour accéder aux ressources physiques du réseau comme les ports de sortie des commutateurs. Cela peut entraîner une congestion de trames à l'intérieur des ports de sortie et résulter en une séquence de trames accolées (en anglais : **SBF (Sequence of Back-to-back Frames)**) envoyées sur le lien physique entre le commutateur et l'ES de réception. Ces trames s'accumulent dans la mémoire de réception et forment ce que l'on appelle un *backlog*. L'estimation de la taille de ce backlog est directement liée à la taille optimale de la mémoire de réception. Parvenir à construire le pire backlog possible (en anglais : **WFB (Worst Frame Backlog)**) compte tenu d'une CTRES donnée revient à déterminer la taille optimale de la mémoire de réception.

Le dimensionnement de la mémoire de réception a reçu peu d'attention dans la littérature, le Network Calculus fournissant une estimation pessimiste de celle-ci. Cette solution communément utilisée dans l'industrie surdimensionne la mémoire allouée à la réception des trames. De la ressource mémoire est ainsi gaspillée, considérant les centaines d'ESs constituant le réseau. Or, le fait de concevoir un ES sur une plateforme FPGA pose de fait une limitation de la mémoire sur puce et le

dimensionnement de la mémoire devient une question critique à la fois pour éviter une perte de trame et pour ne pas surestimer les ressources mémoires nécessaires.

Principales contributions

Notre travail traite de l'optimisation de la taille de la mémoire de réception dans un contexte de mémoires statiques (FIFOs) qui affectent à la fois la fiabilité du réseau et les délais de transmission.

Dans un premier temps, nous proposons une approche basée sur un modèle pessimiste pour calculer la taille optimale de la mémoire de réception. Le modèle pessimiste s'appuie sur des hypothèses et des heuristiques visant à conditionner la forme du flux de trames entrant dans la mémoire. Ainsi, nous supposons que le pire scénario de réception pour le flux entrant consiste en la réception de la plus longue séquence de trames accolées (en anglais : **LSBF (Longest Sequence of Back-to-back Frames)**). Pour une CTRES donnée, nous proposons alors une méthode pour construire le LSBF en ajoutant des retards dans un flux périodique, puis un modèle d'écriture et de lecture de trames dans la mémoire. Nous réalisons ensuite un simulateur codé en C mettant en œuvre la méthode de construction pessimiste. Ce simulateur estime le WFB à partir de la CTRES pour une vitesse de lecture constante. Il est alors aisé de déduire la taille de la mémoire de réception.

Ensuite, nous proposons une amélioration du modèle précédent en se basant non pas sur un flux périodique mais sur les intervalles de réception des trames prises individuellement. Nous appelons ce modèle : le modèle *par intervalles*. Le modèle par intervalles implique une nouvelle caractérisation du flux entrant en considérant la variation du délai de bout en bout de chaque **VL (Virtual Link)**. Un nouveau simulateur est créé en C sur la base de cette méthode de construction du LSBF plus réaliste, et les résultats conduisent à une réduction de la taille de la mémoire. De plus, d'autres résultats mettent en évidence l'influence des paramètres CTRES sur le LSBF (et donc sur le WFB) en proposant des métriques pour classer les paramètres CTRES.

Le but de ces deux méthodes de construction est de déterminer le WFB pour une CTRES donnée. Cependant, à ce stade, nous ne disposons d'aucune mesure du nombre d'occurrences des LSBFs calculés, et nous ne savons pas si ces LSBFs construits par nos méthodes se produisent fréquemment ou non. C'est l'objet de la troisième contribution de cette thèse. En effet, nous nous concentrons sur les SBFs et nous évaluons leurs occurrences avec une méthode probabiliste basée à la fois sur le modèle par intervalles et sur des distributions de probabilité. À nouveau, un simulateur en C est produit pour réaliser des simulations. Enfin, nous comparons les résultats de simulation de la méthode probabiliste avec les résultats de la méthode de construction pessimiste afin d'estimer la pertinence de la prise en compte des LSBFs pour l'estimation de la taille mémoire.

L'ultime contribution de ce travail est l'utilisation d'un code de compression pour réduire l'espace mémoire nécessaire au stockage des trames. La compression est un domaine transverse : elle est largement employée en communication mais aussi pour le stockage de fichiers volumineux. Cela dit, aucune technique de compression n'est implémentée sur le réseau AFDX, probablement pour des questions de fiabilité. Pour répondre aux exigences de certification aéronautique, les codes de

compression sans perte à dictionnaires constituent des candidats prometteurs. En particulier, une adaptation du code **LZW (Lempel Ziv Welch)** est codée en VHDL (couple encodeur/décodeur) , puis elle est implémentée sur une plateforme matérielle de test créée pour les besoins de mesure. Des mesures de gain sont réalisées pour plusieurs jeux de trames AFDX dont nous avons modélisé la génération et mis au point un générateur de trames codé en C. Les plus forts gains mesurés s'élèvent à plus de 20% ce qui en fait une méthode prometteuse que nous songeons à étendre aux communications AFDX.

Organisation du mémoire

À la suite de l'introduction générale, le mémoire est organisé en quatre grandes parties dans l'ordre des contributions : l'état de l'art, les méthodes de construction du LSBF, la méthode probabiliste de recherche de SBFs et la compression sans perte.

L'état de l'art suit le déroulement des travaux jusqu'à la définition de la problématique de recherche. Il présente le contexte industriel et les travaux académiques pour poser le contexte scientifique dans lequel nous allons apporter notre contribution.

Les méthodes de construction du LSBF sont scindées en deux chapitres et proposent donc deux méthodes pour caractériser le flux de trames entrant à partir d'une CTRES quelconque, basées sur des heuristiques et des modèles du flux entrant.

La modélisation probabiliste apporte un éclairage supplémentaire sur les probabilités d'occurrence des LSBFs établis dans la partie précédente.

La compression sans perte propose l'implémentation d'un code de compression pour réduire l'espace de stockage des trames dans la mémoire.

Au terme de ces quatre parties, la conclusion générale résume les principaux aspects de l'étude.

Dans un souci de continuité de ces travaux, une ultime partie de perspectives présente trois pistes intéressantes. Quelques résultats préliminaires sont donnés en guise d'illustration.

Première partie

État de l’art

Chapitre 1 : État de l'art

L'objectif de cet état de l'art est de fournir une vision transverse sur les dernières avancées industrielles et académiques, tout en reprenant la chronologie des explorations qui ont conduit à la réalisation de ce travail.

Le caractère industriel de cette thèse l'ancre profondément dans un contexte applicatif contraint. C'est pourquoi la Section 1.1 propose d'introduire les aspects spécifiques du développement d'électronique embarquée en aéronautique. Le rôle des instances internationales de certification et de standardisation est détaillé, puis nous nous concentrons sur le développement des architectures matérielles en aéronautique, et en particulier, sur l'avènement des architectures multicœurs. Ensuite, nous présentons des méthodes de conception pour les architectures matérielles et le principe de l'isolation fonctionnelle dont le but principal est de réduire les coûts de certification. Dans le cadre d'architectures multicœurs, les ressources partagées sont des points de rupture potentiels du partitionnement de l'architecture et c'est pourquoi notre étude se recentre sur les ressources partagées, et notamment l'IO AFDX (ou End-System).

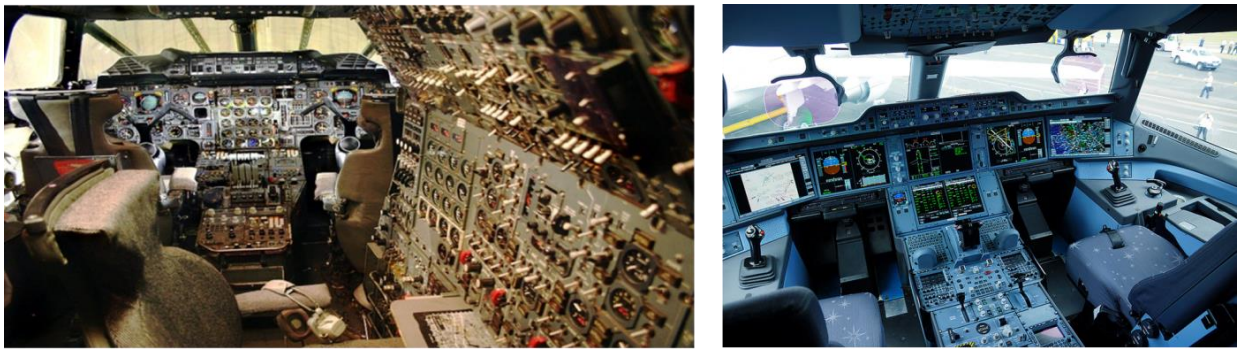
Pour bien appréhender cette IO complexe, la Section 1.2 s'attache à présenter le réseau AFDX dans son ensemble. L'infrastructure du réseau et le protocole AFDX sont décrits en détail ainsi que le fonctionnement des commutateurs AFDX. Les principaux axes de recherche portant sur le réseau AFDX sont exposés. Enfin, la Section 1.3 se focalise sur l'ES à la fois sur la partie transmission et sur la partie réception. Enfin, l'attention se concentre sur la question du dimensionnement de la mémoire de réception ES et nous positionnons notre étude par rapport à l'état de l'art.

1.1 Contexte Industriel de la thèse

1.1.1 Vers l'avion tout électrique

À l'heure de la rédaction de ce rapport, la stratégie commerciale de l'industrie aéronautique converge vers l'avion tout électrique comme l'attestent les communiqués de presse des avionneurs comme Airbus, Boeing, Dassault ou Irkut Corporation, mais également ceux des équipementiers aéronautiques (en anglais : **OEM (Original Equipment Manufacturers)**) comme Safran, Thales, Rockwell Collins ou Leonardo. Ce courant industriel répond à des contraintes principalement d'ordre économique. En effet, les avantages sont multiples : réduction du volume des équipements embarqués, amélioration de la fiabilité, diminution du poids à vide de l'appareil et réduction de la consommation globale d'énergie.

L'utilisation de systèmes électroniques à bord des aéronefs permet d'intégrer des fonctions avioniques parfois hétérogènes sur un même support. Cela permet de gagner en ergonomie et de simplifier les processus de maintenance afin de limiter l'immobilisation de l'appareil au sol. En guise d'illustration, la Figure 1 présente deux photographies des cockpits de deux appareils qui ont marqué leur époque : le Concorde (a) et l'Airbus A350 (b). En trois décennies, l'intégration des fonctions avioniques a beaucoup progressé comme l'atteste l'épuration des panneaux de contrôle. Un seul écran affiche désormais plusieurs données qui nécessitaient auparavant plusieurs jauges, où une seule commande multidirectionnelle remplace à présent plusieurs boutons poussoirs ou des interrupteurs à bascule.



(a) (b)

Figure 1 – Photographies du cockpit du Concorde (premier vol en 1979) (a) et de l'Airbus A350 (premier vol en 2013) (b).

Les conséquences économiques d'une telle évolution se traduisent par la possibilité d'embarquer davantage de passagers ou de fret tout en réalisant une économie de carburant et en améliorant la sécurité des systèmes. Le prix d'exploitation de l'appareil s'en retrouve diminué, et cela est un argument de vente clef auprès des compagnies aériennes.

1.1.2 Normes et standards aéronautiques

La mise sur le marché d'équipements aéronautiques est rigoureusement encadrée par des agences internationales qui veillent à la conformité des équipements vis-à-vis de normes de sécurité. Ces agences sont principalement :

- L'**ICAO (International Civil Aviation Organization)** au niveau des Nations Unies.
- L'**EASA (European Aviation Safety Agency)** au niveau européen.
- La **FAA (Federal Aviation Administration)** au niveau américain.
- L'agence **Rosaviatsia** pour la Russie.
- La **CAAC (Civil Aviation Administration of China)** pour la Chine.

Ces agences procèdent à la certification des équipements et des appareils, et délivrent pour cela des agréments aux entreprises qui réalisent la conception, la fabrication et la maintenance d'équipements aéronautiques. Un agrément est la reconnaissance officielle attestant de la conformité de l'équipement produit vis-à-vis des règles de sécurité.

En parallèle, il existe des associations qui fédèrent l'ensemble des acteurs du domaine de l'aviation civile afin d'établir des standards portant sur l'ensemble des systèmes utilisés à bord des avions. Les associations les plus connues sont : l'**EUROCAE (EURopean Organisation for Civil Aviation Equipment)** au niveau européen dont les documents portent l'en-tête « **ED » (EUROCAE Document)**, et la **RTCA (Radio Technical Commission for Aeronautics)** son équivalent américain dont les documents portent l'en-tête « **DO » (DOcument)**. Ces associations regroupent des constructeurs aéronautiques, des fournisseurs de service, des autorités nationales et internationales de l'aviation civile, des compagnies aériennes et des opérateurs.

Concernant plus spécifiquement le domaine des communications, l'**ARINC (Aeronautical Radio, INCorporated)**, société américaine détenue à 100% par Rockwell Collins depuis 2013, édite les principaux standards pour la communication à bord des aéronefs et pour la communication entre les aéronefs et le sol. Ces standards sont édités par un comité d'experts regroupés au sein de l'**AEEC (Airlines Electronic Engineering Committee)**. Ils portent sur les infrastructures de communication que sont les bus et les réseaux embarqués, et sur les protocoles de communication. Notons que ces standards ont valeur de recommandation et il ne s'agit pas de normes aéronautiques à proprement parler. Ainsi, un industriel peut choisir de ne pas s'y conformer et tout de même obtenir un agrément pour son équipement. Toutefois, il peut arriver qu'un standard soit si bien implanté dans son volet technologique qu'il prend quasiment valeur de norme comme c'est le cas pour certains ARINCs.

Le Tableau 1 présente les différentes séries de standard ARINC, chaque série étant relative à un type d'équipement particulier. L'ARINC 429 (2) dont la première version a été éditée en 1966, est de nos jours le bus de communication le plus utilisé à bord des aéronefs et il équipe une large part des avions Airbus et Boeing. Cependant, de par sa conception limitant son évolutivité (en particulier, une seule source possible sur un lien physique), l'ARINC 429 n'est pas en mesure de répondre à l'évolution des besoins en termes de connectivité et de bande passante.

Séries	Description
Série **400**	Fournit les recommandations pour l'installation, le câblage, les bus de données et les bases de données. ex: **ARINC 429** définit les caractéristiques électriques et les formats de données d'un bus série qui peut comporter une source et jusqu'à 20 receveurs. Ce bus est capable de fonctionner à un débit maximal de 100 kbit/s.
Série **500**	Décrit des équipements d'avionique analogiques généralement devenus obsolètes.
Série **600**	Contient des recommandations relatives à la conception et à la communication pour les équipements d'avionique numériques. ex: **ARINC 653** définit un système d'exploitation temps réel (RTOS) pour le partitionnement spatial et temporel des ressources de calcul. Le standard définit également des **APIs (Application Programming Interfaces)** pour placer une couche d'abstraction entre les ressources software et hardware et les applications hébergées. **ARINC 664** définit l'utilisation d'un réseau Ethernet déterministe en tant que bus de communication avionique utilisé à bord des avions récents tels l'Airbus A380, le Boeing 787 Dreamliner et le Sukhoi Superjet 100.
Série **700**	Décrit les équipements et les systèmes numériques dont les interfaces mécaniques, électriques et logiques des **LRUs (Line Replaceable Units)**.
Série **800**	Décrit des bus de données à haute vitesse basés sur l'utilisation de la fibre optique. ex: **ARINC 825** définit le protocole **CAN (Controller Area Network)** pour un usage aéronautique.
Série **900**	Définit les systèmes d'avionique modulaire, leurs fonctions et interfaces.

Tableau 1 – Description des séries de standards ARINC.

Pour pallier à ces besoins, l'ARINC 664 (3) propose l'utilisation de réseaux déterministes pour assurer la communication entre les équipements de bord. En particulier, la partie 7 de l'ARINC 664 définit l'architecture et le protocole AFDX sur lesquels portent cette thèse. Nous y revenons dans la section suivante.

1.1.3 Contexte de développement des architectures matérielles, architectures multicœurs

Historiquement, les marchés de l'aéronautique et de la Défense ont exercé une influence notable sur les développements technologiques, notamment sur le développement des composants électroniques. Toutefois, cette influence s'est atténuée au profit de celle exercée par le marché grand public dont les cycles d'innovation rapprochés ont permis la conception de composants à haut niveau d'intégration, à bas coûts et à basse consommation d'énergie.

1.1.3.1 Processeurs COTS

Parmi ces composants à hautes performances, les processeurs **COTS (Component Off The Shelf)** monocœur proposés par des sociétés comme Freescale ou Texas Instruments (4), ont séduit les équipementiers aéronautiques. Ces **CPUs (Central Processing Unit)** sont conçus pour des applications générales et ils utilisent des bus comme supports de communication internes. Ils peuvent également présenter un ou plusieurs éléments dédiés à la gestion de périphériques de communication appelés **IOs (Input/Output)**.

Depuis le succès commercial de l'architecture à deux cœurs Core 2 Duo® proposée par Intel en 2006, les fondeurs de processeurs ont pris un virage technologique et ils se sont lancés sur la voie des processeurs multicœurs en partie pour compenser le ralentissement de l'évolution des procédés de gravure (5). Depuis quelques années, l'évolution du marché est telle que les processeurs COTS monocœur sont progressivement remplacés par des processeurs COTS basés sur des **SoCs (System on Chip)** multicœurs.

Dans l'industrie aéronautique, les processeurs COTS multicœurs posent de nouveaux défis de certification notamment pour les systèmes critiques où l'obtention des certifications RTCA DO-178C et EUROCAE ED-12C sont nécessaires. Certifier un processeur COTS représente un coût important pour les équipementiers, d'autant plus lorsque l'accès au code source de l'architecture est protégé par le fournisseur. L'avènement des processeurs COTS multicœurs a considérablement alourdi les coûts de certification (plusieurs millions de dollars pour le dossier de certification de l'architecture matérielle) du fait de la complexité croissante des architectures.

Enfin, alors que les processeurs COTS multicœurs connaissent une forte croissance sur le marché grand public et proposent des performances toujours plus élevées, les fournisseurs d'équipements aéronautiques renoncent parfois à des optimisations et peuvent aller jusqu'à désactiver des cœurs pour exploiter le processeur multicœurs comme un processeur monocœur. Ceci permet de réduire le coût de certification dans une moindre mesure en simplifiant le fonctionnement du processeur, et ce dans le but de tirer le meilleur parti du compromis coût de certification/performance.

1.1.3.2 Technologie FPGA

Mis au point dans les années 1980, le **FPGA (Field Programmable Gate Array)** a connu un certain succès dans l'industrie des applications électroniques. Au cours des dix dernières années, les progrès de la technologie FPGA ont fourni des composants flexibles pour créer et pour débugger des architectures complexes sur des cycles courts. Ces architectures incluent des processeurs hardcore (par exemple : ARM 9) ou softcore (par exemple : MicroBlaze® ou NiOS II), de la mémoire sur puce, des multiplicateurs à virgule flottante et divers IOs spécifiques.

Afin de promouvoir la flexibilité, les architectures matérielles mises en œuvre sur FPGA ont amélioré les principes de conception sur-mesure et de réutilisation des composants appelées **IPs (Intellectual Property)**. De nos jours, les fournisseurs de FPGAs offrent de larges bibliothèques d'IPs pour répondre aux besoins courants de la conception d'architectures. Généralement, ces IPs peuvent être portées sur différents FPGAs, ce qui résout dans la foulée les problèmes d'obsolescence rencontrés par les processeurs COTS. En revanche, le coût unitaire d'un FPGA (jusqu'à cent fois plus cher qu'un processeur COTS) reste un inconvénient majeur à l'utilisation des FPGAs en tant que supports pour une architecture complexe.

Comme de nombreux systèmes sont le résultat d'un assemblage de composants logiciels et matériels, le concept de processeur softcore a gagné en popularité pour développer des architectures complexes en tant que solution flexible et indépendante du support matériel. LEON® est un processeur

RISC 128 bits configurable développé pour les besoins de l'**ESA (Agence Spatiale Européenne)**, à ce jour décliné en quatre versions. La dernière version LEON4 (6) atteint une efficacité de performance maximale de 1,7 DMIPS/MHz. D'autres processeurs softcores bien connus comme le MicroBlaze de Xilinx (7) et le NiOS II d'Altera (8) ont des fonctionnalités et des performances comparables.

En outre, plusieurs processeurs softcores peuvent être implémentés sur la même puce FPGA. Ceci permet la conception d'architectures dites **MSPs (Multi Softcore Processors)** conçues dans le but de satisfaire des applications logicielles ayant un fort besoin en puissance de calcul. Le nombre de processeurs softcores implémentés sur un composant FPGA n'est limité que par les ressources logiques disponibles. De grandes architectures multicœurs peuvent être conçues puisque les dernières générations de FPGA sont constituées de millions d'éléments logiques, de registres et de milliers de blocs de mémoire (9). Enfin, les améliorations portées aux mécanismes de protection matériels contre les fautes transitoires garantissent une utilisation en environnement hostile (10). Pour toutes ces raisons, les FPGAs sont progressivement exploités plus largement pour la conception de prototypes mais aussi comme support final pour les systèmes embarqués complexes.

1.1.3.3 Temps d'exécution des partitions logicielles sur une architecture multicœurs

D'une manière générale, l'exécution de partitions logicielles sur une architecture multicœurs présente des temps d'exécution moyens plus faibles que les architectures monocœur, et c'est là tout leur intérêt. La Figure 2 illustre l'évolution des temps d'exécution en fonction des avancées technologiques en matière d'architecture de processeur (architecture de Von Neumann, niveaux de cache, mécanismes de pipeline, et architecture multicœurs). Si les temps moyens d'exécution décroissent linéairement avec l'augmentation de la complexité, les pires délais d'exécution (en anglais : **WCETs (Worst Case Execution Time)**) augmentent de manière exponentielle à mesure que la complexité de l'architecture croît. L'augmentation des WCETs est d'autant plus importante lors du saut technologique entre les architectures monocœur et les architectures multicœurs.

L'augmentation des WCETs pour une architecture multicœurs par rapport à une architecture monocœur s'explique par la présence de ressources partagées entre les cœurs. En effet, les délais pire-cas pour l'accès aux ressources partagées (mémoire cache et IOs) sont importants. La gestion des ressources partagées requiert une étude précise pour l'utilisation des architectures multicœurs dans un environnement embarqué critique. Dans cette optique, de nombreux travaux scientifiques ont souligné l'importance d'une bonne gestion temporelle et spatiale des différents niveaux de mémoire cache (11), (12) et de l'accès aux IOs (13). Actuellement, il n'existe pas de règles ou de directives industrielles portant sur l'usage des processeurs multicœurs dans les systèmes avioniques.

Pour être conforme à la norme DO-178C, les autorités de l'aviation civile exigent un haut degré de déterminisme pour les temps d'exécution des partitions logicielles, et ce quelle que soit l'architecture matérielle sous-jacente. En d'autres termes, le calcul des WCETs de chaque partition logicielle est requis puisque les exigences en matière de certification sont telles que seuls les WCETs sont pris en compte et non les temps moyens d'exécution.

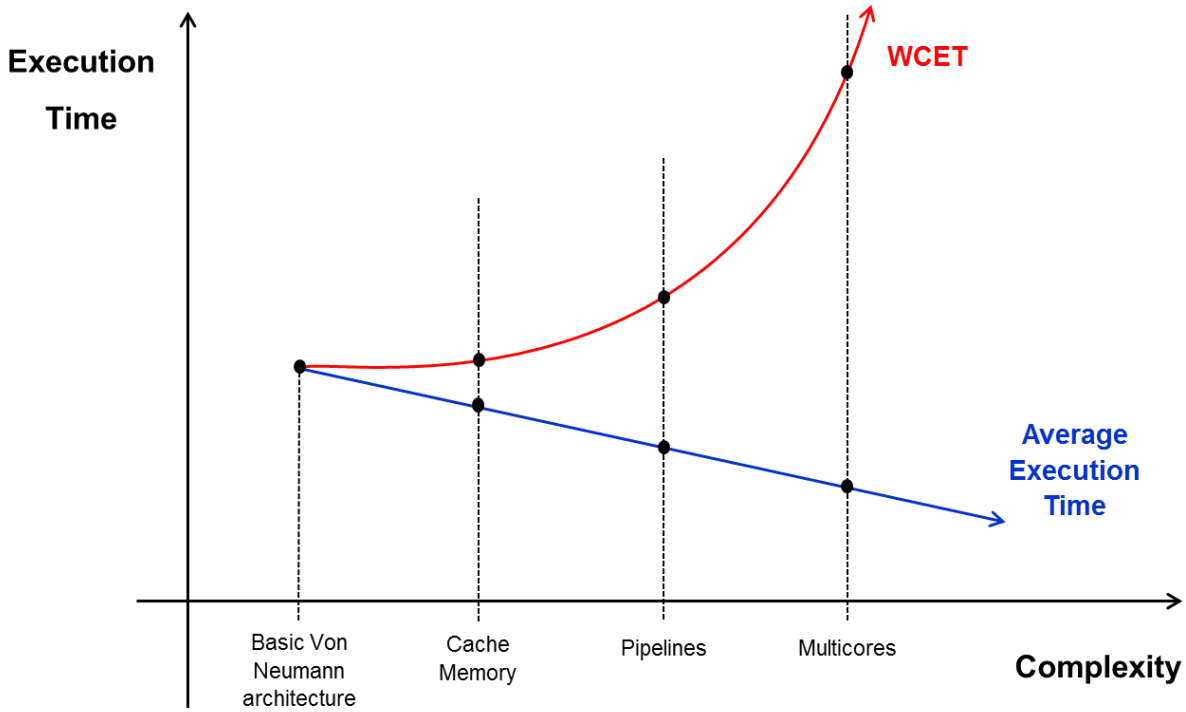


Figure 2 – Temps moyens d'exécution et WCETs en fonction de la complexité de l'architecture matérielle.

Pour résumer, les architectures multicœurs peuvent être utilisées en aéronautique à condition d'être en mesure de prouver le respect de WCETs pour les temps d'exécution des partitions logicielles hébergées. Or, la complexité intrinsèque de ces architectures rend difficile le calcul des WCETs. Pour un processeur COTS multicœurs, le calcul des WCETs peut être simplifié en restreignant les fonctionnalités et les optimisations du processeur. En d'autres termes, les architectures multicœurs doivent être configurées pour réduire les mécanismes qui introduisent des retards conséquents telles que les unités de pré-fetch, un nombre important d'étages de pipeline ou des mécanismes d'exécution spéculative (14). En revanche, les architectures MSPs sur FPGA présentent un degré de maitrise de la complexité supérieur aux processeurs multicœurs par l'emploi d'IPs matérielles et par le design d'IPs sur-mesure. Ainsi, le calcul des WCETs est en principe plus simple pour une architecture MSP que pour une architecture multicœurs basée sur un processeur COTS.

Par ailleurs, l'exigence du déterminisme a été exacerbée alors que les architectures avioniques ont connu une transition. En effet, les architectures fédérées basées sur des **LRUs (Line Remplaceable Units)** tendent à être abandonnées au profit d'architectures distribuées mettant en œuvre le principe de l'**IMA (Integrated Modular Avionics)** (15) embarqué à bord d'aéronefs comme l'Airbus A380 et le Boeing B787.

1.1.4 Méthodes de conception d'architectures matérielles

1.1.4.1 Architecture IMA

Le concept d'IMA est mis en avant par la plupart des acteurs du domaine aéronautique en tant que plateforme hautement intégrée permettant d'héberger plusieurs applications avioniques présentant différents niveaux de criticité. Prisaznuk (16) a été le premier à formuler clairement la définition de l'IMA : « *l'IMA est formé autour du concept de puissants modules de traitement informatique couplés à un système d'exploitation logiciel qui permet une exécution indépendante des partitions applicatives.* ». Par conséquent, l'objectif principal de l'IMA est d'apporter davantage d'intégration aux systèmes avioniques sur une plateforme partagée et partitionnée offrant le même niveau de confinement des fautes et de fiabilité qu'une architecture fédérée.

Par ailleurs, le facteur économique incluant la mise à jour d'éléments de l'architecture, la mise en place de nouvelles fonctions/services, une maintenance rapide et peu coûteuse (par la réduction du nombre de pièces de rechange) et la prévention de l'obsolescence, motivent le développement d'architectures IMAs.

1.1.4.2 Certification incrémentale

Le concept d'IMA a permis de reconsidérer le processus global de certification à l'échelle d'une seule plateforme partagée. Comme expliqué précédemment, une architecture multicœurs comprend un ensemble de ressources partagées entre les partitions logicielles. Le processus de certification « traditionnel » appliqué à une architecture multicœurs est une procédure longue et coûteuse. Elle consiste en l'obtention de crédits de certification à la fois pour les composants matériels et logiciels (17). L'inconvénient prévalent du processus de certification traditionnel est que toute modification de composants matériels ou toute mise à jour des partitions logicielles implique de reprendre tout le processus de certification. Cela engendre un coût financier en certification proche du coût initial de certification de l'architecture.

Pour éviter cela, la mise en œuvre de l'IMA au sein d'une architecture distribuée permet de certifier individuellement des modules et des composants logiciels de telle sorte que les crédits de certification peuvent être capitalisés et réutilisés partiellement en cas d'évolution de l'architecture. C'est ce qu'on appelle la *certification incrémentale* (18).

La certification incrémentale peut encore être définie comme un procédé d'obtention de crédits en vue de la certification en acceptant ou en concluant que le module IMA est conforme à des exigences spécifiques (DO-178C, DO-254, etc…). Cette acceptation progressive est divisée en tâches et les crédits accordés pour les tâches individuelles contribuent à l'objectif global de certification.

Six tâches définissent l'acceptation d'une architecture IMA du point de vue de la certification incrémentale :

- Tâche 1 : l'acceptation d'un module.
- Tâche 2 : l'acceptation matérielle et logicielle.
- Tâche 3 : l'acceptation du système IMA.
- Tâche 4 : l'intégration du système IMA à bord de l'avion.
- Tâche 5 : le changement de modules ou d'applications.
- Tâche 6 : la réutilisation de modules ou d'applications.

La certification incrémentale permet d'intégrer et d'accepter de nouvelles applications et/ou des modules dans une architecture IMA et de maintenir des applications et/ou des modules existants sans avoir à recertifier l'architecture complète (Figure 3). Sur le long terme, la certification incrémentale conduit à d'importantes économies.

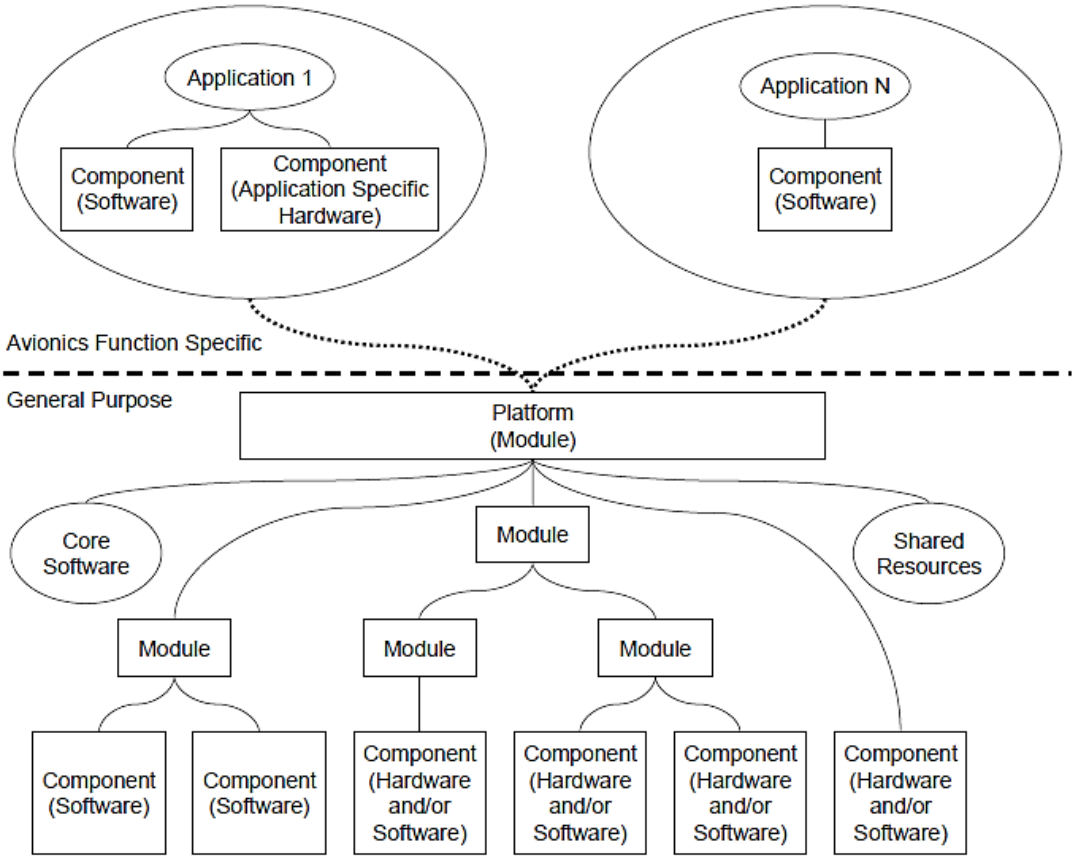


Figure 3 – Principe architectural de la certification incrémentale et de l'architecture IMA.

De plus, bien que plusieurs fonctions avioniques soient intégrées sur une unique plateforme, la responsabilité de la certification peut être répartie entre les parties prenantes, ce qui permet le développement parallèle de partitions logicielles et réduit le délai de mise sur le marché. Néanmoins, une isolation fonctionnelle entre les modules est requise pour faire de l'IMA, notamment au niveau des ressources partagées. On parle alors de *partitionnement*.

1.1.5 Méthodes d'isolation fonctionnelles

1.1.5.1 Partitionnement de DAL

Le partitionnement est une technique architecturale consistant à assurer l'indépendance de fonctions avioniques ou d'applications logicielles de façon à garantir que les couplages fonctionnels ne se produisent qu'intentionnellement. Il permet également de regrouper et d'isoler les fautes permanentes et transitoires mais aussi de réduire la complexité des processus logiciels de vérification. On distingue deux niveaux de partitionnement : le partitionnement dit de **DAL (Design Assurance Level)** et le partitionnement robuste.

Le partitionnement de DAL assure un niveau d'isolation tel qu'une partition logicielle peut affecter une autre partition mais de façon contrôlée pour que la fonction globale soit assurée et que les conséquences en matière de sécurité soient acceptables. Il est mis en œuvre dans le cas de partitions ayant des niveaux de DAL différents mais au sein d'une seule et même fonction avionique. L'objectif des analyses de partitionnement de DAL est de démontrer qu'un groupe de partitions ayant des niveaux de DAL différents se comportent comme une seule partition du plus haut DAL. D'un point de vue fiabilité, des erreurs résiduelles peuvent exister mais en accord avec le niveau de DAL imposé.

1.1.5.2 Partitionnement robuste

Le partitionnement robuste est généralement implémenté par construction dans les architectures fédérées pour assurer le confinement des pannes, mais il doit être garanti sur les plateformes IMAs afin que la certification incrémentale soit réalisable. Le partitionnement robuste a reçu de multiples définitions notamment dans des normes et dans des standards aéronautiques (19), (19) et (20). En particulier, l'ARINC 653 (20) stipule : « *Dans le but d'isoler des partitions sur des ressources partagées, l'architecture matérielle devrait fournir à l'OS la capacité de restreindre l'espace mémoire, le temps d'exécution et l'accès aux IOs pour chaque partition.* ».

Des études académiques ont tenté une définition du partitionnement robuste comme le *Gold Standard for Partitionning* formulée par J. Rushby (21) : « *Un système fortement partitionné garantit un niveau de confinement de panne équivalent à son système fédéré fonctionnellement équivalent.* ». De même, Wilding et al. (22) ont proposé une définition plus pratique nommée *Alternative Gold Standard for Partitionning* : « *Le comportement et la performance des logiciels dans une partition ne doivent pas être affectés par le logiciel de d'autres partitions.*» qui est généralement préférée en tant que condition suffisante pour démontrer le partitionnement robuste.

Le partitionnement robuste est utilisé dans le cas d'équipements hébergeant plusieurs fonctions avioniques utilisant des partitions logicielles ayant des niveaux de DAL identiques ou différents. Les analyses doivent démontrer que l'équipement est strictement équivalent d'un point de vue certification à plusieurs équipements hébergeant chacun une seule fonction avionique. Les considérations de DAL ne suffisent plus à la démonstration et le partitionnement robuste doit être

réalisé au plus haut niveau de DAL des partitions hébergées. Par conséquent, l'application de la définition de Wilding impose une isolation spatiale et temporelle des modules avec des répercutions probables sur les WCETs, et plus généralement sur les performances de la plateforme.

L'isolation spatiale est une contrainte qui protège les données d'une partition logicielle situées dans une mémoire partagée contre les accès invalides demandés par d'autres partitions. Sur une architecture multicœurs, les accès mémoires sont souvent traités par la **MMU (Memory Management Unit)** du processeur piloté par un **RTOS (Real Time Operating System)** (23). Le principe est de compléter une table de configuration système avec les exigences de la mémoire physique et de cartographier les adresses virtuelles avec les adresses physiques grâce à des structures de décodage d'adresses générées par le RTOS ou directement attribuées via la table de configuration. En ce qui concerne les IOs partagées, l'isolation spatiale peut être assurée par le respect des exigences relatives à la prévention des violations d'accès.

Pour sa part, l'isolation temporelle exige que les caractéristiques de synchronisation d'une partition ne soient pas affectées par l'exécution d'une autre partition hébergée sur le même processeur ou sur un autre processeur. L'isolation temporelle facilite considérablement le calcul des WCETs puisque la plateforme embarquée doit fournir des réponses en temps-réel et présenter un comportement déterministe. Cependant, les architectures multicœurs sont conçues pour réduire les temps moyens d'exécution au détriment du niveau de complexité globale, qui lui augmente, et donc au détriment des propriétés de prévisibilité et des analyses temporelles (24).

1.1.5.3 Ressources partagées

Les ressources partagées (IOs et mémoires caches) peuvent être sources de conflits en raison de la présence d'arbitres optimisés qui implémentent des politiques dynamiques pour accorder l'accès aux ressources partagées via un moyen de communication ou via une mémoire partagée. Ces composants sont généralement mal documentés et aucune assurance n'est fournie sur la rigueur et la pertinence des tests effectués pour mesurer les pires délais d'accès.

Ainsi, bien que certains industriels aéronautiques tentent de certifier leur architecture multicœurs pour leurs séduisantes caractéristiques **SWaP (Size, Weight and Power)** (17), (25), il apparaît généralement que seules de fortes contraintes sur la conception de l'architecture, et notamment sur l'accès aux ressources partagées, peuvent rendre ces architectures déterministes. Par conséquent, les WCETs des architectures multicœurs sont sévèrement impactées, d'autant plus si le support choisi est un processeur COTS offrant peu de flexibilité. En revanche, une architecture MSP sur FPGA offre plus de souplesse sur la gestion des ressources partagées puisqu'il est possible de créer des arbitres sur mesure implémentant la politique de gestion d'accès appropriée afin de respecter la contrainte de déterminisme.

En conclusion, faire de l'IMA et de la certification incrémentale sur une architecture multicœurs tout en présentant des WCETs raisonnables est un défi majeur pour les équipementiers aéronautiques. Si la technique du partitionnement robuste apporte un haut degré d'isolation fonctionnelle entre les

partitions logicielles, les ressources partagées doivent être convenablement gérées pour éviter la propagation de pannes mais aussi le respect du déterminisme.

Dans cette optique de gestion déterministe des ressources partagées, nous en sommes venus à nous intéresser aux IOs. Après avoir établi la liste l'ensemble des IOs de l'architecture multicœurs, nous nous sommes concentrés sur l'IO présentant de loin le fonctionnement le plus complexe : l'**ES (End-System)**. L'ES est en charge de la conversion de protocole pour la communication via le réseau AFDX entre différents équipements de bord.

L'ES est divisé en deux parties fonctionnant en parallèle, l'une pour la transmission et l'autre pour la réception des trames. En réception, les trames reçues doivent être stockées dans une mémoire en attendant d'être traitées par l'ES. Le dimensionnement de cette mémoire est crucial pour assurer un traitement déterminisme de toutes les trames reçues. Il s'agit de la problématique traitée dans cette thèse.

Dans la section suivante, nous présentons le réseau AFDX afin de comprendre son fonctionnement général et les grands principes de la communication réseau.

1.2 Réseau de communication embarqué AFDX

Dans un contexte de forte augmentation du trafic aérien, les OEMs développent des systèmes embarqués de plus en plus complexes afin de répondre aux exigences en matière de sécurité et de confort. Les équipements aéronautiques sont constitués d'un ensemble de capteurs, d'unités de traitement de données et d'actionneurs. Ils sont stratégiquement distribués à bord pour réaliser diverses fonctions avioniques.

Alors que la complexité et le nombre de systèmes interconnectés grandissent, un équipement en fonctionnement échange à tout instant une importante quantité de données avec d'autres équipements. Ils doivent donc être correctement déployés et connectés via un réseau fiable. Le réseau AFDX répond aux besoins en bande passante et en fiabilité des équipements électroniques et il propose une transmission de trames déterministe et à haute fréquence.

1.2.1 L'AFDX : un réseau déterministe

Depuis sa mise en œuvre réussie sur l'Airbus A380, le réseau AFDX a été adopté comme une solution technologique crédible pour répondre aux contraintes fortes des communications embarquées. Le réseau AFDX est le fruit de la mise en œuvre de deux standards : l'ARINC 664 (3) pour le déterminisme, la redondance et le caractère full-duplex, et la norme IEEE 802.3 pour le protocole Ethernet et les couches logicielles. L'ARINC 664 standardise les réseaux déterministes, et en particulier le réseau AFDX dans la partie 7. Il pose des exigences pour disposer d'un moyen de communication temps-réel, fiable et déterministe.

Cependant, l'ARINC 664 ne décrit pas les moyens techniques pour atteindre les performances requises. En particulier, le déterminisme est la propriété fondamentale pour assurer qu'une trame transite de son terminal de transmission à son terminal de réception en un temps fini. En d'autres termes, le délai de bout en bout pendant lequel une trame traverse le réseau appartient à un intervalle de temps borné. Il revient à l'intégrateur réseau de prouver l'existence de ces bornes temporelles pour obtenir les crédits de certification émanant des autorités.

Par ailleurs, l'AFDX présente des améliorations significatives en termes de bande passante disponible, de fiabilité et d'adressage par rapport à de plus anciennes normes de communication telles l'ARINC 429 (2) ou l'ARINC 419 (26). Le réseau AFDX met en œuvre le concept de virtualisation de canaux pour réaliser des connexions multipoints et il propose une transmission de données à grande vitesse (100 Mbps).

1.2.2 Infrastructure du réseau AFDX

Dans cette étude, nous considérons un réseau AFDX homogène, c'est-à-dire qui ne contient que des éléments structurels propres à l'AFDX et non des interfaces (en anglais : *gateways*) vers d'autres réseaux ou bus de communication. Le réseau AFDX offre un service de transmission déterministe aux applications hébergées.

Comme illustré sur la Figure 4, trois éléments principaux constituent l'infrastructure d'un réseau AFDX : les terminaux de transmission et de réception (ESs), les commutateurs réseaux (en anglais : *switchs*) et les fils. Les ESs sont les nœuds sources et de réception du réseau. Les commutateurs sont les nœuds intermédiaires qui interconnectent les ESs via des fils. Les fils sont d'une conception particulière. Il s'agit de deux paires différentielles, l'une dédiée à la transmission et l'autre à la réception afin de permettre la transmission et la réception de trames simultanément et éviter les collisions de trames. Un bon niveau de robustesse contre les interférences électromagnétiques et le caractère full-duplex sont garantis par l'utilisation de ces fils d'interconnexion.

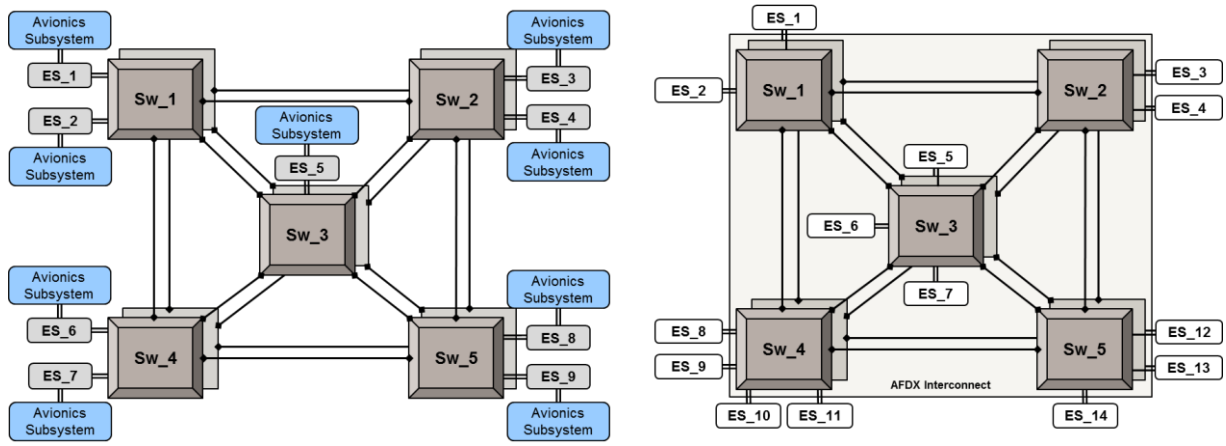


Figure 4 – Deux exemples de réseau AFDX comprenant cinq commutateurs redondés et de multiples ESs.

De plus, la redondance se traduit par la duplication des commutateurs et des fils. Ainsi, deux sous-réseaux AFDX parallèles sont constitués suivant la spécification donnée par l'ARINC 664 pour satisfaire la contrainte de fiabilité. Ces deux sous-réseaux transmettent les mêmes trames simultanément. Les ESs dupliquent les trames lors de la transmission et sélectionnent la première trame valide à la réception. Ces composants physiques du réseau AFDX assurent l'isolation spatio-temporelle des flux de trames.

Les deux exemples de la Figure 4 présentent des configurations restreintes de réseaux AFDX comprenant cinq commutateurs redondés (Sw_1 à Sw_5) et plusieurs ESs (ES_1 à ES_9 ou ES_14 suivant l'exemple). Chaque ES gère à la fois la transmission et la réception des trames. Un ES transmet des trames par un port de sortie vers un et un seul port d'entrée de commutateur, ce qui conduit par construction à un réseau en topologie étoile. Chaque port de sortie est associé à une mémoire régie par une politique **FIFO (First In First Out)** (3) et les ESs reçoivent les trames via un port d'entrée.

1.2.3 Mécanismes de ségrégation de flux de trames

La propriété de déterminisme et l'isolation spatio-temporelle des flux de trames nécessitent des mécanismes de ségrégation des flux de trames. Le premier mécanisme consiste en une contrainte d'isolation spatiale des trames. Il s'agit de permettre la communication multipoint entre les ESs, tout en garantissant une indépendance entre les communications par la mise en œuvre de canaux virtuels appelés **VLs (Virtual Links)**.

Le second mécanisme consiste en une contrainte d'isolation temporelle des trames. Il s'agit de réguler le flux de trames sur chaque VL en imposant un délai minimum entre l'envoi de trames successives sur un même VL. Ce délai est appelé le **BAG (Bandwidth Allocation Gap)** et il est l'un des paramètres caractéristiques d'un VL. Ces deux mécanismes de ségrégation sont présentés en détail dans les paragraphes suivants.

1.2.3.1 Notion de Virtual Link

Les VLs sont des canaux virtuels de communication assurant la transmission déterministe de données entre un ES source et un ou plusieurs ESs de réception. Il est précisé qu'un ES est capable de gérer à la fois la transmission (ES source) et la réception (ES de réception) des trames.

La Figure 5 montre un réseau comprenant trois commutateurs (S1, S2 et S3) et sept ESs. Les VLs sont des chemins unidirectionnels définis statiquement. Un VL partage la bande passante disponible sur le lien physique en canaux virtuels de telle sorte que les partitions logicielles puissent instancier des communications indépendantes. Par conséquent, les ESs sources peuvent instancier des communications vers un seul ES de réception (communications *unicasts*) ou plusieurs ESs de réception (communications *multicasts*). Sur la Figure 5, un ensemble de VLs unicasts interconnecte des ESs sources à des ESs de réception. Un chemin de communication est défini comme une série d'éléments du réseau en commençant par l'ES source et en terminant par l'ES de réception. Par exemple, $P_4 = \{ES_2,$

S_1, S_2, S_3, ES_5} est le chemin suivi par le VL **v4** dont l'ES source est ES_2 et l'ES de réception est ES_5. Les transmissions multicasts ne sont pas considérées dans notre exemple.

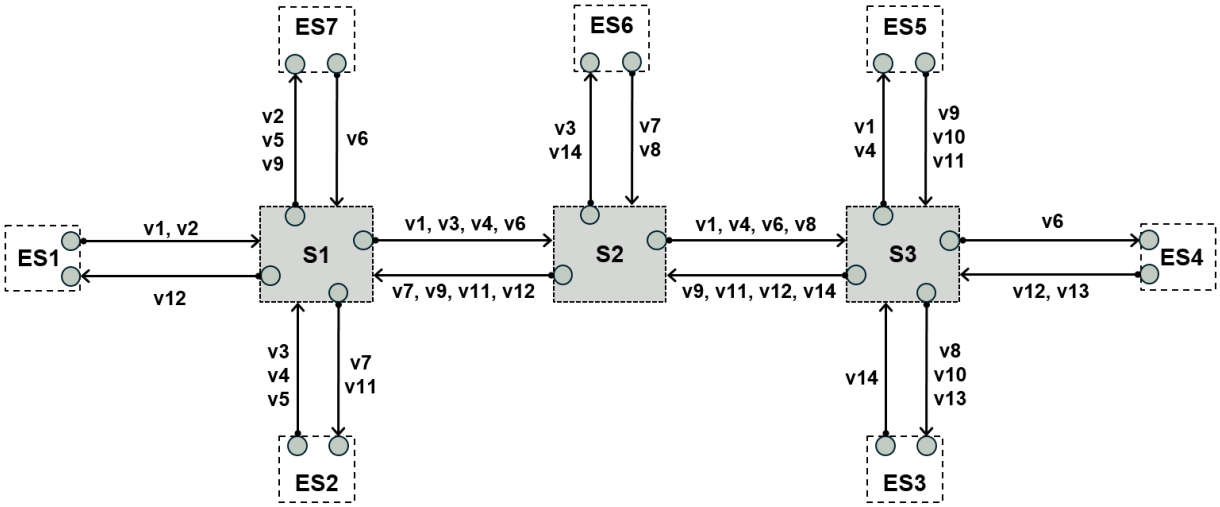


Figure 5 – Exemple de configuration réseau AFDX comprenant de multiples VLs unicasts.

En outre, l'isolation spatiale apportée par les VLs immunise les trames contre la propagation d'un dysfonctionnement d'un ES à l'autre. En effet, si un comportement anormal affecte un VL, les VLs qui partagent les mêmes ressources physiques ne sont pas affectés.

1.2.3.2 Régulation de trafic à la source

Pour faciliter la démonstration du déterminisme, les VLs sont contraints de respecter une durée minimale entre la transmission de deux trames consécutives envoyées sur un même VL : le BAG.

Notons $\boldsymbol{i}$ l'identifiant de l'$i^{\text{ème}}$ VL de la table de configuration de l'ES comprenant $\boldsymbol{n}$ VLs. Chaque $\mathbf{VL}_{\boldsymbol{i}}$ est caractérisé par une valeur unique, notée $\mathbf{BAG}_{\boldsymbol{i}}$, qui varie en puissance de 2 de 1,0 ms à 128 ms (3), soit :

$$\forall\, i \in [\![1\,;n]\!], \exists\, p \in [\![0\,;7]\!] \quad \text{tel que} \quad \mathrm{BAG}_i = 2^p$$

Le BAG garantit une charge globale limitée sur le réseau. Il est ajusté en fonction des besoins en bande passante des partitions logicielles. Cependant, le BAG ne limite pas la durée maximale entre deux trames consécutives de telle sorte que la transmission des trames sur un VL n'est pas forcément périodique.

Enfin, un VL_i est caractérisé par la longueur maximale d'une trame susceptible de circuler sur celui-ci. Nous notons $\boldsymbol{L_{i,\max}}$ cette longueur. L'ARINC 664 contraint la plage de valeurs que peut prendre $L_{i,\max}$ telle que :

$$\forall\, i \in [\![1\,;n]\!], \quad L_{i,\max} \in [64\,;1\,536]$$

Ainsi, un couple (BAG_i ; $L_{i,\max}$) définit un VL_i et ces paramètres sont inscrits pour chaque VL_i dans la table de configuration de l'ES.

1.2.3.3 Preuve du déterminisme

La preuve du déterminisme du réseau AFDX repose avant tout sur le choix d'une méthode pour démontrer que quel que soit le VL du réseau considéré, le temps de transit maximal de l'ES source à l'ES de réception (délai de bout en bout) est borné et peut être calculé. Cela doit être vrai quelle que soit la configuration du réseau ou le fonctionnement intrinsèque des nœuds de réseau. Les deux méthodes les plus utilisées dans la littérature sont le *Network Calculus* (27) et l'*Approche par Trajectoires* (28). D'autres méthodes existent mais elles sont plus marginales comme la méthode du *Forward End-to-End delays Analysis* (29).

Proposée en 1991 par Cruz (30) et appliquée au contexte avionique en 1998 par Le Boudec (27), la méthode du Network Calculus est actuellement utilisée pour calculer la borne supérieure du délai de bout en bout de chaque VL pour des avions civils comme l'A380, l'A350 ou le Boeing 747. Au moment de la rédaction de ce manuscrit, il s'agit de la méthode majoritairement utilisée et la plus fiable, l'Approche par Trajectoire n'ayant pas encore démontré un calcul des bornes temporelles fiable comme l'attestent des contre-exemples mettant à mal l'Approche par Trajectoires (31) malgré des études adressant ce genre de contre-exemples (32).

Le Network Calculus consiste à établir une courbe dite *courbe de service* pour chaque nœud du réseau, puis de sommer les pires délais sur un VL à chaque fois qu'une trame traverse un nœud. Cette courbe traduit le pire délai de traitement qu'un nœud puisse présenter. Plus précisément, le flux est caractérisé en un point du réseau par une fonction croissante qui représente le trafic cumulé durant un certain temps (t_1 ; t_2), noté $\boldsymbol{R(t_1, t_2)}$. Une rafale de trafic se traduit par une forte augmentation de R sur un intervalle temporel de faible amplitude. L'idée est de borner l'importance de ces rafales par une fonction qui ne dépend que de l'intervalle temporel, il s'agit d'une enveloppe de flux $\boldsymbol{R^*}$ telle que :

$$\forall(t_1,\ t_2), \qquad 0 < t_1 < t_2 \ \Rightarrow \ R(t_1, t_2) \ < \ R^*(t_2 \ - \ t_1)$$

Dans ses travaux de thèse, Grieu analyse le Network Calculus et présente des méthodes pour réduire la borne haute du délai basées sur la priorisation des trames, des heuristiques et des algorithmes génétiques multicritères (33). Par ailleurs, Ridouard et al. proposent une approche stochastique du Network Calculus dans le contexte de files d'attente à priorités statiques (34). Adnan et al. se concentrent plutôt sur la mise au point d'un automate temporel basé sur le Network Calculus pour calculer précisément les bornes supérieures du délai de bout en bout (35). En revanche, cette dernière méthode ne donne pas de très bons résultats lorsque le nombre de VLs devient élevé. Charara et al., quant à eux, ont conçu un outil de simulation basé sur le Network Calculus afin de montrer l'existence d'un délai maximum de bout en bout mesurable pour chaque VL (36).

Le Network Calculus présente l'avantage de fournir des bornes absolues car tous les intervalles temporels sont considérés et ces bornes sont obtenues en considérant les extremagoulot des temps de

traitement et d'attente. Cette méthode permet d'évaluer la taille en octets des files d'attente des commutateurs pour le stockage des trames par différence verticale entre les courbes d'arrivée et de service.

Des méthodes plus réalistes ont émergé à la suite du Network Calculus pour calculer des bornes hautes des délais de bout en bout. Parmi ces méthodes, la plus connue est celle proposée par Bauer et al. nommée l'Approche par Trajectoires (37), (38) et traitée par la suite dans des travaux d'optimisation (31), (39). Le principe général de cette méthode repose sur la considération de l'influence individuelle d'un VL particulier sur les autres et le chemin physique du VL, sans faire d'hypothèses sur la date d'arrivée des trames. Ainsi, les bornes hautes calculées avec cette méthode sont valides pour toutes les dates d'arrivée possibles des trames.

Enfin, d'autres travaux mettent l'accent sur des solutions pour réduire les délais de bout en bout des trames sur le réseau. Suthaputchakun et al. (40) et Gurjar et al. (41) ont prouvé que des politiques d'ordonnancement de type *Longest Queue* ou *FIFO* conduisent à une réduction du jitter de transmission. De plus, une attribution de priorités multiples aux trames d'un VL offre différents niveaux de **QoS (Quality of Service)** selon le niveau de criticité de la trame. Ainsi, Hamza et al. (42) s'appuient sur des niveaux de priorités appliqués aux trames et ils appliquent l'Approche par Trajectoires pour montrer une réduction des délais de bout en bout pour les trames de priorités supérieures.

1.2.4 Commutateur AFDX

Une fois transmise au réseau AFDX par l'ES source, la trame AFDX circule sur l'infrastructure réseau et elle est amenée à traverser un ou plusieurs commutateurs AFDX. Le rôle d'un commutateur s'apparente à celui d'une station d'aiguillage : une trame entre dans le commutateur, son en-tête **MAC (Media Access Control)** indique son adresse de destination et le commutateur place cette trame sur le port de sortie adéquat, de façon à rapprocher la trame de son ES de destination (ES de réception) selon le chemin statique de son VL. Le choix du port de sortie est basé sur l'adéquation entre les paramètres de la table de routage statique du commutateur et les informations contenues dans l'en-tête MAC de la trame. Cette politique de commutation est de type *store and forward*. Les commutateurs AFDX prévoient une mémoire de stockage de type FIFO sur chaque port de sortie pour éviter une perte de trames en cas de congestion. Les ports de sortie comprennent un étage de mémoire pouvant stocker jusqu'à 512 trames de taille maximale (3). Lors de l'envoi de trames sur le lien physique en sortie de commutateur à la vitesse de de 100 Mbps, plusieurs trames se retrouvent accolées les unes aux autres et forment ainsi une séquence de trames accolées (en anglais : **SBF (Sequence of Back-to-back Frames)**).

Un commutateur peut traiter plusieurs trames en entrée simultanément mais ces traitements induisent une latence sur la retransmission des trames, ce qu'on appelle *délai de commutation*. Pour l'essentiel, ce délai consiste en une latence technologique et en une phase d'attente dans le port de sortie. Le délai de commutation est variable et dépend de la charge du commutateur à un instant donné. Il est inclus dans le calcul du délai bout en bout pour chaque VL donc il est nécessairement borné, même en cas de congestion d'un ou plusieurs ports de sortie, puisque les délais de bout en bout

sont eux-mêmes bornés. Un fort trafic de trames peut conduire à l'apparition de congestions et augmenter le délai de bout en bout de certaines trames. En particulier, un conflit d'accès à un port de sortie entre deux trames peut entraîner une latence supplémentaire pour la trame retardée. Ce délai de commutation variable aura son importance dans la suite de notre étude.

À cela s'ajoutent d'autres fonctions de services obligatoires ou recommandées par l'ARINC 664 comme indiqué sur la Figure 6. Ces fonctions de services sont : des fonctions de filtrage, de surveillance des trames, de prise de décision en cas de fonctionnement anormal, mais aussi des fonctions qui surveillent l'état de santé du commutateur (en anglais : fonctions de *monitoring*)), et qui font remonter tout dysfonctionnement.

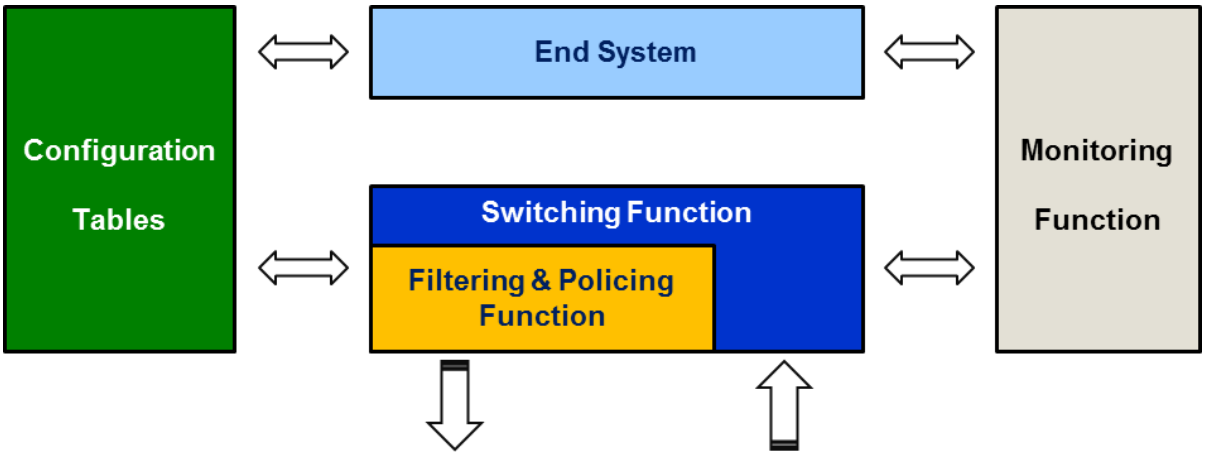


Figure 6 – Principaux blocs fonctionnels d'un commutateur AFDX selon l'ARINC 664.

Par ailleurs, la Figure 6 indique la présence d'une table de configuration qui contient tous les paramètres nécessaires à l'initialisation du commutateur et d'un ES commutateur. L'ES commutateur permet de communiquer avec le commutateur lors d'une phase de chargement de données (en anglais : *data loading*) ou lors d'opérations de monitoring. Ces modes de fonctionnement sont en dehors du cadre de ces travaux qui se concentrent uniquement sur un mode de fonctionnement standard de transmission entre des ESs placés au sein d'équipements aéronautiques autre que l'infrastructure réseau.

1.3 End-System, IO de transmission et de réception AFDX

Un ES est un élément de l'infrastructure réseau permettant d'échanger des données de manière fiable et déterministe entre les équipements aéronautiques. Il met en œuvre le protocole AFDX pour assurer la communication par l'échange de trames et il est partagé en deux parties traitant indépendamment les trames en réception et en transmission.

Dans notre étude, nous nous concentrons sur la partie réception de l'ES, mais certains aspects de la partie transmission sont également présentés, notamment ceux qui ont des conséquences sur la forme du flux de trames entrant dans la mémoire de réception.

1.3.1 End-System source

1.3.1.1 Vue d'ensemble de la partie transmission

La Figure 7 représente le modèle d'un ES source et illustre les différentes opérations qu'une trame subit depuis la génération des données par les partitions logicielles et le chargement de la trame sur le réseau AFDX. En transmission comme en réception, les services ESs sont divisés en trois couches : la couche UDP + IP, la couche *AFDX Special Features* et la couche MAC + PHY (IEEE 802.3). Ces couches rappellent en partie le protocole classique Ethernet. La principale différence entre le protocole AFDX et le protocole Ethernet réside dans la couche intermédiaire AFDX Special Features.

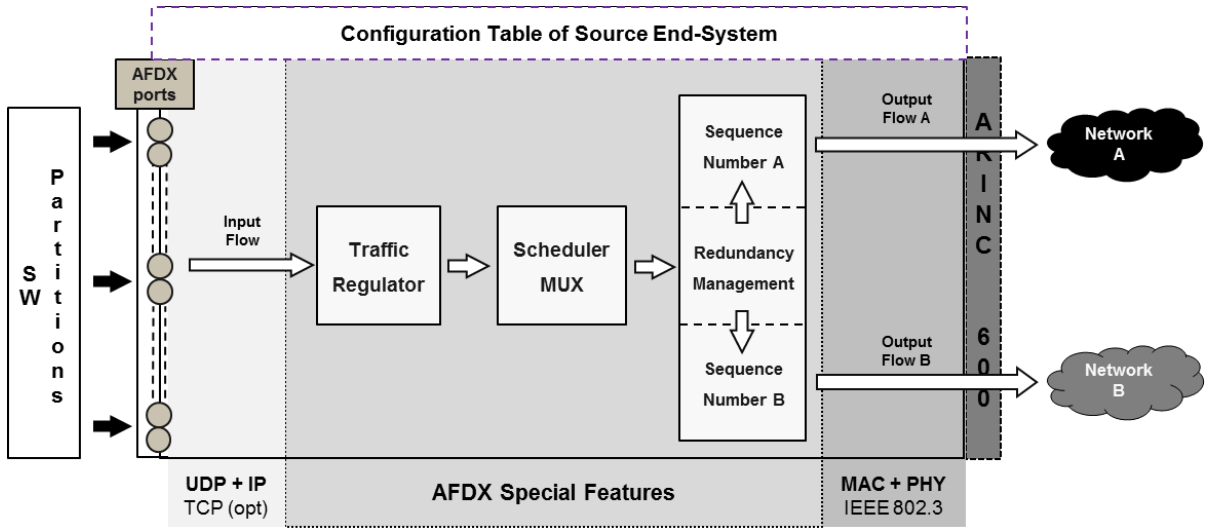


Figure 7 – Modèle de l'ES source.

Détaillons à présent le cheminement des données au travers de l'ES source, ce qui permet d'expliquer le rôle de la couche AFDX Special Features. Les données à transmettre sont générées par les partitions logicielles. Ces données sont placées sur les ports d'entrée AFDX de l'ES source (port *Sampling*, *Queuing* ou **SAP (Service Access Point)**). Ensuite, les trames AFDX sont créées par l'agrégation de données applicatives et par l'ajout d'en-têtes UDP et IP dont le rôle est d'assurer la transmission des dites données vers le bon équipement.

Il s'en suit les opérations liées à la couche AFDX Special Features. Le régulateur de trafic assure l'envoi périodique des trames suivant le BAG de leur VL, inscrit dans la table de configuration de l'ES source. Puis, l'ordonnanceur multiplexe les envois de trame pour s'assurer de la disponibilité du lien physique puisqu'une seule trame à la fois peut être transmise au réseau. L'unité de gestion de la redondance duplique les trames si un envoi redondant est défini en table de configuration, et l'unité

d'attribution de **SNs (Sequence Numbers)** fournit un numéro unique à chaque trame transmise en fonction du VL. Ce nombre est incrémenté de 1 à chaque envoi sur un VL.

Enfin, les trames reçoivent leur en-tête MAC et la couche PHY transfert la trame aux sous-réseaux AFDX A et B.

1.3.1.2 Jitter de transmission

Si nous faisons l'hypothèse d'un ES source comprenant un seul VL, la transmission des trames via ce VL peut être BAG-périodique puisqu'il n'y a pas de concurrence pour l'utilisation du lien physique.

À présent, supposons que l'ES source possèdent un nombre n de VLs en transmission, ce qui est un cas de figure plus général que le précédent. En effet, les partitions logicielles nécessitent habituellement plusieurs VLs pour communiquer avec plusieurs autres équipements. Comme les VLs de l'ES source doivent se partager le lien physique, l'ordonnanceur placé en aval du régulateur de trafic multiplexe les trames concurrentes selon une politique FIFO. Par conséquent, des trames peuvent se retrouver retardées dans leur transmission sur le réseau en raison d'un conflit d'accès au lien physique. Le retard ainsi généré est nommé *jitter de transmission.*

La Figure 8 représente le jitter de transmission $\boldsymbol{J^T_{i,j}}$ comme un intervalle de temps borné, où $\boldsymbol{i}$ est l'identifiant de l'$i^{\text{ème}}$ VL de la table de configuration de l'ES source et $\boldsymbol{j}$ l'identifiant unique de la trame sur le VL_i. La valeur maximale du jitter de transmission est notée $\boldsymbol{J^T_{\max}}$. L'ES source assure la transmission des trames $F_{i,1}$, $F_{i,2}$ et $F_{i,3}$ sur chaque VL_i de telle sorte que le premier bit de chaque trame apparaisse sur le réseau AFDX avant la date $J^T_{\max}$.

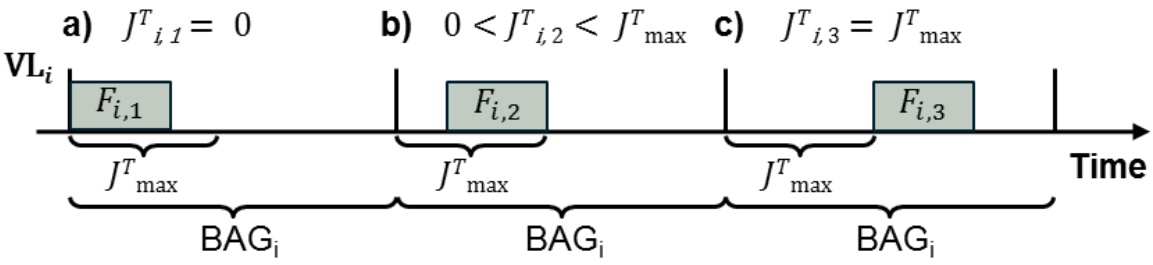


Figure 8 – Effets du jitter de transmission sur la transmission des trames pour un VL_i.

L'ARINC 664 impose une valeur pour la borne maximale du jitter de transmission $J^T_{\max}$ telle que, pour chaque VL d'un ES source :

$$\begin{cases} J^T_{\max} \leq 40 + \sum_{i=1}^{n} \dfrac{(20 + L_{i,\max}) \times 8}{C} \\ \text{ou} \\ J^T_{\max} \leq 500\ \mu\text{s} \end{cases}$$

Où :

- $J_{\max}^{T}$ est la valeur maximale autorisée du jitter de transmission, en µs.
- 40 est le jitter technologique minimum, en µs.
- 20 est la somme en octets de trois champs de la trame AFDX : le préambule (7 octets), le **SFD (Start Frame Delimiter)** (1 octet) et l'**IFG (Inter Frame Gap)** (12 octets).
- $L_{i,\max}$ est la longueur maximale d'une trame circulant sur VL_i définie sur la table de configuration de l'ES source, en octets.
- C est la bande passante moyenne ou la vitesse de transmission du réseau, en bits/s.

Ainsi, l'intégrateur réseau dispose de deux moyens de calcul pour la borne maximale de $J_{\max}^{T}$. Quelle que soit la table de configuration de l'ES source, $J_{\max}^{T}$ ne peut excéder 500 µs. La première formule impose une valeur de $J_{\max}^{T}$ inférieure à 500 µs pour les petites tables de configuration comprenant peu de VLs, dans le cas d'une politique FIFO en transmission. Certains travaux comme ceux de Ding et al. (43) ou de Ren et al. (44) traitent du contrôle du jitter de transmission et proposent des modèles pour simuler et vérifier la propriété du déterminisme. D'autres travaux optimisent l'utilisation du VL en proposant un mécanisme de changement de phase (45) ou en proposant des topologies de réseau optimisées (46).

Le jitter de transmission constitue une première source de perturbation sur la périodicité du flux de trames en réception. Bien que sa valeur soit faible, l'accumulation de jitters de transmission sur plusieurs VLs issus d'ESs sources différents peut avoir une influence non négligeable sur la manière dont les trames sont reçues au niveau de l'ES de réception.

Dans l'ensemble des développements de cette thèse, nous prenons $J_{\max}^{T} = 500$ µs.

1.3.1.3 Couche de traitement MAC + PHY

Dans cette sous-partie, nous revenons sur la couche de traitement MAC + PHY afin de la présenter plus en détail. Comme suggérée par la Figure 7, les composants de la couche MAC + PHY sont redondés. La couche MAC + PHY fait partie de la norme IEEE 802.3 (47) faisant elle-même partie d'un ensemble de normes édictées sous l'égide du comité de standardisation IEEE 802. Il s'agit de spécifications pour l'implémentation de réseaux numériques locaux à liaison filaire. Par rapport au modèle de référence **OSI (Open Systems Interconnection)**, la norme 802.3 décrit les contraintes des couches physiques (PHY) (couche 1) et liaisons (couche 2), cette dernière étant plus communément appelée couche MAC. La couche MAC + PHY regroupe donc deux couches : la couche MAC et la couche PHY.

La couche MAC insère des champs dans la trame AFDX représentée sur la Figure 9 et notamment : le mode de réception de la trame (unicast ou multicast), l'adresse source et l'adresse de destination suivant les paramètres de la table de configuration de l'ES source. Ces opérations sont effectuées en parallèle sur les trames à transmettre aux sous-réseaux A et B. Des champs sont ajoutés aux trames : le

préambule, le SDF, le **CRC (Cyclic Redundancy Code)** et des « 0 » inutiles contenus dans le payload (en anglais : *padding*).

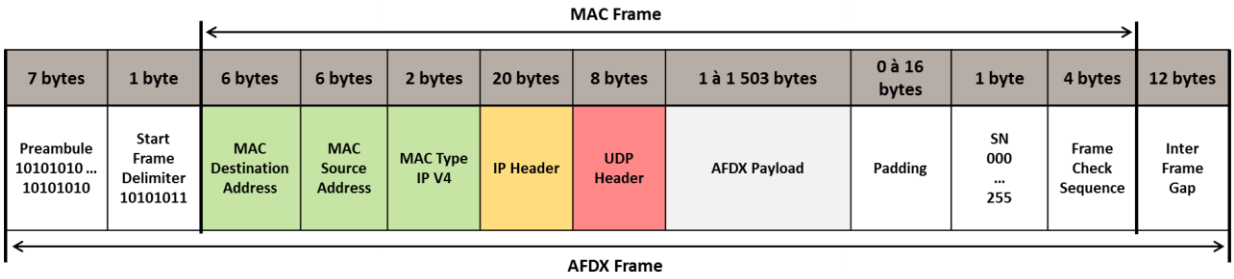


Figure 9 – Structure d'une trame AFDX.

Enfin, la couche PHY se charge de sérialiser le flux de bits grâce à l'émetteur-récepteur (en anglais : *transceiver*). Les transceivers PHY sont également redondés. Le rôle principal d'un transceiver est de générer un signal binaire et différentiel cadencé à 100 Mbps.

Pour un ES source conçu sur un support FPGA, les éléments matériels de la couche MAC sont standards et peuvent être récupérés au sein des librairies d'IPs. Les transceivers de la couche PHY sont généralement externes au FPGA. Pour résumer, l'ensemble des éléments de la couche MAC + PHY sont matériels et les traitements dans cette couche sont donc rapides.

1.3.1.4 Couche de traitement UDP + IP

La couche UDP + IP du standard ARINC 664 est la couche la plus éloignée du matériel. La couche UDP + IP regroupe les couches **IP (Internet Protocol)** et **UDP (User Data Protocol)** normalisées dans le protocole Ethernet (48).

1 byte	1 byte	2 bytes	1 byte	2 bytes	1 byte	1 byte	2 bytes	4 bytes	4 bytes
Vers – 4 bits IHL – 4 bits	Service	Longueur Totale	Indentification	Flag – 3 bits Position fragment – 13 bits	**TTL (Time To Live)**	Protocole	Checksum	IP Source	IP Destination

Figure 10 – Structure de l'en-tête IP.

La couche IP correspond à la couche réseau (couche 3) du modèle OSI. La Figure 10 représente l'en-tête standard IP avec ses différents champs. En transmission, la couche IP doit contenir de nombreuses informations et notamment :

- La version de protocole IP.
- L'**IHL (Internet Header Length)** qui est la longueur en mots de 32 bits de l'en-tête IP.
- Le champ Service : gestion de la qualité de service.
- La longueur totale : en-tête IP + payload.

- Le champ Identification : numéro de fragment. Tous les fragments ont le même numéro d'identification.
- Le champ Flag : état de la fragmentation.
- Le champ Position fragment : position du fragment par rapport aux autres.
- Le champ **TTL (Time To Live)** : durée de vie en secondes de la trame.
- Le champ Protocole : type de données qui se trouvent derrière l'en-tête IP.
- Le calcul du Checksum (49).
- L'adresse source.
- L'adresse de destination.

La couche UDP correspond à la couche transport (couche 4) du modèle OSI. Le protocole UDP est utilisé pour transmettre de petites quantités de données de façon rapide. Au niveau de la couche transport, on ne parle plus d'adresses mais de ports. Un numéro unique est alloué à chaque port. Celui-ci est standardisé dans l'ARINC 664 comme l'indique le Tableau 2.

Type of port	Type of communication	Port range	Commentaries
AFDX Communication port	AFDX ⇔ AFDX AFDX ⇔ Compliant network	1 024 – 65 535	Used for sampling and queuing communications
SAP	AFDX ⇔ AFDX AFDX ⇔ Compliant network	0 – 1 023	Used for standard communications e.g Port 69 to open TFTP, Data loading, SNMP, etc…
	AFDX ⇔ AFDX AFDX ⇔ Compliant network	1 024 – 65 535	Used for bi-directional communication : specific TFTP, etc…

Tableau 2 – Allocation des ports UDP selon l'ARINC 664.

De plus, comme l'AFDX est un réseau fermé, l'architecte réseau est en principe libre de choisir les numéros de port en utilisant la totalité des valeurs possibles : 0 – 65 535. Cependant, la spécification incite à respecter l'utilisation de valeurs préconisées par le Tableau 2, cela pour éviter d'éventuels conflits avec un réseau adjacent si le réseau AFDX est connecté à un autre réseau via une gateway (cas des réseaux hétérogènes, non traités ici).

La Figure 11 représente l'en-tête standard UDP avec ses différents champs. En réception, la couche UDP est chargée de quelques vérifications comme le contrôle du port source, du port destination et un checksum.

2 bytes	2 bytes	2 bytes	2 bytes
Port Source	Port Destination	Longueur	Checksum

Figure 11 – Structure de l'en-tête UDP.

Les traitements de la couche UDP + IP sont réalisés par des fonctions logicielles en raison de la complexité des tests à mener et de la mouvance de certains champs (par exemple, les adresses de ports sont susceptibles d'évoluer). Les informations requises pour les traitements IP et UDP sont disponibles dans la table de configuration de l'ES source. Enfin, le détail des mécanismes logiciels ne sera pas spécifié dans cette étude pour des raisons de confidentialité industrielle.

1.3.1.5 Payload

Le payload est l'ensemble des données « utiles » transportées à l'intérieur de la trame AFDX. La structure du payload est présentée sur la Figure 12, d'après la spécification Airbus (3).

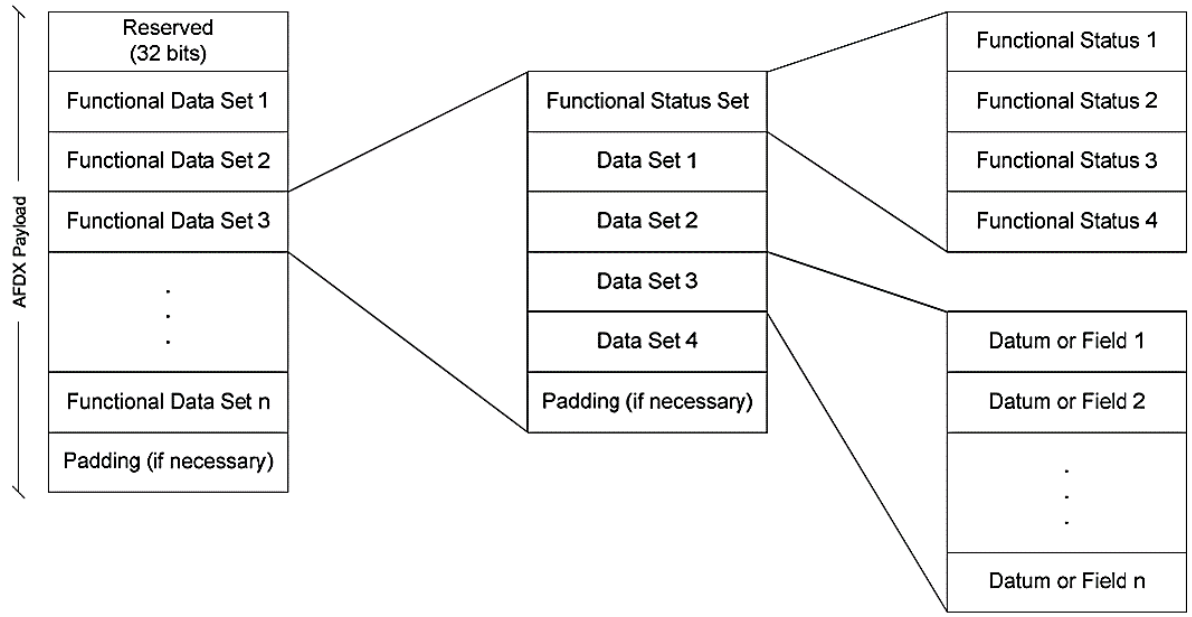


Figure 12 – Structure du payload de la trame AFDX selon la spécification Airbus.

Le payload est constitué de plusieurs **FDS (Functional Data Set)**. Chaque FDS contient quatre **DS (Data Set)** et un **FSS (Functional Status Set)**. Un FSS est formé de quatre octets qui indiquent le statut de chaque DS. Enfin, un DS est une collection de données paramétriques (altitude, vitesse, pression, etc…).

1.3.2 End-System de réception

1.3.2.1 Vue d'ensemble de la partie réception

La Figure 13 représente le modèle d'un ES en réception et illustre les différentes opérations qu'une trame subit entre sa réception depuis le réseau AFDX jusqu'à la mise à disposition des données auprès des applications logicielles.

De même que pour la partie transmission, les services de l'ES de réception sont divisés en trois couches : MAC + PHY (IEEE 802.3), AFDX Special Features et UDP + IP.

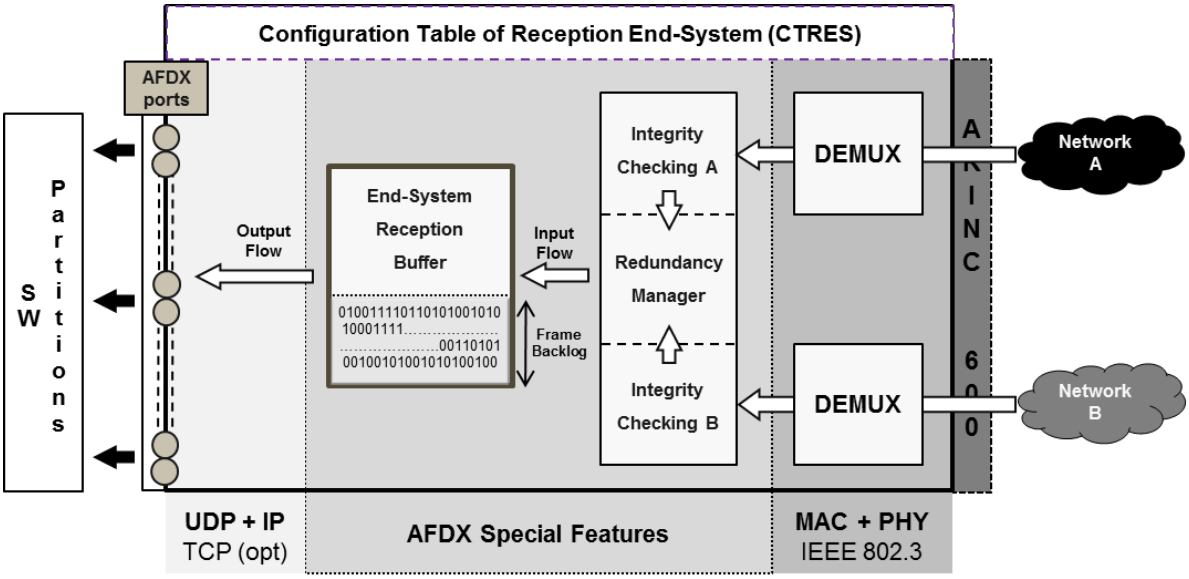


Figure 13 – Modèle de l'ES de réception.

Reprenons le cheminement des trames au travers de l'ES de réception. Le flux de bits provenant des sous-réseaux A et B est d'abord de-sérialisé par les transceivers de la couche PHY. Les transceivers PHY sont redondés comme en transmission et sont externes au FPGA. Le rôle d'un transceiver en réception est d'échantillonner le signal différentiel et de générer un signal cadencé à 50 MHz sur 2 bits.

La couche MAC vérifie ensuite les informations relatives à la trame, et notamment : la validité de l'adresse de la trame vis-à-vis de la table de configuration en réception de l'ES (CTRES), la validité du code CRC et la longueur de la trame (min : 64 octets et max : 1 536 octets). Ces tests sont effectués en parallèle sur les trames issues des sous-réseaux A et B et donc sur deux exemplaires de la même trame. Les trames invalides sont filtrées et des champs sont supprimés : le préambule, le SDF, le CRC et le padding.

Si les trames passent la couche MAC avec succès, elles sont transférées à la couche supérieure : la couche AFDX Special Features. La couche AFDX Special Features gère la redondance et elle réalise différentes opérations de contrôle sur les trames. Le bloc *Integrity Checking* vérifie le numéro du SN en considérant le VL auquel appartiennent les trames. Puis, le bloc *Redundancy Manager* supprime la redondance des trames. Il ne conserve que la première trame valide reçue. Cette trame est alors stockée dans la mémoire de réception dans l'attente de son traitement UDP et IP tandis que l'autre trame est définitivement supprimée.

De la même manière que pour l'ES source, l'ensemble des traitements évoqués jusqu'à présent pour l'ES de réception sont réalisés avec des composants matériels (IPs VHDL) tandis que les traitements de la couche UDP + IP sont réalisés avec des composants logiciels. Dans ce rapport, nous

ne détaillons pas les traitements de la couche UDP + IP en réception qui se trouve être symétrique à ceux menés en transmission.

1.3.2.2 Dimensionnement de la mémoire de réception

Au sein de l'ES de réception, nous venons de voir que les mécanismes de traitement répartis en couches sont réalisés par des composants à la fois logiciels et matériels. Les composants matériels traitent les trames au fil de l'eau et ralentissent peu les trames en arrivée. Il n'en est pas de même pour les composants logiciels de la couche UDP + IP. Le nombre conséquent de vérifications logicielles de cette couche peut être à l'origine de délais importants. Du point de vue fonctionnel, l'interface entre la couche matérielle AFDX Special Features et la couche logicielle UDP + IP est donc un goulot d'étranglement de l'ES. Il en résulte le placement stratégique d'une mémoire de réception de type FIFO à cette interface afin de garantir la non-perte de trames.

Ce premier phénomène est amplifié par l'absence de synchronisation des ESs sources du réseau avec une horloge globale. Par conséquent, les trames peuvent se concurrencer pour l'utilisation des ressources du réseau comme les ports de sortie des commutateurs non-préhensibles et cela peut entraîner des congestions sporadiques. Dans un tel scénario, une séquence de trames accolées (SBF) se propage sur le lien physique du commutateur jusqu'à l'ES de réception. Ainsi, une accumulation de trames en attente de traitement apparaît dans la mémoire de réception et forment ce que l'on appelle un *backlog*.

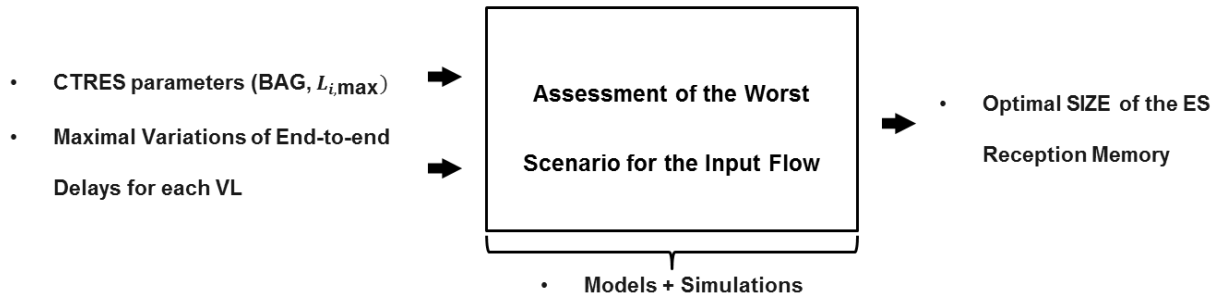


Figure 14 – Modèle synthétique de la problématique du dimensionnement de la mémoire de réception ES.

Notre travail porte sur l'optimisation de la taille de la mémoire de réception de type FIFO. La taille de cette mémoire affecte à la fois la fiabilité du réseau et les délais de transmission. La Figure 14 présente un modèle synthétique de notre problématique avec ses entrées et ses sorties. Ainsi, à partir des éléments de la CTRES c'est-à-dire de l'ensemble des paramètres des VLs, et des variations maximales des délais de bout en bout données pour chaque VL, nous allons construire des modèles pour estimer le pire scénario de réception dans le but de déterminer la taille optimale de la mémoire de réception ES. Ces modèles permettent de construire le flux de trames entrant et sortant de la mémoire, la taille mémoire étant alors obtenue par l'écart maximal entre ces deux flux.

La problématique du dimensionnement de la mémoire de réception apparaît dans une publication de Bauer et al. où une méthode basée sur l'Approche par Trajectoires permet une estimation du backlog dans les ports de sortie d'un commutateur (38). D'autres études se sont basées sur des VLs à différents niveaux de priorités et les conséquences sur le backlog des ports de sortie en considérant deux niveaux de priorités (50), (51) ou n niveaux de priorités comprenant une mémoire par niveau de priorités (52).

Cependant, peu d'attention a été accordée à la mémoire de réception ES parce qu'elle n'est pas prise en compte dans le calcul des délais bout en bout. À notre connaissance, aucune étude antérieure n'a exploré la question de l'optimisation de la mémoire de réception à partir de la CTRES et de la variation des délais. L'ES de réception doit traiter les trames en temps-réel (3), tout en assurant un niveau de fiabilité tel qu'aucune trame ne soit ni perdue ni corrompue, en particulier en cas de SBFs. Par conséquent, la mémoire de réception mérite une attention particulière afin d'éviter des débordements ou au contraire, le gaspillage de ressource mémoire.

Deuxième partie

Estimations du pire scénario de réception pour la mémoire de réception ES

Chapitre 2 : Première approche : modèle pessimiste

Une mémoire de type FIFO (désignée par la suite comme la *mémoire de réception*) est placée dans l'ES de réception, à l'interface entre les couches de traitement AFDX Special Features et UDP + IP. Jusqu'à présent, la solution retenue consistait à dimensionner la mémoire de réception de façon à être suffisamment grande pour éviter toute perte de trame. Or, une architecture matérielle sur FPGA impose une limitation de la mémoire RAM sur puce. Il s'agit donc de limiter l'espace mémoire requis pour l'architecture, en particulier l'espace mémoire occupé par la mémoire de réception ES.

Pour ce faire, nous modélisons le flux de trames entrant et sortant de la mémoire. En réception, un ES reçoit un flux de trames selon sa CTRES, puis le remplissage de la mémoire de réception suit le principe du *seau percé*. D'un côté, la mémoire se remplit lors de la réception de trames, et de l'autre, elle se vide avec un débit supposé plus faible lors de la lecture de trames. L'enjeu principal de notre étude porte sur la modélisation du flux de trames entrant, c'est-à-dire sur les trames provenant du réseau AFDX, et nous essayons d'estimer le pire scénario de réception conduisant à un remplissage maximal de la mémoire. Une heuristique est alors formulée sur ce pire scénario en supposant que le remplissage maximal se produit lors de la réception de la plus longue séquence de trames accolées. Plus le nombre de trames reçues ou la quantité d'octets reçue à la suite est conséquente, plus l'espace mémoire requis pour stocker les trames en attente de traitement est important.

Construire la plus longue séquence de trames accolées est l'enjeu de ce chapitre. Dans la Section 2.1, un modèle pour le flux entrant composé d'un seul VL est proposé et nous formalisons les délais de bout en bout ainsi que la variation de la durée entre la réception de deux trames consécutives. La Section 2.2 présente le modèle théorique de flux entrant sur un nombre n de VLs en posant toutes les heuristiques et les hypothèses permettant la construction pessimiste de la plus longue séquence de trames accolées. La Section 2.3 propose des résultats de simulation pour l'estimation du pire scénario de réception pour une CTRES industrielle, mais également l'étude de l'influence des paramètres CTRES sur le calcul du flux entrant. Enfin, la Section 2.4 propose un modèle simple du flux de trames sortant de la mémoire à partir duquel il est possible de réaliser des simulations du pire backlog mesuré. La Section 2.5 conclut le chapitre.

2.1 Modèle du flux de trames entrant pour un VL

Afin d'estimer le WFB, nous devons d'abord caractériser précisément le flux de trames entrant de la mémoire de réception ES. Dans cette première section, toutes les notations et les hypothèses adéquates pour caractériser le flux sur un seul VL sont posées.

2.1.1 Délai de bout en bout $\Gamma_{i,j}^{R}$

2.1.1.1 Définition

Plaçons-nous dans un scénario typique de transmission d'une trame $F_{i,j}$ d'un ES source vers un ES de réception. Après un temps d'attente égal au jitter de transmission $J_{i,j}^{T}$, l'ES source charge la trame $F_{i,j}$ sur le réseau. Ensuite, la trame $F_{i,j}$ traverse un ou plusieurs commutateurs AFDX suivant le chemin de son VL_i dans lesquels elle subit des traitements qui induisent un délai de commutation noté $\boldsymbol{\theta_{i,j}}$.

En outre, le délai de bout en bout dépend du VL_i et de la trame $F_{i,j}$ considérés puisque le chemin physique sur le réseau suivi par chaque VL peut traverser différents commutateurs AFDX et rencontrer des charges réseaux variables à un instant t. Plusieurs VLs traversent chaque commutateur et il existe des phénomènes de couplage entre les VLs dans le sens où la charge d'un VL peut influencer les délais de commutation $\theta_{i,j}$ subis par les trames des autres VLs partageant le même commutateur. Le chemin suivi par un VL_i reste identique même si une congestion se produit sur le port de sortie d'un commutateur.

Ainsi, pour une trame $F_{i,j}$ sur un VL_i, le délai de bout en bout noté $\boldsymbol{\Gamma_{i,j}^{R}}$ pour la réception de $F_{i,j}$ est la somme de son jitter de transmission $J_{i,j}^{T}$ et de son délai de commutation $\theta_{i,j}$, ce qui s'écrit :

$$\Gamma_{i,j}^{R} = J_{i,j}^{T} + \theta_{i,j}$$

Où i est l'identifiant du VL de la CTRES et j l'identifiant unique de la trame sur le VL_i.

La Figure 15 rassemble ces premières notations pour une trame $F_{i,j}$ émise à l'instant 0 sur un VL_i.

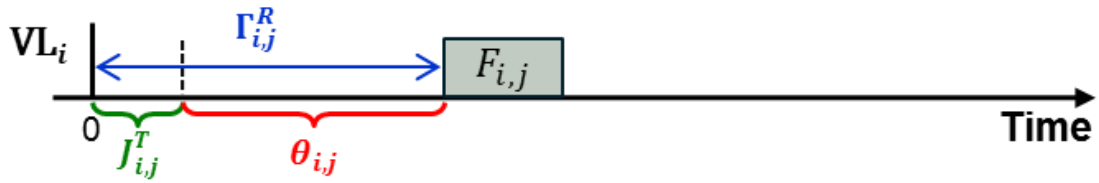


Figure 15 – Modèle pour les délais appliqués à une trame $F_{i,j}$ appartenant à un VL_ide la CTRES.

Pour rappel, le jitter de transmission est borné et sa valeur maximale est $J^T_{\max} = 500$ µs . De même, le délai de commutation $\theta_{i,j}$ est un délai variable et borné (43) car il est inclus dans le calcul du délai de bout en bout $\Gamma^R_{i,j}$ qui doit être borné pour tout VL_i afin d'assurer le caractère déterministe du réseau (3). Ceci peut se formuler de la manière suivante pour un nombre n de VLs dans la CTRES :

$$\forall\,(i,j) \in [\,1\,;\,n\,] \times [\,0\,;\,+\infty[, \qquad \begin{cases} J^T_{i,j} \in [0\,;\,J^T_{\max}] \\ \theta_{i,j} \in [\theta_{i,\min}\,;\,\theta_{i,\max}] \end{cases}$$

Où $\theta_{i,\min}$ est le délai de commutation minimal que peut subir une trame sur le VL_i et $\theta_{i,\max}$ le délai de commutation maximal.

Nous pouvons en déduire l'expression des valeurs minimales et maximales du délai de bout en bout $\Gamma^R_{i,j}$.

2.1.1.2 Délai de bout en bout minimal $\boldsymbol{\Gamma^R_{i,\min}}$

Le délai de bout en bout minimal, noté $\boldsymbol{\Gamma^R_{i,\min}}$, peut être mesuré si la trame $F_{i,j}$ est envoyée sur le réseau AFDX immédiatement après sa création dans l'ES source (jitter nul), et si la trame $F_{i,j}$ ne rencontre aucune congestion durant la traversée du réseau. $\Gamma^R_{i,\min}$ vaut alors :

$$\Gamma^R_{i,\min} = J^T_{\min} + \theta_{i,\min}$$

$$\boxed{\Gamma^R_{i,\min} = \theta_{i,\min}}$$

La Figure 16 illustre le cas d'un délai de bout en bout minimal pour une trame $F_{i,j}$ émise à l'instant 0 sur un VL_i.

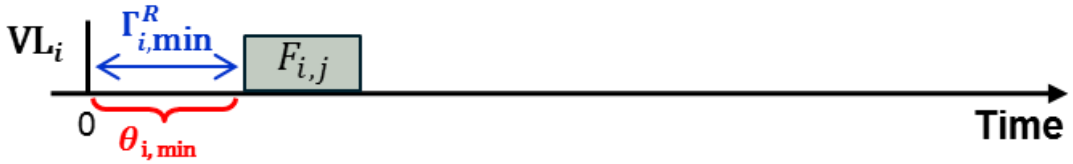


Figure 16 – Modèle du délai minimal appliqué à une trame $F_{i,j}$ appartenant à un VL_ide la CTRES.

2.1.1.3 Délai de bout en bout maximal $\boldsymbol{\Gamma^R_{i,\max}}$

Le délai de bout en bout maximal, noté $\boldsymbol{\Gamma^R_{i,\max}}$, peut être mesuré si la trame $F_{i,j}$ est envoyée sur le réseau AFDX avec un jitter maximal, et si le retard de commutation est maximal pour le VL_i (la trame $F_{i,j}$ peut avoir rencontré une ou plusieurs congestions durant la traversée du réseau suivant la charge réseau). $\Gamma^R_{i,\max}$ vaut alors :

$$\boxed{\Gamma^R_{i,\max} = J^T_{\max} + \theta_{i,\max}}$$

La Figure 16 illustre le cas d'un délai de bout en bout maximal pour une trame $F_{i,j}$ émise à l'instant 0 sur un VL_i.

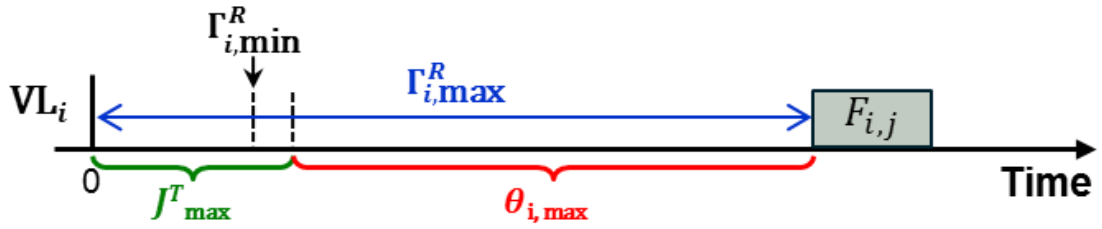


Figure 17 – Modèle du délai maximal appliqué à une trame $F_{i,j}$ appartenant à un VL_ide la CTRES.

2.1.2 Variations du délai de bout en bout $\Delta\Gamma^R_{i,j}$

2.1.2.1 Définition

La variation du délai de bout en bout, notée $\boldsymbol{\Delta\Gamma^R_{i,j}}$, entre deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ sur un VL_i s'écrit :

$$\Delta\Gamma^R_{i,j} = \Gamma^R_{i,j+1} - \Gamma^R_{i,j}$$

À présent, nous allons chercher à caractériser la variation maximale du délai de bout en bout.

2.1.2.2 Amplitude maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$

Nous rappelons que le jitter de transmission $J^T_{i,j}$ et le délai de commutation $\theta_{i,j}$ sont variables et bornés. Ce sont les deux composants essentiels du délai de bout en bout $\Gamma^R_{i,j}$ que nous considérons dans notre modèle de réception de trames. Concernant la variation du délai de bout en bout $\Delta\Gamma^R_{i,j}$, il existe une valeur maximale pour la variation des délais de bout en bout compte tenu de l'existence d'un délai de bout en bout maximal $\Gamma^R_{i,\max}$, et ce pour chaque VL_i. Cette valeur est notée $\boldsymbol{\Delta\Gamma^R_{i,\max}}$. $\Delta\Gamma^R_{i,\max}$ est la différence entre le délai de bout en bout minimal $\Gamma^R_{i,\min}$ et le délai de bout en bout $\Gamma^R_{i,\max}$ qu'une trame $F_{i,j}$ peut rencontrer. Cela s'écrit :

$$\Delta\Gamma^R_{i,\max} = \Gamma^R_{i,\max} - \Gamma^R_{i,\min}$$

Puis, en remplaçant par les expressions précédentes, nous obtenons :

$$\Delta\Gamma^R_{i,\max} = J^T_{\max} + \theta_{i,\max} - \theta_{i,\min}$$

L'expression de $\Delta\Gamma^R_{i,\max}$ qui nous intéresse dans notre modèle est la première. Les valeurs de $\Gamma^R_{i,\min}$ $\Gamma^R_{i,\max}$ sont des entrées du modèle pessimiste permettant d'estimer le pire-cas de réception pour la mémoire de l'ES. Ces mesures de délai de bout en bout ont été réalisées dans différents travaux (53) (54) et il est donc facile d'accéder à la valeur de $\Delta\Gamma^R_{i,\max}$.

2.1.2.3 Durée de réception effective d'une trame δ_i^R

Enfin, nous posons $\boldsymbol{\delta_i^R}$ comme étant la durée effective de la réception de la trame $F_{i,j}$ au niveau de l'ES de réception, c'est-à-dire la durée de chargement de la trame depuis le réseau vers l'ES de réception. δ_i^R dépend de la vitesse du réseau notée $\boldsymbol{C}$, et de la longueur maximale d'une trame du VL_i notée $\boldsymbol{L_{i,\max}}$ telle que :

$$\delta_i^R = \frac{L_{i,\max}}{C}$$

2.1.3 Durée entre la réception de deux trames $\Psi_{i,j}$

La caractérisation des délais de bout en bout $\Gamma_{i,j}^R$ pour la réception d'une trame $F_{i,j}$ appartenant à un VL_i étant faite, le point de vue est changé et nous nous plaçons du côté de l'ES de réception. En réception, l'ES n'a pas accès aux délais de transit des trames reçues sur le réseau. En revanche, il peut mesurer la durée entre la réception de deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ du même VL_i sur le lien physique. Cette durée est notée $\boldsymbol{\Psi_{i,j}}$. Elle dépend du VL_i et des dates de réception des trames $F_{i,j}$ et $F_{i,j+1}$, comme présenté sur la Figure 18.

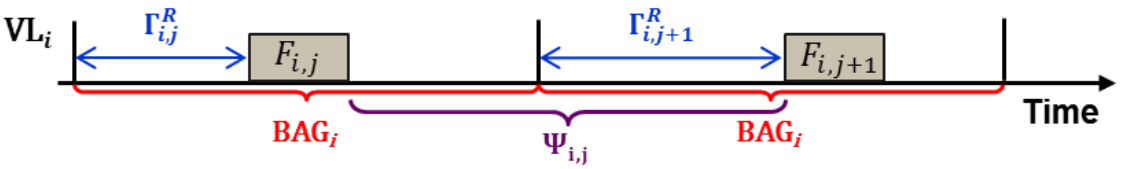


Figure 18 – Durée $\Psi_{i,j}$ pour la réception de deux trames consécutives sur un VL_i.

Sur la Figure 18, la valeur de $\Psi_{i,j}$ est moyenne, c'est-à-dire proche de la valeur de BAG_i.

De plus, la durée $\Psi_{i,j}$ illustre la variabilité du délai de bout en bout $\Delta\Gamma_{i,j}^R$. En effet, il existe un lien entre la durée entre la réception de deux trames consécutives $\Psi_{i,j}$ et le délai de bout en bout $\Gamma_{i,j}^R$. Ce lien peut être déterminé de façon analytique tel que, si nous considérons deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ sur un même VL_i, il vient :

$$\Psi_{i,j} = \Gamma_{i,j+1}^R + \left(\mathrm{BAG}_i - \delta_i^R - \Gamma_{i,j}^R\right)$$

$$\Psi_{i,j} = \Delta\Gamma_{i,j}^R + \left(\mathrm{BAG}_i - \delta_i^R\right)$$

2.1.3.1 Durée minimale $\Psi_{i,\,\min}$

La durée minimale entre la réception de deux trames consécutives sur un VL_i est notée $\boldsymbol{\Psi_{i,\,\min}}$. $\Psi_{i,\,\min}$ peut être mesurée si la trame $F_{i,j}$ subit un délai de bout en bout maximal, et si la trame suivante $F_{i,j+1}$ subit un délai de bout en bout minimal, comme illustré sur la Figure 19. Ce scénario de réception

peut se produire dans des conditions défavorables sur les délais de bout en bout (congestion ponctuelle, jitter).

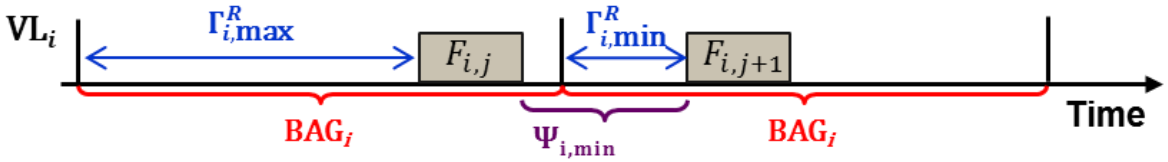


Figure 19 – Durée $\Psi_{i,\min}$ pour la réception de deux trames consécutives sur un VL_i.

De même que précédemment, nous proposons une expression analytique de $\Psi_{i,\min}$:

$$\Psi_{i,\,\min} = \Gamma^R_{i,\min} + \left(\mathrm{BAG}_i - \delta^R_i - \Gamma^R_{i,\max}\right)$$

$$\boxed{\Psi_{i,\,\min} = -\Delta\Gamma^R_{i,\max} + \left(\mathrm{BAG}_i - \delta^R_i\right)}$$

Le signe « - » devant le terme $\Delta\Gamma^R_{i,\max}$ retient immédiatement l'attention. $\Psi_{i,\min}$ étant une durée, elle ne peut pas être négative, ce qui impose qu'un certain seuil à $\Delta\Gamma^R_{i,\max}$. Ce seuil est obligatoirement respecté à cause des contraintes physiques de la réception sur le réseau : une seule trame peut être reçue à la fois. De plus, l'ordre d'arrivée des trames à l'ES de réception respecte l'ordre de transmission de celles-ci sur le réseau pour un cas de fonctionnement normal. Autrement dit, nous faisons l'hypothèse que les trames reçues sont valides du point de vue du bloc Integrity Checking de la couche AFDX Special Features.

Par ailleurs, l'ARINC 664 impose un délai minimum à respecter entre deux trames circulant sur le réseau AFDX (3). Ce délai minimum est noté $\boldsymbol{\varepsilon}$ et il vaut :

$$\boxed{\varepsilon = \frac{20_{\mathrm{bytes}}}{C}}$$

Où 20 est la somme en octets de trois champs de la trame AFDX : le préambule (7 octets), le SFD (1 octet) et l'IFG (12 octets), et C est vitesse de transmission du réseau en byte/s.

Donc, par définition, ε est la plus petite durée $\Psi_{i,\min}$, ou encore qu'ε minore la durée $\Psi_{i,\min}$:

$$\varepsilon \leq \Psi_{i,\,\min}$$

Il s'en suit l'expression suivante pour le seuil de $\Delta\Gamma^R_{i,\max}$:

$$\varepsilon \leq -\Delta\Gamma^R_{i,\max} + \left(\mathrm{BAG}_i - \delta^R_i\right)$$

$$\boxed{\Delta\Gamma^R_{i,\max} \leq \mathrm{BAG}_i - \delta^R_i - \varepsilon}$$

2.1.3.2 Conditions d'accolement de trames

À partir des équations précédentes, nous obtenons des résultats très importants pour la suite de nos développements. Il s'agit des conditions pour que deux trames consécutives en réception sur un même VL_i soient accolées.

La première condition porte sur la durée minimale $\Psi_{i,\,\mathrm{min}}$ entre la réception de deux trames sur un VL_i. Pour pouvoir observer un potentiel accolement de trames sur un VL, il faut nécessairement que :

$$\Psi_{i,\,\mathrm{min}} \;=\; \varepsilon$$

Et donc que $\Delta\Gamma^{R}_{i,\mathrm{max}}$ atteigne sa valeur de seuil :

$$\boxed{\Delta\Gamma^{R}_{i,\mathrm{max}} \;=\; \mathrm{BAG}_i - \varepsilon - \delta^{R}_{i}}$$

La seconde condition porte sur la durée $\Psi_{i,j}$ entre la réception de deux trames sur un VL_i. Pour que deux trames soient accolées, il faut que $\Psi_{i,j}$ atteigne localement la valeur d'ε, pour un j donné.

$$\Psi_{i,j} \;=\; \varepsilon$$

D'où :

$$\boxed{\Delta\Gamma^{R}_{i,j} \;=\; \varepsilon \;-\; (\mathrm{BAG}_i - \delta^{R}_{i})}$$

Cela peut se produire si $\Gamma^{R}_{i,j+1}$ est plus petit que $\Gamma^{R}_{i,j}$ pour que la trame $F_{i,j+1}$ « rattrape » la trame $F_{i,j}$. Dans ce cas, un SBF de deux trames est reçu au niveau de l'ES de réception.

2.1.3.3 Durée maximale $\Psi_{i,\,\mathrm{max}}$

La durée maximale entre la réception de deux trames consécutives sur un VL_i est notée $\boldsymbol{\Psi_{i,\,\mathrm{max}}}$. La durée $\Psi_{i,\,\mathrm{max}}$ dépend des extrema des délais de bout en bout pour un VL_i donné, eux-mêmes fournis par l'utilisateur. Elle aura une importance capitale pour l'estimation du pire scénario de réception sur n VLs présenté dans la section suivante.

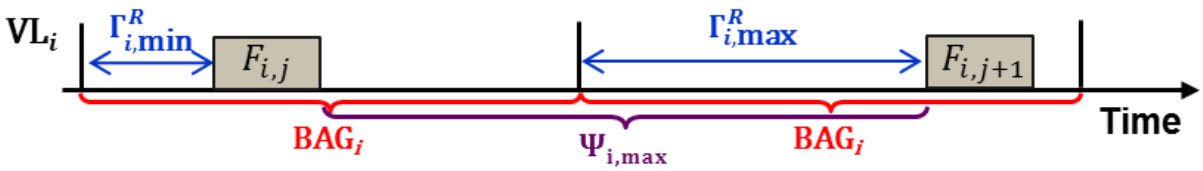


Figure 20 – Durée $\Psi_{i,\mathrm{min}}$ pour la réception de deux trames consécutives sur un VL_i.

À l'inverse de la durée minimale $\Psi_{i,\,\mathrm{min}}$, $\Psi_{i,\,\mathrm{max}}$ peut être mesurée si la trame $F_{i,j}$ subit un délai de bout en bout minimal, et si la trame suivante $F_{i,j+1}$ subit un délai de bout en bout maximal, comme

illustrée sur la Figure 20. Ce scénario de réception peut se produire dans des conditions défavorables sur les délais de bout en bout (congestion ponctuelle, jitter).

De même que précédemment, nous proposons une expression analytique de $\Psi_{i,\max}$:

$$\Psi_{i,\,\max} = \Gamma^R_{i,\max} + \left(\mathrm{BAG}_i - \delta^R_i - \Gamma^R_{i,\min}\right)$$

$$\boxed{\Psi_{i,\,\max} = \Delta\Gamma^R_{i,\max} + \left(\mathrm{BAG}_i - \delta^R_i\right)}$$

2.2 Méthode de construction pessimiste du flux de trames entrant sur n VLs

Pour évaluer la taille de la mémoire de réception ES, nous devons d'abord établir la forme du flux de trames entrant dans la mémoire pour un nombre n de VLs, et ensuite considérer ce flux entrant dans un scénario de réception conduisant à un backlog maximal dans la mémoire. Pour construire ce scénario de réception pour n VLs, une méthode de construction pessimiste basée sur des heuristiques est mise en œuvre.

La méthode de construction pessimiste comprend deux étapes principales. La première étape consiste à générer un flux de trames périodique et fini sur n VLs dans des conditions défavorables en supposant les variations de délais de bout en bout $\Delta\Gamma^R_{i,j}$ nulles. Cette étape est détaillée, une fois les hypothèses sur la périodicité du flux formulées.

La variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ pour chaque VL_i est prise en compte dans la seconde étape de la méthode de construction.

2.2.1 Heuristiques et hypothèses générales

Pour appliquer les développements précédents à une CTRES comprenant n VLs, nous proposons une méthode de construction basée sur des heuristiques et des hypothèses afin d'estimer le pire scénario de réception de trames.

2.2.1.1 LSBF en réception

L'heuristique majeure sur laquelle va reposer l'ensemble de la méthode de construction se formule ainsi :

Heuristique 2.1 : *Le scénario de réception conduisant au backlog maximal (WFB) dans la mémoire de réception ES correspond à la réception de la plus longue séquence de trames accolées (LSBF) sur le flux entrant dans la mémoire.*

Cette première heuristique séduit notre intuition scientifique dans le sens où il paraît naturel de considérer que la réception d'un grand nombre de trames accolées conduit à un remplissage maximal de la mémoire, et ce quelle que soit la forme du flux de sortie. Par conséquent, le but de la méthode de construction pessimiste n'est pas de construire le flux de trames entrant sur une longue durée, mais uniquement de déterminer le LSBF en réception pour toute CTRES, le LSBF étant le plus grand des SBFs reçus.

L'heuristique 2.1 s'accompagne d'une hypothèse de transmission maximale sur le réseau pour une configuration réseau donnée, ce qui revient énoncer l'hypothèse :

Hypothèse 2.1 : *Tous les ESs sources du réseau transmettent leurs trames BAG-périodiquement sur leurs VLs respectifs.*

En effet, au sein des ESs sources, le régulateur de trafic veille à ce que deux trames consécutives envoyées sur le même VL_i respectent la condition de BAG_i. Cependant, deux trames consécutives peuvent être séparées par un délai plus important que le BAG puisque le BAG ne limite que la période minimale entre deux envois successifs sur un VL_i. Si l'envoi de trames successives est plus lent que le BAG, la contribution d'un VL_i à la charge réseau n'est pas maximale. Or, comme nous cherchons à déterminer le LSBF, il semble que les SBFs ont plus de chance de se produire si la charge du réseau est maximale. Ainsi, il est nécessaire que tous les VLs du réseau saturent leur bande passante respective.

Une dernière hypothèse porte sur la longueur $L_{i,\max}$ des trames $F_{i,j}$ circulant sur chaque VL_i, toujours dans l'optique de maximiser la charge sur le réseau pour une configuration réseau donnée :

Hypothèse 2.2 : *Pour un VL_i donné, la longueur de toutes les trames de ce VL est égale à la longueur maximale $L_{i,\max}$.*

2.2.1.2 Charge du lien physique

Enfin, la validité de la méthode de construction repose sur le fait que les paramètres de la CTRES (nombre de VLs, BAGs et $L_{i,\max}$) conduisent à une charge limitée sur le lien physique entre le commutateur final et l'ES de réception.

Pour commencer, nous définissons la charge du lien physique (en anglais : *linkload*). La charge (en %) est le rapport du temps dédié à la transmission effective de trames entre le commutateur final et l'ES de réception sur une période de fonctionnement. La charge se calcule à partir des paramètres CTRES, et comme la CTRES est statique, la valeur de la charge est constante et s'écrit :

$$\mathrm{Linkload} = \frac{\sum_{i=1}^{n} \left(\frac{\mathrm{BAG}_{\max}}{\mathrm{BAG}_i} \times L_{i,\max} \times C \right)}{\mathrm{BAG}_{\max}}$$

Où $\mathbf{BAG_{max}}$ est la valeur maximale du BAG parmi toutes les valeurs des BAG_i de la CTRES. La durée $\mathrm{BAG}_{\max}$ est la période sur laquelle nous calculons la charge (*cf* : définition de l'hyperpériode dans la partie suivante).

Nous définissons quatre niveaux de charge :

- Moins de 10% – charge légère – CTRES légère.
- Entre 10% et 20% – charge modérée – CTRES modérée.
- Entre 20% et 40% – charge lourde – CTRES lourde.
- Plus de 40% – charge très lourde – CTRES très lourde.

Dans les cas pratiques, la charge doit être limitée pour assurer l'existence d'un ordonnancement de trames en réception sur le lien physique. Nous en venons à formuler l'hypothèse 2.3 sur la limitation de la charge :

Hypothèse 2.3 : *La charge du lien physique est limitée à 20%.*

Une charge limitée à 20% est réaliste du point de vue des configurations réseaux mises en œuvre sur les avions commerciaux (33).

2.2.2 Construction du flux périodique de trames

Venons-en à la première étape de la méthode de construction pessimiste. Pour cela, nous devons formuler un certain nombre d'hypothèses sur le flux de trames entrant sur n VLs.

2.2.2.1 Hypothèse sur la périodicité du flux

Nous formulons l'hypothèse suivante :

Hypothèse 2.4 : *Le flux de trames entrant est périodique sur chaque VL_i de la CTRES.*

Dans le cadre de l'hypothèse 2.1, l'hypothèse 2.4 implique que les délais de bout en bout $\Gamma_{i,j}^{R}$ de chaque VL_i soient constants, c'est-à-dire que sur un VL_i donnée, le jitter de transmission $J_{i,j}^{T}$ et le délai de commutation $\theta_{i,j}$ sont égaux pour toutes les trames $F_{i,j}$. Cela implique que la variation du délai de bout en bout $\Delta\Gamma_{i,j}^{R}$ entre deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ sur un VL_i est nulle :

$$\Delta\Gamma_{i,j}^{R} = 0$$

Et la durée $\Psi_{i,j}$ entre la réception de deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ sur un VL_i vaut :

$$\Psi_{i,j} \;=\; \mathrm{BAG}_i \;-\; \delta_i^{R}$$

De plus, l'hypothèse 2.4 implique que la réception des trames sur le lien physique est périodique en réception, et nous garantissons un placement des trames sur le lien physique tel que les trames n'entrent pas en collision.

2.2.2.2 Heuristique sur le placement initial des trames dans le flux

L'hypothèse 2.4 apporte un flux de trames entrant périodique et l'heuristique 2.1 nous invite à rechercher le LSBF à l'intérieur de ce flux. À partir du flux périodique, il est possible de créer un SBF par un placement des trames en cascade. Cela fait référence aux conditions défavorables évoquées au début de cette partie.

Heuristique 2.2 : *Les dates de transmission des trames au niveau des ESs sources rattachées à la CTRES associées à des délais de bout en bout constants et particuliers, peuvent conduire à une réception de SBFs périodiques.*

En réception, un SBF est donc reçu périodiquement, conformément aux hypothèses 2.1 et 2.4. Dans ce cas précis, cet SBF est appelé la plus longue séquence périodique de trames accolées (en anglais : **LPS (Longest Periodical Sequence)**). Le LPS se répète donc périodiquement. L'ordre des trames appartenant au LPS est aléatoire et nous verrons qu'il n'a pas d'impact sur la méthode de construction.

Le nombre de trames constituant le LPS, noté $\boldsymbol{k_{\mathrm{LPS}}}$, est proche du nombre de VLs. Si chaque VL de la CTRES contribue au LPS à hauteur d'une trame alors le LPS comprend n trames. Il est toutefois possible qu'un VL particulier (ou plusieurs) contribue à hauteur de deux trames, suivant la valeur des BAGs des VLs de la CTRES. Une contribution au-delà de deux trames par VLs est peu probable, compte tenu de l'hypothèse 2.3 qui limite la charge du lien à 20%.

De même que la réception pour un seul VL, une réception de trames accolées sur plusieurs VLs signifie que les trames consécutives sont séparées d'une durée ε sur le lien physique. Pour représenter les trames sur le lien physique à partir des VLs, il suffit de considérer une projection des trames de chaque VL_i sur un seul axe temporel.

Enfin, le LPS dure $\boldsymbol{\lambda_{\mathrm{LPS}}}$ et la durée λ_{LPS} est donnée par :

$$\lambda_{\mathrm{LPS}} = \sum_{j=1}^{k_{\mathrm{LPS}}} \frac{\left(20 + \left(L_{i,\max}\right)_j\right) \times 8}{C}$$

2.2.3 Prise en compte de la variation des délais de bout en bout, construction du LSBF

À l'issue de la première étape de la méthode de construction pessimiste, nous disposons d'un flux périodique de trames entrant dans la mémoire de réception. Une heuristique sur le placement des trames a été formulée, ce qui a conduit à la définition du LSP. Les délais de bout en bout $\Gamma_{i,j}^{R}$ étaient alors supposés constants sur chaque VL_i.

À présent, la variation maximale du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$ est prise en compte pour chaque VL_i, et nous étudions ses implications sur la forme du flux périodique et sur le LSBF.

2.2.3.1 Hyperpériode T_{hyp}

Nous rappelons que le BAG est défini en puissance de 2 tel que :

$$\forall\, i \in [\![1\,;n]\!], \quad \text{BAG}_i = 2^p \text{ où } p \in [\![0\,;7]\!]$$

D'après la définition du BAG, il existe une corrélation entre les valeurs des BAG_i des différents VL_i. Cette corrélation s'écrit :

$$\forall\, k \in [\![1\,;n]\!], \forall\, m \in [\![1\,;n]\!], \quad \text{BAG}_k \geq \text{BAG}_m \Leftrightarrow \text{BAG}_k = \text{BAG}_m \times 2^p \text{ où } p \in [\![0\,;7]\!]$$

Par conséquent, il existe une durée, appelée hyperpériode et notée $\boldsymbol{T_{\text{hyp}}}$, après laquelle les trames sont reçues suivant la même séquence de VL_i. De même que pour un problème d'ordonnancement, l'hyperpériode T_{hyp} est définie comme le **PPCM (Plus Petit Commun Multiple)** parmi tous les BAG_i, ce qui s'écrit :

$$\forall\, i \in [\![1\,;n]\!], \quad T_{\text{hyp}} = \textbf{PPCM}(\text{BAG}_i)$$

$$\forall\, i \in [\![1\,;n]\!], \quad T_{\text{hyp}} = \min(m \in \mathbb{N}) \text{ tel que } \text{BAG}_i \mid m$$

Par la définition du BAG et la corrélation entre les valeurs de BAG, il vient immédiatement :

$$\forall\, i \in [\![1\,;n]\!], \quad \exists\, p \in [\![0\,;7]\!], \quad \frac{\text{BAG}_{\max}}{\text{BAG}_i} = 2^p$$

Donc :

$$\forall\, i \in [\![1\,;n]\!], \quad T_{\text{hyp}} = \max(\text{BAG}_i)$$

$$\boxed{T_{\text{hyp}} = \text{BAG}_{\max}}$$

2.2.3.2 Heuristique pessimiste

Pour chaque VL_i, la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ est donnée. Nous en venons à formuler l'heuristique centrale de la méthode de construction pessimiste, appelée heuristique pessimiste, selon les termes suivants :

Heuristique pessimiste (2.3) : *Pour la construction du LSBF, la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^R_{max}$ est appliquée à l'ensemble des VL_i de la CTRES.*

Cela signifie qu'une fois que nous disposons de l'ensemble des variations maximales des délais de bout en bout $\Delta\Gamma^R_{i,\max}$ pour chaque VL_i, la plus grande de ces variations maximales, notée $\boldsymbol{\Delta\Gamma^R_{\max}}$, est choisie telle que :

$$\boxed{\forall\, i \in [\![1\,;n]\!], \quad \Delta\Gamma^R_{\max} = \max\left(\Delta\Gamma^R_{i,\max}\right)}$$

L'heuristique pessimiste est une première approche, certes un peu grossière, mais qui présente l'avantage d'être facile à mettre en œuvre avec un outil de simulation. Elle permet aussi de calculer une première estimation du pire scénario de réception pour le flux de trames entrant dans la mémoire de réception.

2.2.3.3 Flux fini de trames

La recherche du LSBF sur un flux de trames infini est une tâche complexe et longue en termes d'algorithme et de temps de simulation. Limiter le flux de trames à un nombre fini de trames serait très intéressant pour simplifier la recherche du LSBF. Or, si nous nous appuyons sur la définition de l'hyperpériode associée à la variation maximale du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$, il est possible de justifier une heuristique affirmant que le LSBF peut être construit sur une durée égale à quelques hyperpériodes.

Heuristique 2.4 : *Quelle que soit la CTRES donnée, le LSBF peut être construit sur un flux comprenant un nombre limité de trames et donc sur une durée égale à quelques hyperpériodes.*

Pour justifier l'heuristique 2.4, la Figure 21 représente un flux de trames entrant de trois VLs. Les trames sont représentées sur leur VL respectif mais aussi en projection sur le lien physique pour une durée de réception égale à deux hyperpériodes.

Le placement des trames est tel qu'il respecte l'ensemble des heuristiques et des hypothèses formulées jusqu'alors. L'hypothèse 2.2 sur la longueur des trames est bien visible, ainsi que l'hypothèse 2.4 sur la périodicité de la réception des trames sur chaque VL. L'heuristique 2.2 est appliquée et nous voyons apparaître le LPS dans chacune des hyperpériodes, d'une durée égale à λ_{LPS}. De plus, l'hyperpériode T_{hyp} est placée de sorte que la fin de celle-ci coïncide avec la date de fin de la dernière trame du LPS ($F_{2,4}$).

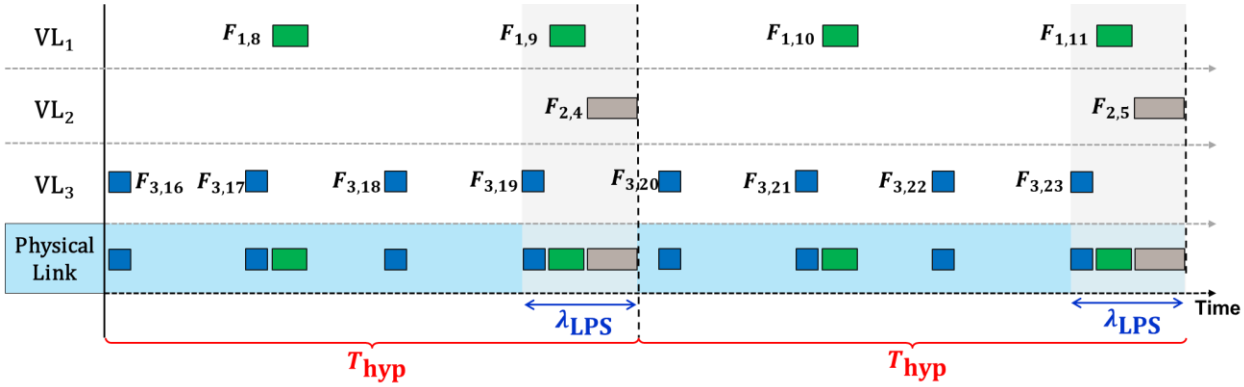


Figure 21 – Flux fini de trames sur deux hyperpériodes pour une CTRES de trois VLs.

Reste à définir la durée en nombre d'hyperpériodes nécessaire à la construction du LSBF. Pour cela, intéressons-nous à la variation maximale du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$, considérée comme étant nulle dans la première étape de la méthode de construction.

À partir de l'heuristique pessimiste, nous pouvons définir deux fenêtres temporelles de taille $\Delta\Gamma^R_{\max}$ autour du LPS comme représenté sur la Figure 22.

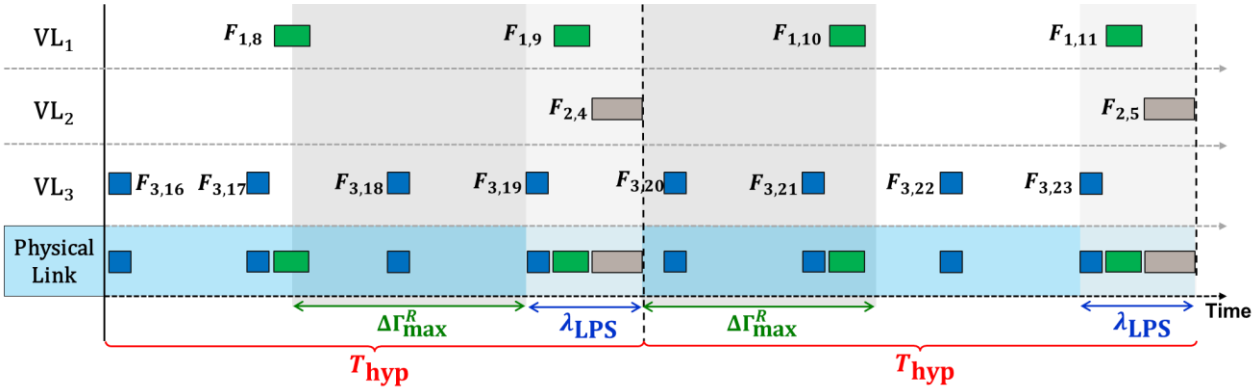


Figure 22 – Fenêtres temporelles $\Delta\Gamma^R_{\max}$ autour du LPS pour une CTRES de trois VLs.

Ainsi, nous postulons que la prise en compte de la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^R_{\max}$ avant et après le LPS peut conduire à la formation du LSBF, d'où l'heuristique 2.5 :

Heuristique 2.5 : *Le LSBF se forme par l'accolement des trames présentes dans le LPS et dans les fenêtres temporelles adjacentes de taille* $\Delta\Gamma^R_{max}$.

En effet, sous l'effet de valeurs particulières de jitters de transmission ou de retards de commutation compris dans $\Delta\Gamma^R_{\max}$, les trames proches du LPS peuvent s'accoler aux trames constituant le LPS et former le LSBF.

Enfin, revenons à la justification de l'heuristique 2.4. Compte tenu de la position de l'hyperpériode sur le flux de trames par rapport au LPS, de l'heuristique pessimiste qui impose de prendre en compte uniquement $\Delta\Gamma^R_{\max}$ comme variation maximale des délais de bout en bout pour l'ensemble des VLs de la CTRES, et de l'heuristique 2.5, le LSBF requiert les trames du LPS et celles des fenêtres adjacentes pour sa formation. Le nombre d'hyperpériodes, noté $\boldsymbol{N_{\text{hyp}}}$, requis peut s'exprimer par une division euclidienne comme suit :

$$N_{\text{hyp}} = E\left(\frac{\Delta\Gamma^R_{\max} + \lambda_{\text{LPS}}}{T_{\text{hyp}}} + 1\right) \times 2$$

Dans l'exemple de la Figure 22, la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^R_{\max}$ et la durée du λ_{LPS} donnent $N_{\text{hyp}} = 2$.

2.2.3.4 Accolement de trames sur le lien physique, LSBF

Dans cette sous-partie, les conditions d'accolement des trames pour la construction du LSBF sont définies. Dans la section précédente, nous avons défini la durée entre la réception de deux trames consécutives sur un VL, notée $\Psi_{i,j}$, ainsi que ses valeurs minimales et maximales $\Psi_{i,\min}$ et $\Psi_{i,\max}$. Le

lien est fait entre la condition d'accolement de trames sur un VL et la variation maximale du délai de bout en bout $\Delta\Gamma^{R}_{i,\max}$. Lorsque nous passons d'un seul VL au lien physique sur lequel transitent les trames de n VLs, la condition d'accolement mérite d'être reprécisée.

Conformément à l'ARINC 664 (3), le délai minimum entre deux trames consécutives reçues au niveau de l'ES de réception est notée $\boldsymbol{\varepsilon}$ et vaut :

$$\varepsilon = \frac{20_{\text{bytes}}}{C}$$

De ce délai vient la condition d'accolement de trames sur le lien physique :

Condition d'accolement de trames sur le lien physique : *Si la durée entre la réception de deux trames sur le lien physique vaut ε, alors ces deux trames sont accolées.*

Ainsi, les trames du LSBF, et plus généralement, les trames du LPS sont séparées d'une durée ε.

La Figure 23 représente le LSBF suivant l'heuristique 2.5. Nous constatons que les trames $F_{1,8}$ et $F_{3,18}$ présentes dans la fenêtre $\Delta\Gamma^{R}_{\max}$ précédant le LPS viennent s'ajouter à celui-ci, et de même pour les trames $F_{3,20}$, $F_{3,21}$ et $F_{1,10}$ présentes dans la fenêtre $\Delta\Gamma^{R}_{\max}$ suivant le LPS. Un LSBF constitué de la séquence de trames $\{F_{1,8}\ ;\ F_{3,18}\ ;\ F_{3,19}\ ;\ F_{1,9}\ ;\ F_{2,4}\ ;\ F_{3,20}\ ;\ F_{3,21}\ ;\ F_{1,10}\}$ est obtenu.

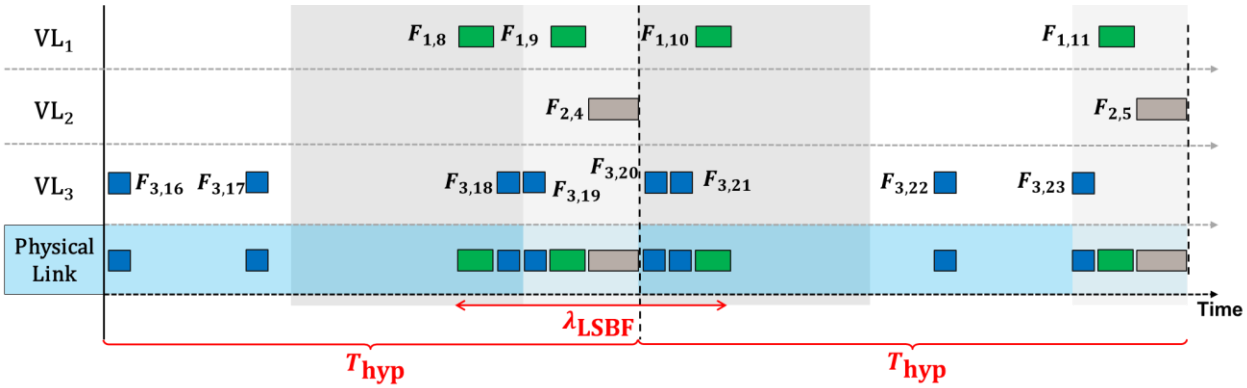


Figure 23 – Représentation du LSBF selon l'heuristique pessimiste pour une CTRES de trois VLs.

Enfin, le LSBF dure $\boldsymbol{\lambda_{\text{LSBF}}}$. Si nous posons $\boldsymbol{k_{\text{LSBF}}}$ comme le nombre de trames constituant le LPS, la durée λ_{LSBF} est donnée par :

$$\lambda_{\text{LSBF}} = \sum_{j=1}^{k_{\text{LSBF}}} \frac{\left(20 + \left(L_{i,\max}\right)_j\right) \times 8}{C}$$

2.2.4 Algorithme de la méthode de construction du LSBF basé sur le modèle pessimiste et outil de simulation

Pour mettre en œuvre notre méthode pessimiste permettant la construction du LSBF sur un flux de trames de n VLs, nous développons un simulateur écrit en langage C. Ce simulateur doit être capable de calculer le LSBF à partir des paramètres d'une CTRES et des variations maximales du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$ pour chaque VL_i. Les paramètres CTRES sont une liste de n VL_i associés chacun à un BAG_i et une longueur de trame maximale $L_{i,\max}$.

La Figure 24 donne un aperçu des algorithmes et de la procédure de calcul.

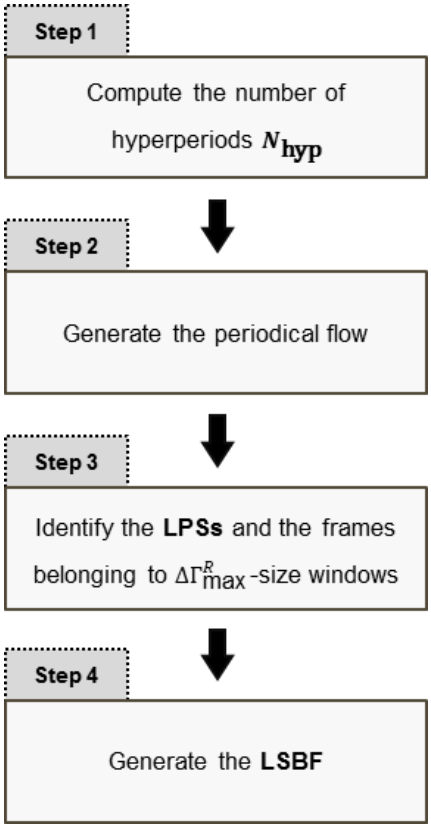


Figure 24 – Algorithme général du modèle pessimiste.

Avant l'étape 1, les paramètres CTRES et les variations maximales du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$ sont téléchargés depuis un fichier texte. Lors de l'étape 1, ils sont utilisés pour allouer la mémoire nécessaire au stockage des résultats en calculant le nombre d'hyperpériode N_{hyp} et de trames nécessaires à la simulation.

L'étape 2 est celle de la génération du flux fini et périodique de trames. Le simulateur génère le flux en calculant toutes les dates de début et de fin de réception des trames et en s'arrangeant pour construire des LPSs pour tous les $\mathrm{BAG}_{\max}$.

L'étape 3 consiste à repérer tous les LPSs dont le nombre est égal au nombre d'hyperpériodes et le simulateur enregistre leurs caractéristiques temporelles. Le LPS « central » est choisi pour la construction du LSBF. Dans le même temps, les conditions d'appartenance des trames aux fenêtres $\Delta\Gamma_{\max}^{R}$ sont testées et les trames élues sont identifiées.

L'étape 4 finale est la génération du LSBF et l'identification des trames appartenant au LSBF. Reste alors à calculer les nouvelles dates d'arrivée des trames appartenant au LSBF. Le simulateur peut alors caractériser le LSBF, en particulier sa durée λ_{LSBF}.

2.3 Application de la méthode pessimiste pour la construction du LSBF sur une CTRES industrielle

Dans cette partie, nous présentons les résultats liés à la construction du LSBF obtenus pour une CTRES industrielle dont nous détaillons les caractéristiques en termes de BAGs et de $L_{i,\max}$.

2.3.1 Exemple de CTRES industrielle

Nos simulations se basent sur une CTRES standard inspirée d'un cas pratique industriel pour réaliser un premier calcul du LSBF. La CTRES choisie comprend cinquante VLs sachant que certaines solutions ESs peuvent gérer un maximum de 512 VLs en réception (55).

Le Tableau 3 indique la répartition des paramètres CTRES en termes de BAG et de longueur maximale de trame $L_{i,\max}$. Le tableau de gauche indique la répartition des VLs pour les valeurs de BAG. Les valeurs élevées de BAG sont privilégiées car elles sont plus fréquemment utilisées que les petites valeurs. Dans notre exemple, plus de 80% des VLs ont des valeurs de BAG réparties entre 16, 32, 64 ou 128 ms. Seulement 10% des VLs ont un BAG de 2,0 ms ou de 4,0 ms et aucun VL n'a de BAG de 1,0 ms. Un BAG de 1,0 ms est rarement utilisé dans des applications industrielles bien qu'il soit autorisé par l'ARINC 664.

BAG (ms)	**Number of VLs**
2	**2**
4	**3**
8	**5**
16	**9**
32	**10**
64	**11**
128	**10**

$L_{i,\max}$ (bytes)	**Number of VLs**
[64 ; 150]	**22**
[151 ; 300]	**13**
[301 ; 600]	**8**
[601 ; 900]	**3**
[901 ; 1200]	**1**
[1201 ; 1518]	**3**

Tableau 3 – Répartition des paramètres CTRES pour une CTRES industrielle de cinquante VLs.

En ce qui concerne la répartition des longueurs maximales de trame $L_{i,\max}$, le tableau de droite indique que les trames courtes (moins de 300 octets) constituent 70% des VLs. Les trames de longueur moyenne (entre 300 et 900 octets) représentent quant à elles 22% de la totalité des VLs. Enfin, les grandes trames (plus de 900 octets) sont plutôt rares dans les configurations industrielles et elles représentent 8% des VLs de la CTRES choisie pour nos simulations.

2.3.2 Résultats de simulation du LSBF

Pour réaliser un calcul de LSBF sur la CTRES industrielle choisie, une valeur pour la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{\max}$ est requise. D'après les mesures de délai de bout en bout de Boyer et al. (53), (54) sur une configuration réseau industrielle, $\Delta\Gamma^{R}_{\max}$ possède une valeur proche de 2,0 ms. Pour nos simulations, nous choisissons une valeur de $\boldsymbol{\Delta\Gamma^{R}_{\max} = 10,0\ \text{ms}}$ afin de s'assurer une marge confortable bien que pessimiste. Cette valeur n'étant qu'un paramètre d'entrée du simulateur, elle est facile à modifier pour conduire de nouvelles simulations.

Le Tableau 4 résume les résultats fournis par le simulateur pour la CTRES de cinquante VLs.

Periodical Flow	N_{hyp}	**346** frames
	Length of the LPS	**50** frames
LSBF	Length	**76** frames
	Duration	**1 887,5** µs
	Size	**22 650** bytes

Tableau 4 – Résultats de simulation (LPS et LSBF) pour une CTRES industrielle de cinquante VLs.

Pour cette CTRES industrielle, la longueur du LPS en nombre de trames est égale au nombre de VLs. De plus, la valeur de $\Delta\Gamma^{R}_{\max}$ ajoute 26 trames au LPS et conduit à un LSBF de 76 trames. Cette séquence de trames dure 1 887,5 µs soit une séquence de 22 650 octets reçue à la vitesse de 100 Mbps. Soulignons que les trames MAC sont considérées et non les trames AFDX puisque le préambule, le SFD et l'IFG sont supprimés avant le stockage des trames dans la mémoire de réception.

2.3.3 Simulations de l'influence des paramètres CTRES sur le LSBF

Dans cette partie, nous analysons l'influence de chaque type de paramètre CTRES (nombre de VLs et BAG) sur le calcul du LSBF. Les résultats de simulation portant sur l'influence de la longueur maximale des trames ayant présenté peu d'intérêt (LSBFs constants en nombre de trames), nous avons fait le choix de ne pas les présenter.

Le but de ces simulations est d'anticiper l'impact d'une évolution de la CTRES sur le LSBF et donc sur le flux de trames entrant de la mémoire de réception. Ainsi, si les résultats de simulation le permettent, il sera possible de prendre en compte des évolutions ciblées de la CTRES dans le dimensionnement de la mémoire de réception.

2.3.3.1 Influence du nombre de VLs

Commençons par étudier le cas de la variation du nombre de VLs comme indiqué sur la Figure 25. La configuration de base est la CTRES industrielle de cinquante VLs. Nous en tirons de nouvelles CTRESs ayant un nombre de VLs qui varie de cinq à cent.

La difficulté de cette mesure est que l'ajout ou le retrait d'un VL_i implique l'ajout ou le retrait d'un couple (BAG_i ; $L_{i,\max}$). Or, le couple (BAG_i ; $L_{i,\max}$) a une influence directe sur le LSBF. Pour pallier à ce problème, nous supposons que l'ensemble des couples (BAG_i ; $L_{i,\max}$) associés aux VLs de la CTRES sont définis de manière à respecter la répartition des valeurs de BAG et de $L_{i,\max}$ de la CTRES industrielle.

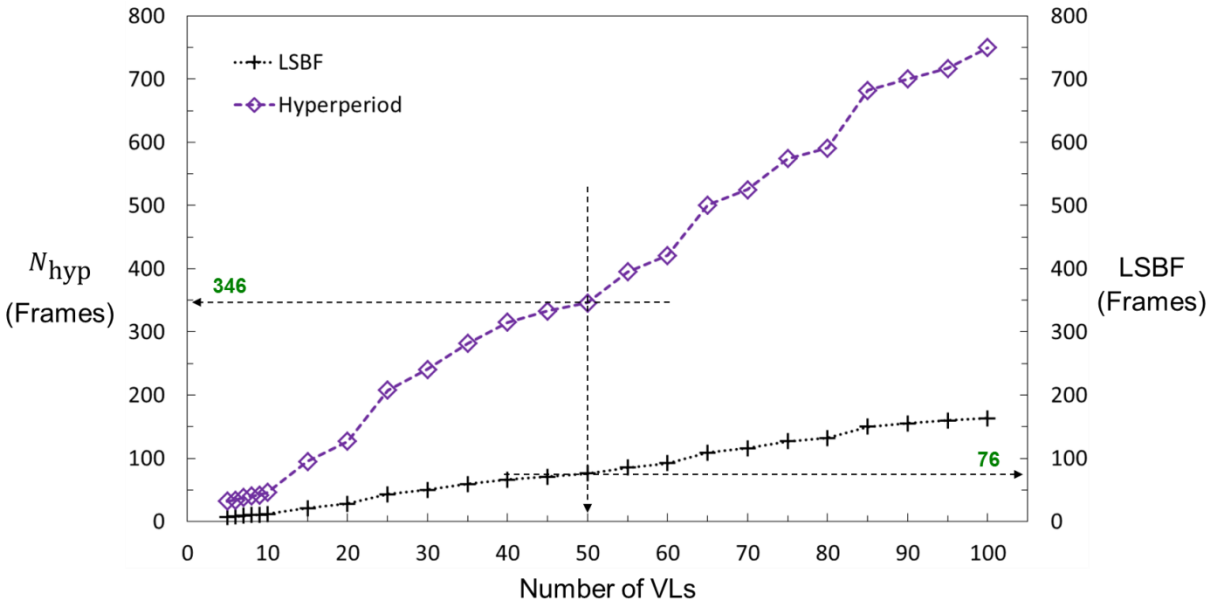


Figure 25 – Nombre de trames dans une hyperpériode N_{hyp} et dans un LSBF en fonction du nombre de VLs.

La Figure 25 montre l'impact d'une augmentation du nombre de VLs sur le nombre de trames de l'hyperpériode N_{hyp} et sur le nombre de trames du LSBF. Ces résultats confirment ce que l'intuition suggère : l'augmentation du nombre de VLs entraîne une augmentation de N_{hyp}, et du nombre de trames constituant le LSBF.

Cet accroissement n'est pas régulier à cause du fait que chaque point est le résultat d'une simulation et non la moyenne de plusieurs simulations sur des CTRESs différentes, ce qui aurait lissé la

courbe obtenue. Dans notre cas, les couples (BAG_i ; $L_{i,\max}$) causent des irrégularités locales sur la courbe.

2.3.3.2 Influence du BAG moyen

Évaluer l'influence du BAG sur le LSBF réclame davantage de soin en raison du caractère non-linéaire du BAG (puissances de 2). Ainsi, nous devons choisir une métrique pour mesurer les variations du BAG sur l'ensemble de la CTRES et la moyenne est un bon candidat pour évaluer les variations de BAG. La CTRES industrielle est reprise et nous réalisons plusieurs simulations en faisant varier le BAG moyen.

La Figure 26 présente l'évolution du nombre de trames de l'hyperpériode N_{hyp} et du nombre de trames du LSBF lorsque le BAG moyen augmente. Malgré une légère croissance des courbes entre 30,0 et 35,0 ms de BAG moyen due à l'imprécision de la mesure réalisée sur une seule CTRES pour chaque point, la tendance générale est à la diminution du nombre de trames.

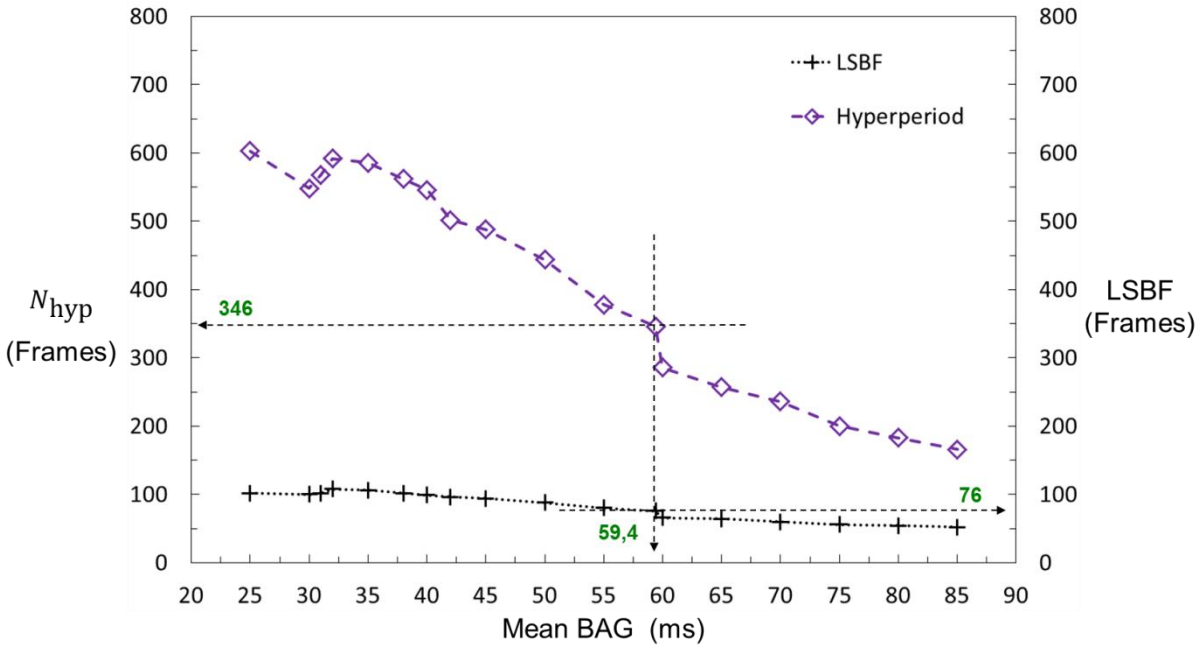


Figure 26 – Nombre de trames dans une hyperpériode N_{hyp} et dans un LSBF en fonction du BAG moyen.

Une augmentation du BAG moyen sur la CTRES implique un flux de trames entrant moins chargé que lorsque le BAG moyen est plus faible. Autrement dit, sur une même unité de temps, le nombre de trames reçues au niveau de l'ES de réception est moins important, d'où la diminution du nombre de trames dans le hyperpériode, et très logiquement, dans le LSBF.

2.4 Estimation du WBF avec le modèle pessimiste

Les développements précédents ont apporté une caractérisation précise du flux de trames entrant dans la mémoire de réception. En particulier, les heuristiques formulées et la méthode de construction du LSBF permettent de donner une estimation du pire scénario de réception. Sans à priori sur la forme du flux de trames sortant, les trames du LSBF s'accumulent dans la mémoire de l'ES de réception et forment le *backlog*. Par conséquent, la taille de la mémoire minimale est égale au WFB.

Afin d'estimer le WFB, nous avons besoin de formaliser les flux de trames entrants et sortants. Le simulateur pourra alors être complété pour fournir des mesures du backlog.

2.4.1 Représentation temporelle des flux dans la mémoire de réception

La formulation du backlog nécessite une formalisation temporelle des flux de trames entrants et sortants de la mémoire de réception. La représentation de ces flux est faite pour des opérations d'écriture ou de lecture de trames dans la mémoire.

2.4.1.1 Modèle en écriture

Une opération d'écriture d'un SBF dans la mémoire consiste en l'alternance de phases d'écriture lors de la réception effective d'une trame, et de phases d'attente entre la fin de la réception d'une trame et le début de la réception de la trame suivante (d'une durée ε). Bien entendu, cette durée peut être plus importante si les trames reçues n'appartiennent pas à un SBF. Lorsque la quantité unitaire de données reçues en fonction du temps est représentée mathématiquement, une fonction en escalier est obtenue comme l'indique la Figure 27.

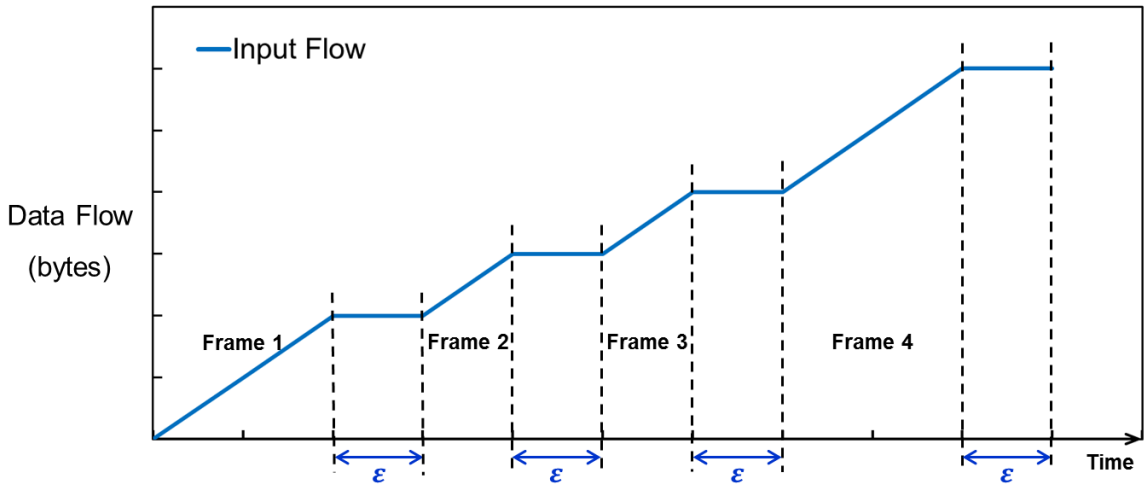


Figure 27 – Modèle en écriture pour la réception d'un SBF.

La courbe présente effectivement une succession de pentes ascendantes et de pentes nulles. Une pente ascendante correspond à une phase d'écriture dans la mémoire, c'est-à-dire à la réception effective d'une trame. Une pente nulle correspond à une phase d'attente dont la durée est égale à ε.

2.4.1.2 Modèle en lecture

Une opération de lecture dans la mémoire consiste en l'alternance de phases de lecture effective lorsqu'une trame est signalée reçue, et de phases d'attente entre la fin de la lecture d'une trame et le début de la lecture de la trame suivante. Une certaine durée s'écoule entre l'instant où une trame est signalée présente dans la mémoire et l'instant où les mécanismes logiciels de la couche UDP + IP déclenchent la lecture de la trame depuis la mémoire de réception. Cette durée est notée $\boldsymbol{\theta_{\mathrm{READ}}}$.

De même que pour l'écriture, la représentation mathématique de la quantité unitaire de données reçues en fonction du temps donne une fonction en escalier comme l'indique la Figure 28.

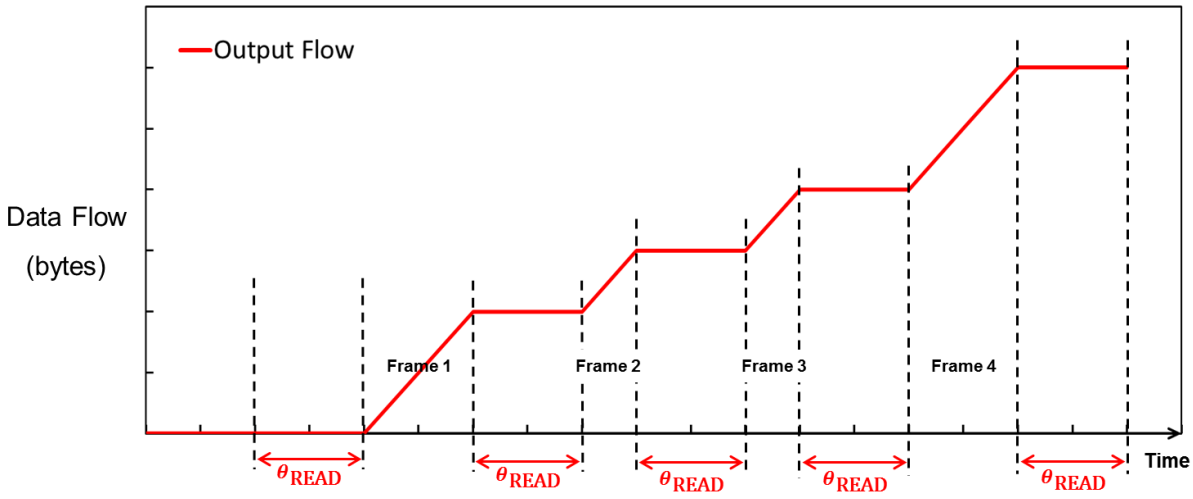


Figure 28 – Modèle en lecture pour la réception d'un SBF.

Cette courbe représente la somme en octets des trames retirées de la mémoire de réception suite à des opérations de lecture en fonction du temps, lors de la réception d'un SBF. L'instant « 0 » coïncide avec celui de la Figure 27, c'est-à-dire le début de l'écriture de la première trame. Une fois la première trame entièrement en mémoire, il s'en suit un délai θ_{READ} avant le début de la lecture. De même, une pente ascendante correspond à une phase de lecture effective dans la mémoire. Nous supposons que la vitesse de lecture est la même que la vitesse d'écriture, donc les pentes ascendantes du flux de sortie ont le même coefficient directeur que celle du flux d'entrée.

Puis, à l'instant où la première trame est entièrement lue, la deuxième trame est présente en mémoire mais un délai θ_{READ} est de nouveau requis avant la lecture de la deuxième trame. Ce cycle de lecture et d'attente reprend jusqu'au vidage complet de la mémoire.

2.4.2 Simulation du backlog

Au simulateur développé pour calculer le LSBF à partir d'un ensemble de paramètres CTRES et de la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{\max}$, nous ajoutons des fonctions afin de simuler l'écriture et la lecture du LSBF en mémoire de façon simultanée. Ces nouvelles fonctions basées sur les modèles présentés dans la partie précédente permettent de calculer le backlog dans la mémoire de réception comme étant la différence entre le flux entrant (écriture) et le flux sortant (lecture) de la mémoire et d'en déduire la taille optimale de la mémoire de réception. En effet, le WFB est valeur maximale du backlog qui apparaît lors de la réception du LSBF (heuristique 2.1).

Pour ce faire, une valeur pour θ_{READ} est choisie comme nouvelle valeur d'entrée de notre simulateur. Nous faisons l'hypothèse que $\boldsymbol{\theta_{\mathrm{READ}} = 10,0\ \mu s}$, ce qui est un délai moyen d'accès en lecture. Pour ce qui est du reste des données en entrée, la CTRES industrielle est reprise.

La Figure 29 montre les flux entrants (en bleu) et sortants (en rouge) de la mémoire de réception lors de la réception du LSBF de 76 trames. Le début de la première trame est reçu à 0 ms et la fin de la dernière trame est reçue à 1,887 5 ms, qui est la durée totale de la réception du LSBF λ_{LSBF}. La mémoire est vide à la date 2,550 4 ms.

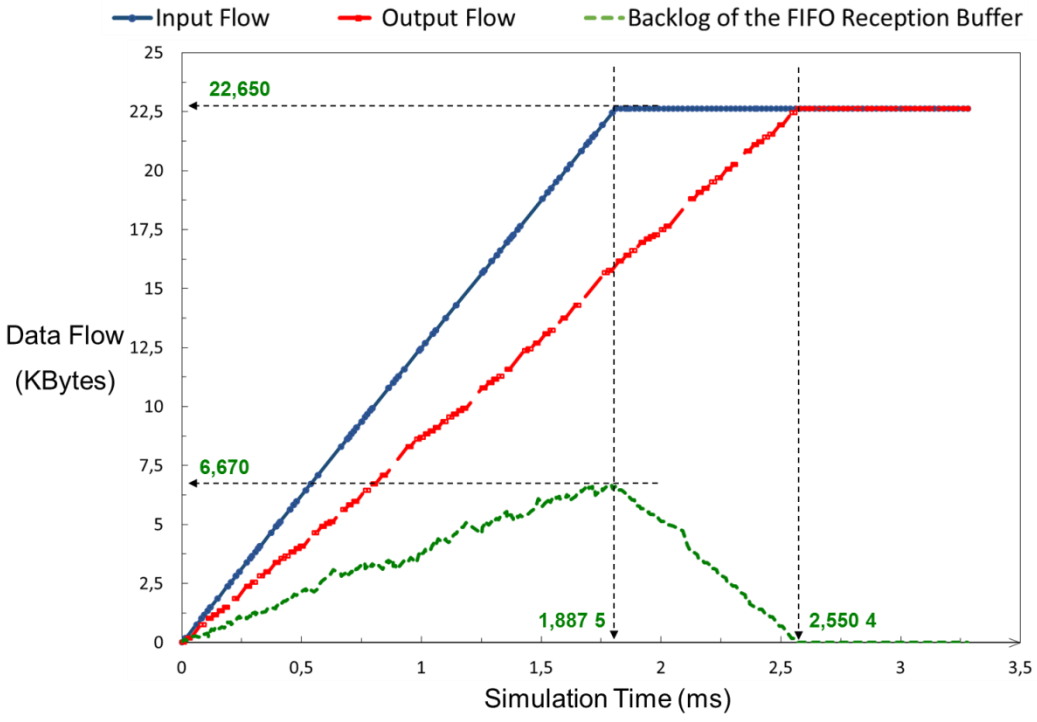


Figure 29 – Représentation des flux de trames entrants et sortants de la mémoire de réception et calcul du WFB pour la réception du LSBF calculé pour une CTRES industrielle de cinquante VLs.

Enfin, le backlog (en vert) est la différence entre les flux d'entrée et de sortie. En d'autres termes, il s'agit de la mesure de l'écart instantané entre les courbes bleues et rouges. Le WFB apparaît à 1,887 5 ms ce qui coïncide avec la fin de réception du LSBF. Le WFB mesuré est de 6 670 octets.

Nous en déduisons immédiatement la taille de la mémoire de réception ES qui doit être d'au moins 6 670 octets à laquelle une marge de sécurité peut être ajoutée. Étant donné que les tailles mémoires sont des multiples de 2^k, la taille réelle de la mémoire serait donc de 8 192 octets.

2.4.3 Simulations de l'influence des paramètres CTRES sur le WFB

Dans cette partie, nous analysons l'influence des paramètres CTRES sur le calcul du WFB en isolant l'influence de chaque paramètre sur le résultat final. Il s'agit de reprendre les simulations mesurant l'influence des paramètres CTRES sur le LSBF et de les étendre au WFB, ceci pour anticiper des évolutions potentielles de la CTRES.

2.4.3.1 Influence du nombre de VLs

Comme précédemment, commençons par étudier une variation du nombre de VLs comme indiqué sur la Figure 30. Le nombre de VLs varie de cinq à cent et nous présentons les tailles en octets du LSBF et du WFB calculées par simulation. De même que lors des précédentes simulations pour le LSBF, nous supposons que l'ensemble des couples (BAG_i ; $L_{i,\max}$) associés aux VLs au sein de la CTRES sont définis de manière à respecter la répartition des valeurs de BAG et de $L_{i,\max}$ présentée pour la CTRES industrielle.

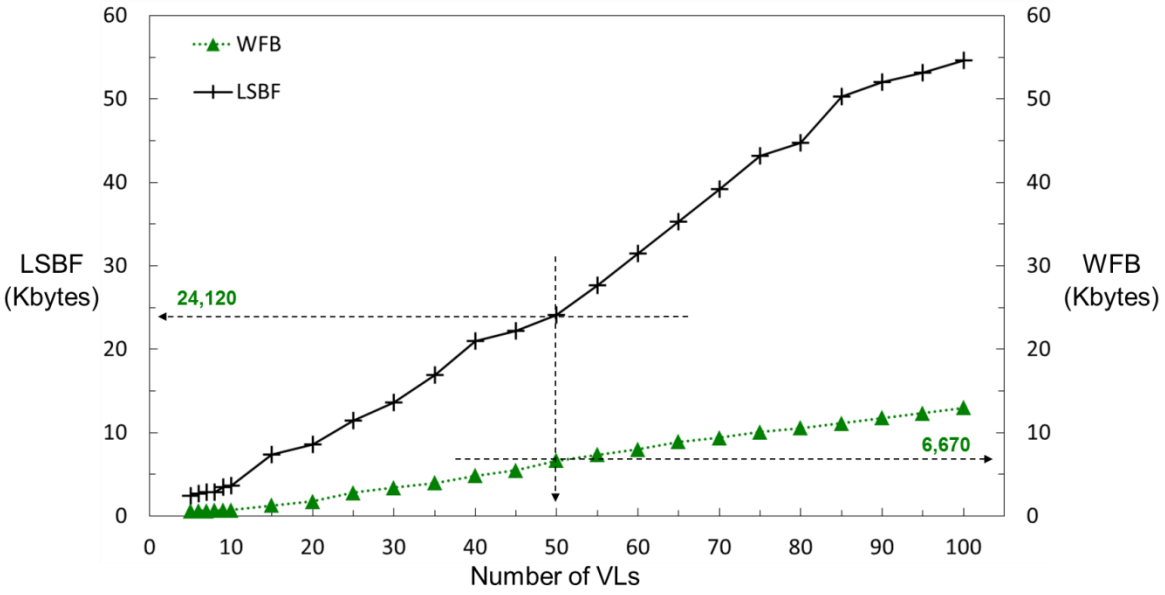


Figure 30 – Taille du LSBF et du WFB en octets en fonction du nombre de VLs.

Sur la Figure 30, la taille du WFB est indiquée sur l'axe vertical de droite en Koctets. Il est mis en regard avec la taille du LSBF indiquée sur l'axe vertical de gauche. De même que pour les mesures précédentes, le WFB et le LSBF augmentent de façon quasi-linéaire lorsque que le nombre de VLs de la CTRES augmente, et ce.

2.4.3.2 Influence du BAG moyen

La Figure 31 montre l'évolution de la taille du WFB et du LSBF en Koctets lorsque le BAG moyen augmente. D'une manière générale, le WFB et le LSBF diminuent avec l'augmentation du BAG moyen. Cet accroissement n'est toutefois pas régulier à cause du fait que chaque point est le résultat d'une seule simulation et non la moyenne de plusieurs simulations sur des CTRESs différentes de même BAG moyen, ce qui aurait abouti à une courbe plus lissée et qui aurait atténué cet accroissement. Dans notre cas, les couples (BAG_i ; $L_{i,\max}$) causent des irrégularités locales sur la courbe.

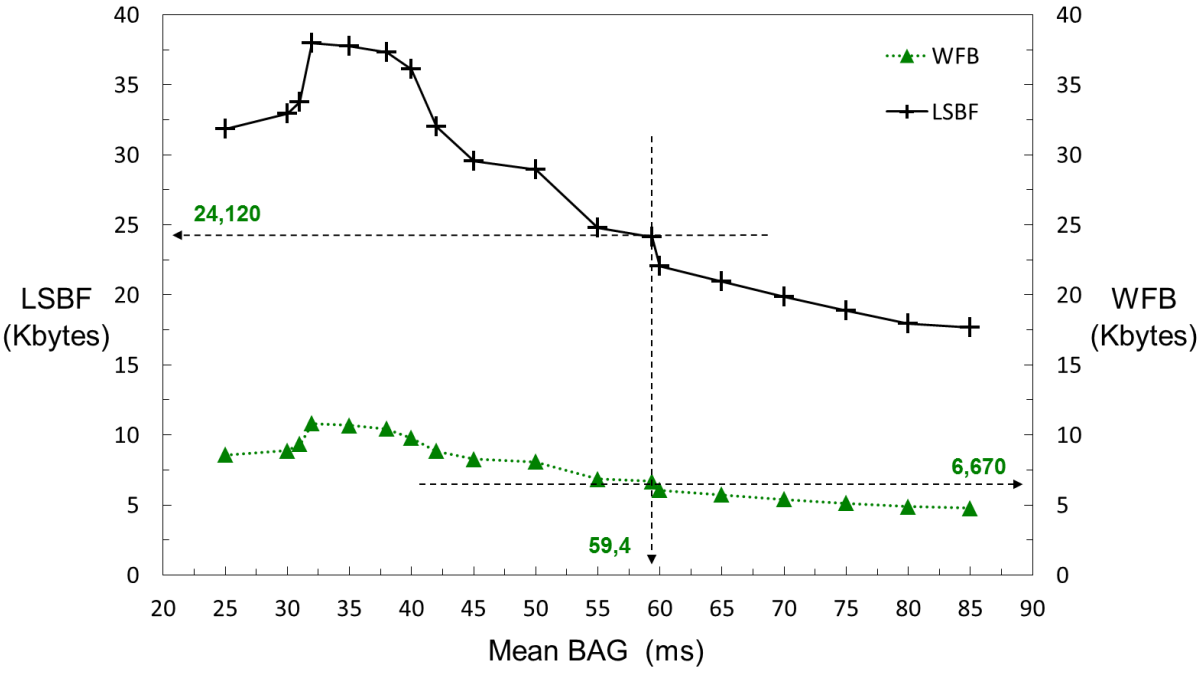


Figure 31 – Taille du LSBF et du WFB en octets en fonction du BAG moyen.

De plus, les tailles du WFB et du LSBF se stabilisent après un BAG moyen d'environ 60 ms, ce qui coïncide avec le BAG moyen de la CTRES industrielle. En effet, à partir de 60 ms de BAG moyen, il devient plus probable que deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$ du même VL_i soient séparées par une durée suffisante telle que les trames $F_{i,j}$ et $F_{i,j+1}$ ne puissent pas contribuer ensemble au LSBF.

Par conséquent, au-delà d'une certaine valeur de BAG moyen suivant les CTRESs, les VLs ne contribuent plus qu'à la hauteur d'une trame au LSBF.

2.4.3.3 Influence du $L_{i,\max}$ moyen

Enfin, nous nous concentrons sur l'influence de la longueur maximale de trame $L_{i,\max}$ circulant sur un VL_i sur le calcul du WFB. De même que pour mesurer les variations du BAG, nous utilisons la moyenne des $L_{i,\max}$ sur toute la CTRES.

Sur la Figure 32, nous étudions la taille du WFB et la taille du LSBF en Koctets en fonction du $L_{i,\max}$ moyen variant sur un intervalle de 200 à 500 octets. La CTRES industrielle présente un $L_{i,\max}$ moyen de 310,2 octets.

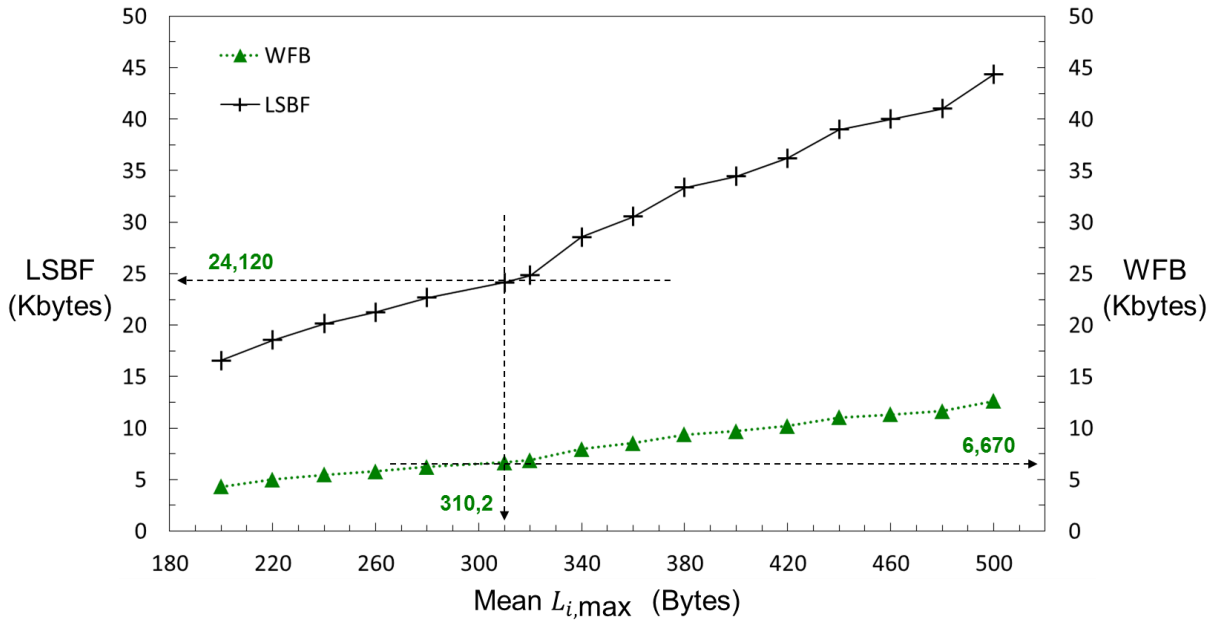


Figure 32 – Taille du LSBF et du WFB en octets en fonction du $L_{i,\max}$ moyen.

Lorsque le $L_{i,\max}$ moyen augmente, la taille du WFB et du LSBF augmentent de manière quasi-proportionnelle.

De plus, l'augmentation du $L_{i,\max}$ moyen ne modifie ni le nombre de trames de l'hyperpériode, ni le nombre de trames appartenant au LSBF. En effet, le $L_{i,\max}$ moyen modifie la taille des trames qui appartiennent déjà au LSBF. Les tailles du WFB et du LSBF sont directement liées à la taille des trames qui constituent le LSBF.

2.5 Conclusion

Le dimensionnement de la mémoire de réception est un point critique pour se conformer aux contraintes aéronautiques portant sur le déterminisme et la fiabilité du réseau AFDX, tout en limitant le gaspillage des ressources mémoires. Dans ce chapitre, nous avons présenté un modèle du flux de trames entrant basé sur une heuristique pessimiste afin d'estimer le pire scénario de réception en terme de backlog dans la mémoire. Nous avons supposé que le WFB est obtenu lors de la réception du LSBF.

À partir de n'importe quelle CTRES, le modèle pessimiste consiste à construire un flux de trames périodique selon des hypothèses défavorables de manière à constituer des LPSs périodiques. La prise

en compte de la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{\max}$ perturbe ce flux périodique de manière à former le LSBF.

L'outil de simulation développé implémente le modèle pessimiste et il fournit une base de résultats pour une CTRES industrielle et des CTRESs dérivées. Nous mettons en évidence l'influence du nombre de VLs, du BAG moyen et de la moyenne des longueurs de trames maximales $L_{i,\max}$ sur le calcul du WFB. Parmi ces trois paramètres, nous observons qu'une variation du nombre de VLs a un impact significatif sur le WFB.

Cependant, l'heuristique pessimiste qui consiste à considérer la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{\max}$ sur l'ensemble des VLs et les hypothèses qui en découlent pour la création du LSBF, pourrait être remplacée par une heuristique plus réaliste en prenant en compte les délais de bout en bout maximum de chaque VL_i. En effet, suivant le chemin suivi dans le réseau AFDX et donc les ESs sources et les commutateurs traversés, les délais de bout en bout maximum $\Gamma^{R}_{i,\max}$ sont variables et peuvent être faibles si la charge rencontrée à la traversée des commutateurs est petite. Par conséquent, le nombre de trames du LSBF pourrait être réduit par la construction d'un modèle plus précis et c'est l'objet du prochain chapitre.

Chapitre 3 : Seconde approche : Modèle basé sur les intervalles de réception

Dans le chapitre précédent, il est établi que la variation du délai de bout en bout $\Delta\Gamma_{i,j}^{R}$ due aux jitters de transmission $J_{i,j}^{T}$ et aux délais de commutation $\theta_{i,j}$ perturbe la périodicité du flux de trames entrant dans la mémoire de réception et conduit à la réception de SBFs. Pour une CTRES donnée, nous avons proposé une méthode de construction basée sur une heuristique pessimiste consistant à prendre en compte la plus grande variation des délais de bout en bout $\Delta\Gamma_{\max}^{R}$ sur l'ensemble des VLs pour construire le LSBF. Puis, à partir d'un modèle d'écriture et de lecture dans la mémoire, le WFB pour une CTRES donnée est déduit ainsi que la taille de la mémoire adéquate suivant la valeur du WFB.

Dans ce chapitre, une seconde approche est présentée, toujours dans le but d'estimer le pire scénario de réception pour le flux entrant. Cette approche a une granularité plus fine que le modèle pessimiste et donc certainement plus proche du pire scénario de réception. Pour cela, nous prenons en compte les plus grandes variations des délais de bout en bout $\Delta\Gamma_{i,\max}^{R}$ de chaque VL_i pris individuellement, et elles sont appliquées sur le flux entrant dans la mémoire en considérant les intervalles de réception de chaque trame.

Ainsi, le modèle du flux entrant évolue avec la notion d'intervalles de réception bornés, ce qui permet de considérer la périodicité de l'envoie des trames sur le réseau AFDX et les délais de bout en bout variables induits par les nœuds du réseau. Le modèle par intervalles permet de caractériser plus précisément un flux de trames comprenant de multiples VLs et de construire des SBFs tant que la charge du lien physique entre le commutateur final et l'ES de réception reste sous 20%.

De même que dans le précédent chapitre, le LSBF est estimé à partir d'une CTRES donnée mais sur la base d'hypothèses plus réalistes. Dans la Section 3.1, nous posons la définition d'un intervalle de réception et la condition d'accolement des trames est redéfinie. La méthode de construction du flux entrant fini sur n VLs est décrite dans la Section 3.2 ainsi que l'algorithme du simulateur codé en C. La Section 3.3 présente des estimations du LSBF obtenues par simulation de la méthode de construction basée sur le modèle par intervalles. La Section 3.4 propose des mesures du WFB pour différents scénarios de réception et une comparaison des deux approches proposées. Enfin, la Section 3.5 conclut ce chapitre.

3.1 Modèle du flux entrant pour un VL

Dans une démarche analogue à celle du chapitre 2, nous présentons le modèle du flux entrant pour un VL basé sur les intervalles de réception des trames. Après la définition de l'intervalle de réception, un flux entrant comprenant un VL est caractérisé.

3.1.1 Intervalle de réception $I_{i,j}$

3.1.1.1 Définition

Le réseau AFDX étant déterministe, le jitter de transmission $J_{i,j}^{T}$ et le délai de commutation $\theta_{i,j}$ sont variables et bornés et cela rejaillit sur le délai de bout en bout $\Gamma_{i,j}^{R}$ qui est donc également variable et borné. Nous avons fait ces mêmes remarques pour mettre en place la première approche, puis les délais de bout en bout ont été définis, leur variabilité et la durée entre la réception de deux trames consécutives, avant d'en venir à décrire la méthode de construction basée en particulier sur une heuristique pessimiste.

Dans cette approche, le constat de la variabilité de $\Delta\Gamma_{i,j}^{R}$ et son encadrement mènent à la définition de l'intervalle de réception, noté $\boldsymbol{I_{i,j}}$, comme suit :

Définition 3.1 *: Les dates de début et de fin de réception de toute trame $F_{i,j}$ au niveau de l'ES de réception appartiennent à un intervalle fini non nul noté $I_{i,j}$.*

Nous posons des notations pour les bornes de l'intervalle $I_{i,j}$. Notons $\boldsymbol{\tau_{i,j}^{s}}$ la date la plus précoce de réception du premier bit de la trame $F_{i,j}$, et $\boldsymbol{\tau_{i,j}^{e}}$ la date la plus tardive de réception totale de la trame $F_{i,j}$ telles que:

$$\forall\, F_{i,j} \in \mathrm{VL}_i, \qquad \exists\, I_{i,j} = \left[\tau_{i,j}^{s}\,;\, \tau_{i,j}^{e}\right] \text{ tel que } F_{i,j} \in I_{i,j}$$

La taille maximale des intervalles $I_{i,j}$ est liée à la valeur maximale de la variation des délais de bout en bout $\Delta\Gamma_{i,\max}^{R}$ telle que :

Propriété 3.1 : $$\forall\, i \in [1\,;n], \quad \tau_{i,j}^{e} - \tau_{i,j}^{s} = \Delta\Gamma_{i,\max}^{R}$$

Enfin, nous exprimons la notion d'intervalle de réception vis-à-vis de l'hypothèse 2.1, ce qui donne la propriété suivante :

Propriété 3.2 : $$\forall\, i \in [1\,;n], \quad \tau_{i,j+1}^{s} = \tau_{i,j}^{s} + \mathrm{BAG}_i$$

La particularité de cette approche consiste à considérer d'abord les intervalles de réception bornés et leur placement respectif, puis le placement des trames dans ces intervalles pour évaluer le LSBF, plutôt que les trames directement comme dans le chapitre 2.

3.1.1.2 Condition d'accolement de trames

Dans le cadre de la définition des intervalles de réception, il est nécessaire de redéfinir la notion d'accolement de trames.

La Figure 33 représente les intervalles de réception de trois trames $F_{i,j}$, $F_{i,j+1}$ et $F_{i,j+2}$ sur un seul VL_i de la CTRES. Les intervalles $I_{i,j}$, $I_{i,j+1}$ et $I_{i,j+2}$ sont les fenêtres de réception respectives des trames $F_{i,j}$, $F_{i,j+1}$ et $F_{i,j+2}$. Les dates $\tau^s_{i,j}$, $\tau^s_{i,j+1}$ et $\tau^s_{i,j+2}$ sont respectivement les dates de début des intervalles $I_{i,j}$, $I_{i,j+1}$ et $I_{i,j+2}$. Ces dates correspondent aux dates les plus précoces de réception du début des trames $F_{i,j}$, $F_{i,j+1}$ et $F_{i,j+2}$.

De même, les dates $\tau^e_{i,j}$, $\tau^e_{i,j+1}$ et $\tau^e_{i,j+2}$ sont respectivement les dates de fin des intervalles $I_{i,j}$, $I_{i,j+1}$ et $I_{i,j+2}$. Ces dates correspondent aux dates les plus tardives de réception totale des trames $F_{i,j}$, $F_{i,j+1}$ et $F_{i,j+2}$.

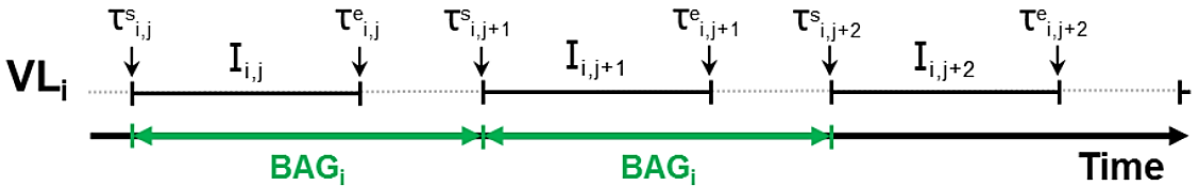


Figure 33 – Intervalles de réception pour un VL pour $\varepsilon < \tau^s_{i,j+1} - \tau^e_{i,j}$.

Dans ce premier scénario, quelle que soit la position des trames à l'intérieur de leur intervalle, un accolement de celles-ci n'est pas possible. En effet, dans ce scénario, les dates $\tau^s_{i,j+1}$ et $\tau^e_{i,j}$ sont telles que :

$$\forall\, i \in [0\,;n], \quad \tau^s_{i,j+1} - \tau^e_{i,j} > \varepsilon$$

Où ε est la plus courte durée entre deux trames consécutives $F_{i,j}$ et $F_{i,j+1}$, définie dans le précédent chapitre.

La Figure 34 présente le deuxième scénario possible lorsque la durée entre $\tau^s_{i,j+1}$ et $\tau^e_{i,j}$ est telle que :

$$\forall\, i \in [0\,;n], \quad \tau^s_{i,j+1} - \tau^e_{i,j} = \varepsilon$$

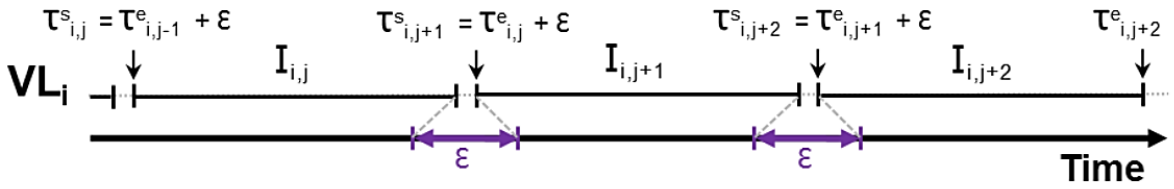


Figure 34 – Intervalles de réception pour un VL pour $\varepsilon = \tau^s_{i,j+1} - \tau^e_{i,j}$.

Dans ce scénario, le placement de trames sur les extrémités de deux intervalles successifs conduit à la formation d'un SBF de deux trames sur le VL_i.

Enfin, la Figure 35 illustre l'ultime scénario lorsque la durée entre $\tau^s_{i,j+1}$ et $\tau^e_{i,j}$ est telle que :

$$\forall\, i \in [0\,;n], \quad \tau^s_{i,j+1} - \tau^e_{i,j} < \varepsilon$$

Dans ce dernier scénario, la mesure de la différence entre les dates $\tau^s_{i,j+1}$ et $\tau^e_{i,j}$ peut même être négative. Un SBF de deux trames peut se produire dans les mêmes conditions que précédemment.

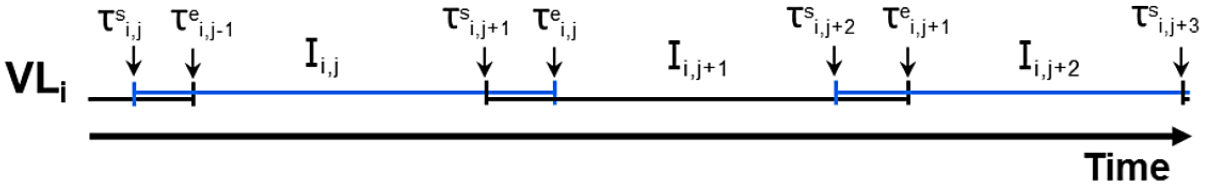


Figure 35 – Intervalles de réception pour un VL pour $\varepsilon > \tau^s_{i,j+1} - \tau^e_{i,j}$.

3.2 Méthode de construction du flux entrant sur n VLs basée sur le modèle par intervalles

Sur la base de ces premières définitions, nous proposons une méthode de construction du flux entrant sur n VLs basée sur le modèle par intervalles. De même que pour la première approche pessimiste, l'évaluation de la taille de la mémoire de réception ES, nous devons considérer le flux entrant dans un scénario de réception conduisant au WFB dans la mémoire.

Pour construire ce scénario pour n VLs, les hypothèses et heuristiques de la première approche sont reprises, excepté l'heuristique pessimiste. Elle est remplacée par une heuristique plus réaliste. Ensuite, nous mettons en œuvre une méthode de construction basée sur les intervalles de réception pour estimer le pire scénario de réception. Cette méthode comprend quatre étapes présentées dans les parties suivantes.

3.2.1 Heuristiques et hypothèses

Les heuristiques et les hypothèses au sujet du LSBF et de la charge du lien physique entre le commutateur final et l'ES de réception sont reprises et les énoncés sont rappelés.

3.2.1.1 LSBF en réception

Heuristique 2.1 : *Le scénario de réception conduisant au backlog maximal (WFB) dans la mémoire de réception ES correspond à la réception du LSBF sur le flux entrant dans la mémoire*.

L'heuristique 2.1 s'accompagne d'une hypothèse de transmission sur le réseau pour une configuration réseau donnée, ce qui revient énoncer l'hypothèse 2.1 :

Hypothèse 2.1 : *Tous les ESs sources du réseau transmettent leurs trames BAG-périodiquement sur leurs VLs respectifs.*

La dernière hypothèse porte sur la longueur des trames $F_{i,j}$ circulant sur chaque VL_i, toujours dans l'optique de maximiser la charge sur le réseau pour une configuration réseau donnée :

Hypothèse 2.2 : *Pour un VL_i donné, la longueur de toutes les trames de ce VL est égale à la longueur maximale $L_{i,\max}$.*

3.2.1.2 Charge du lien physique

La validité de la méthode de construction repose sur le fait que les paramètres de la CTRES (nombre de VLs, BAGs et $L_{i,\max}$) conduisent à une charge limitée sur le lien physique entre le commutateur final et l'ES de réception.

Hypothèse 2.3 : *La charge du lien physique est limitée à 20%.*

3.2.1.3 Périodicité des intervalles

Dans cette seconde approche, nous considérons la périodicité des intervalles du flux entrant, et non directement les trames du flux entrant comme dans la première approche. L'hypothèse 2.4 évolue donc vers une nouvelle formulation comme suit :

Hypothèse 3.4 : *Les intervalles de réception entrants sont périodiques sur chaque VL_i de la CTRES.*

3.2.1.4 Prise en compte de la variation maximale du délai de bout en bout $\Delta\Gamma^{R}_{i,\max}$ pour chaque VL_i

Dans la première approche, la construction du LSBF repose sur l'heuristique pessimiste. Il s'agit de prendre en compte uniquement la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^R_{\max}$ et de l'appliquer à l'ensemble des VLs de la CTRES.

À présent, la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ est prise en compte pour chaque VL_i et c'est le point essentiel sur lequel s'appuie cette seconde approche. Pour une CTRES quelconque, nous construisons un LSBF en prenant donc une granularité plus fine, ce qui devrait donner une estimation du pire scénario de réception au sens de l'heuristique 2.1 plus proche de la réalité.

La considération de la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ justifie la définition des intervalles de réception $I_{i,j}$ et nous avons vu que la taille de l'intervalle $I_{i,j}$ est directement liée à la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ pour un VL_i.

3.2.2 Nombre fini d'intervalles N_I

Limiter le flux de trames entrant à un nombre fini de trames simplifie la recherche du LSBF. Or, si nous nous appuyons sur la définition de l'hyperpériode associée à la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$, les termes de l'heuristique 2.4 peuvent être repris et adaptés au modèle par intervalles, ce qui donne :

Heuristique 3.1 : *Quelle que soit la CTRES donnée, le LSBF peut être construit sur un flux comprenant un nombre limité de trames et donc sur un nombre limité d'intervalles N_I.*

La première étape de la méthode de construction basée sur le modèle par intervalles est le choix du nombre d'intervalles $\boldsymbol{N_I}$ sur lequel porte la méthode. Le nombre d'intervalles N_I à considérer pour la recherche du LSBF est un facteur essentiel : il faut qu'il soit à la fois assez grand pour garantir la construction du LSBF et assez petit pour ne pas engendrer des temps de simulation très longs.

L'heuristique 3.1 se justifie en s'appuyant partiellement sur l'heuristique pessimiste 2.3 de la première approche. En effet, nous pouvons réutiliser les définitions de l'hyperpériode T_{hyp} et de la variation maximale du délai de bout en bout pour l'ensemble des VLs de la CTRES $\Delta\Gamma^R_{\max}$ pour définir deux fenêtres adjacentes dans lesquelles la formation du LSBF est garantie. La taille d'une fenêtre est au moins égale à une hyperpériode, sauf dans le cas où $\Delta\Gamma^R_{\max}$ est plus grand que la durée d'une hyperpériode T_{hyp}. Nous postulons que le LSBF apparaît sur une telle durée.

Ainsi, l'heuristique 3.1 peut s'exprimer en nombre d'hyperpériodes, noté $\boldsymbol{N_{\mathrm{hyp}}}$, requis pour construire le LSBF comme suit :

$$N_{\mathrm{hyp}} = E\left(\frac{\Delta\Gamma^R_{\max}}{T_{\mathrm{hyp}}} + 1\right) \times 2$$

En nombre d'intervalles N_I, cela donne, d'après la définition de l'hyperpériode :

$$N_I = \sum_{i=1}^{n} \frac{T_{\text{hyp}}}{\text{BAG}_i} \times N_{\text{hyp}}$$

Pour illustrer la méthode de construction basée sur les intervalles de réception, nous reprenons l'exemple de la configuration de trois VLs sur la Figure 36. Les valeurs de BAG et la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^R_{\max}$ donnent $N_{\text{hyp}} = 2$.

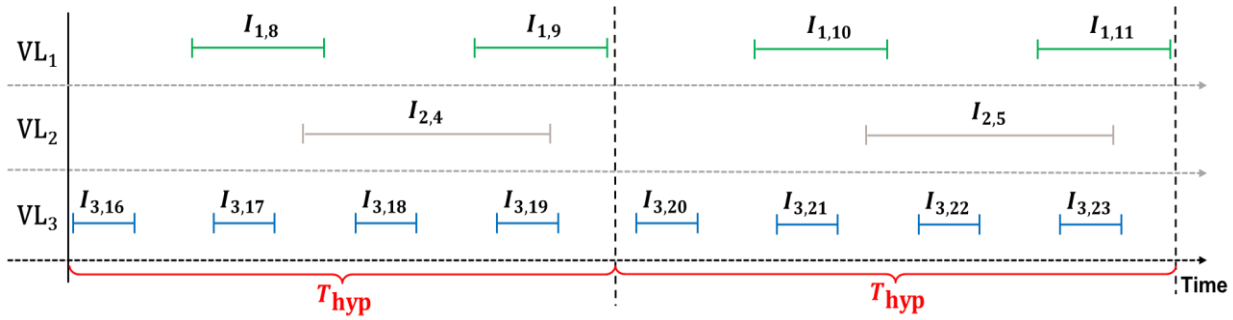


Figure 36 – Intervalles de réception pour trois VLs avec $N_{\text{hyp}} = 2$.

La Figure 36 représente bien l'hypothèse 3.4 sur la périodicité des intervalles et l'heuristique 3.1 sur la limitation du nombre d'intervalles considérés.

La suite de la réflexion porte sur la position relative des intervalles les uns par rapport aux autres puisqu'ils sont pour l'instant placés de façon aléatoire.

3.2.3 Position relative des intervalles

Une fois le nombre d'intervalles N_I fixé, la deuxième étape de la méthode de construction du LSBF consiste à régler la position relative des intervalles de réception. Les positions relatives sont les placements des intervalles $I_{i,j}$ sur chaque VL_i les uns par rapport aux autres. Suivant l'offset temporel du placement du premier intervalle sur chaque VL_i, les intervalles du VL_i ne sont pas placés de la même manière par rapport aux autres.

Reprenons l'exemple de la Figure 36. Les intervalles $I_{2,4}$ et $I_{3,18}$ se chevauchent si nous les projetons sur un seul axe temporel, ce qui fait qu'un placement particulier des trames $F_{2,4}$ et $F_{3,18}$ peut conduire à l'apparition d'un SBF puisque la condition d'accolement ε entre les trames pourrait être respectée.

En revanche, les intervalles $I_{3,16}$ et $I_{1,8}$ sont placés tels que $\tau^s_{1,8} - \tau^e_{3,16} > \varepsilon$. Donc leur trame ne peut pas former un SBF à elles deux. Maintenant, si l'offset du VL_1 est tel que les intervalles du VL_1 sont décalés sur la droite, à partir de l'instant où $\tau^s_{1,8} - \tau^e_{3,16} = \varepsilon$, les trames $F_{1,8}$ et $F_{3,16}$ peuvent se retrouver accolées. L'enjeu est donc d'identifier les positions relatives conduisant à un maximum de trames accolées.

Une première démarche porte à considérer toutes les positions relatives de façon exhaustive. Cependant, le nombre de placements devient extrêmement grand à mesure que le nombre de VLs augmente parce que le nombre de placements dépend du pas d'incrémentation entre deux positions. Le pas d'incrémentation doit être suffisamment petit pour assurer une bonne précision dans la recherche du LSBF. Ceci conduit à des simulations irréalisables en raison de leur temps de calcul très élevé.

Pour résoudre ce problème, nous proposons de limiter le nombre de combinaisons en étudiant les positions relatives qui sont le plus favorable à la construction du LSBF, d'où l'heuristique 3.2 :

Heuristique 3.2 : *Les positions relatives des intervalles étant les plus favorables à la construction du LSBF sont celles qui correspondent à des positions où le nombre de « chevauchements » entre les intervalles de différents VL_i est maximum.*

En outre, quel que soit le nombre de VLs, il existe un positionnement des intervalles conduisant à un SBF comprenant un nombre de trames au moins égale au nombre n de VLs. Ce SBF constitue une référence pour le calcul du LSBF comme le montre la Figure 37.

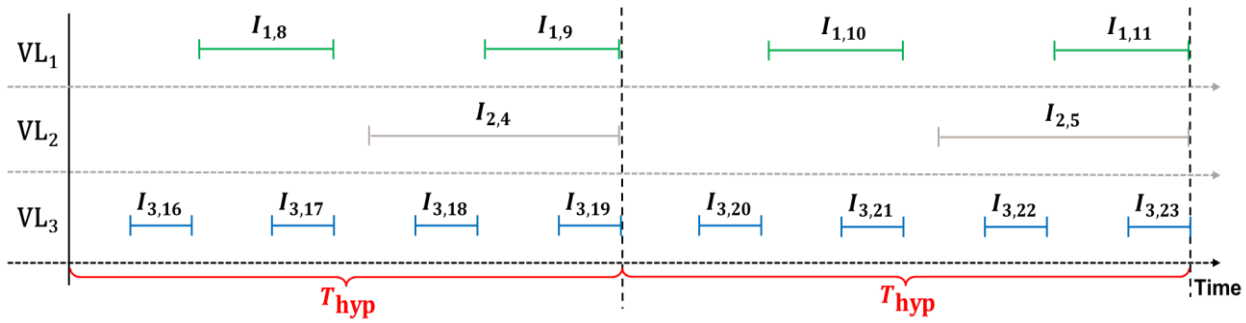


Figure 37 – Position de référence des intervalles de réception pour trois VLs.

Les intervalles de chacun des VLs ont été décalés de façon à faire coïncider une date $\tau_{i,j}^{e}$ avec la date de fin de la première hyperpériode. La stratégie de placement repose sur ce placement de référence conduisant généralement à un SBF au moins égal à n, puis à explorer les placements autour de ce placement initial avec un pas d'incrémentation faible et rebouclage de l'algorithme de recherche.

Nous prenons 20 placements pour chaque VL_i, pour une variation de **2,0 ms** et pour un pas d'incrémentation de **100 µs**, depuis la position de référence, en décalant les intervalles vers la gauche. Cela permet de rechercher des placements favorables à la construction du LSBF autour du placement de référence.

3.2.4 Chaînes d'intervalles $C_{i,j}$

Pour chaque placement d'intervalles, nous devons rechercher le LSBF. Pour ce faire, nous commençons par identifier les intervalles qui peuvent conduire à la formation d'un SBF. Les *chaînes*

d'intervalles, notées $\boldsymbol{C_{i,j}}$, sont alors définies. Une chaîne d'intervalles $C_{i,j}$ est formée d'intervalles dont les trames peuvent former un SBF.

Pour construire toutes les chaînes $C_{i,j}$, chaque intervalle $I_{i,j}$ est considéré comme l'*intervalle central* de la chaîne $C_{i,j}$, puis nous recherchons tous les intervalles proches au sens de la condition d'accolement comme le montre la Figure 38.

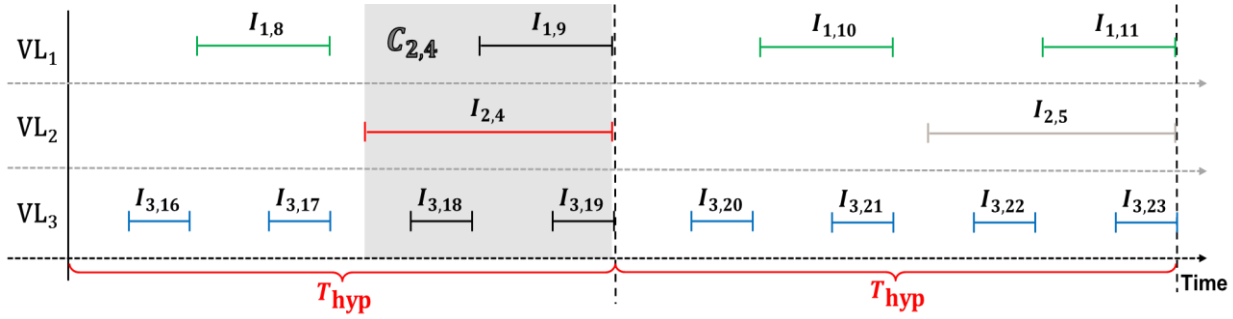


Figure 38 – Chaîne $C_{2,4}$ construite à partir de l'intervalle central $I_{2,4}$.

La Figure 38 montre la chaîne $C_{2,4}$ créée à partir de l'intervalle central $I_{2,4}$ (en rouge). Dans ce cas simple, les intervalles $I_{1,9}$, $I_{3,18}$ et $I_{3,19}$ remplissent les conditions d'accolement et font donc partis de la chaîne $C_{2,4}$. Cela dit, nous prenons aussi en compte le cas où un intervalle ne satisfaisant pas *directement* les conditions d'accolement pourrait tout de même appartenir à une chaîne d'intervalles. Il faut alors vérifier si la trame placée sur cet intervalle peut s'accoler au SBF formé à partir de l'intervalle central via d'autres intervalles appartenant à la chaîne $C_{i,j}$.

3.2.5 Placement des trames

L'étape finale de la méthode de construction du LSBF consiste à placer les trames dans les intervalles de chaque chaîne $C_{i,j}$, puis à déterminer le LSBF en comparant tous les SBFs obtenus. Nous définissons un SBF au sens de l'algorithme de recherche du LSBF comme une liste d'identifiants de trames (i.e. les couples $(i\,;j)$). Avec cet identifiant, nous pouvons identifier le VL_i auquel appartient la trame, et donc sa durée de réception δ_i^R et sa longueur maximale $L_{i,\max}$.

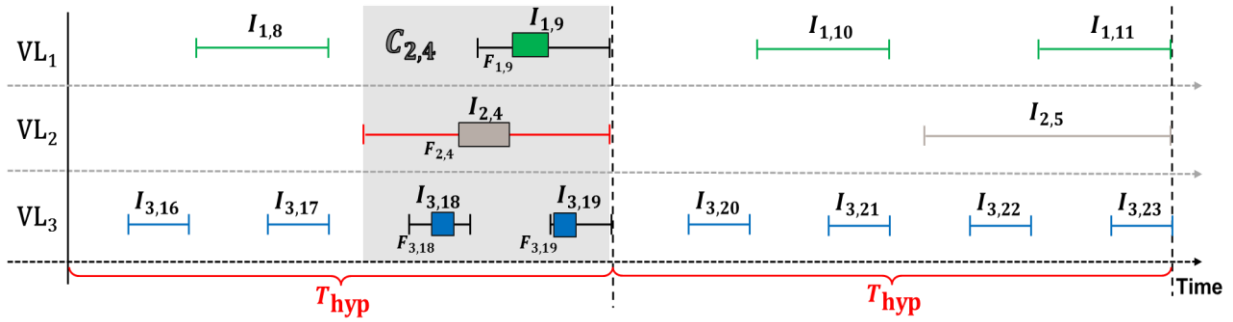


Figure 39 – Placement des trames sur la chaîne $C_{2,4}$ construite à partir de l'intervalle central $I_{2,4}$.

La Figure 39 montre le placement des trames sur leur intervalle respectif. Le SBF ainsi formé est composé de quatre trames, le nombre maximum de trames sur la chaîne $C_{2,4}$.

De plus, le placement d'une trame respecte certains critères pour limiter le nombre de placements de la même manière que pour le placement des intervalles. La position initiale d'une trame est la clef d'un placement conduisant au plus grand SBF pour une chaîne $C_{i,j}$. Les positions privilégiées pour la première trame sont les positions extrêmes : au tout début ou en toute fin d'intervalle. Ensuite, un SBF « primaire » est construit en commençant d'un intervalle de la chaîne jusqu'à l'intervalle central $I_{i,j}$. Les autres trames de la chaîne $C_{i,j}$ sont ajoutées au SBF primaire si la condition d'accolement peut être respectée.

Une fois que tous les plus grands SBFs sont calculées pour chaque chaîne $C_{i,j}$, le LSBF est élu comme étant la plus longue séquence parmi toutes celles trouvées sur les chaînes $C_{i,j}$. Notons qu'il est possible d'obtenir plusieurs LSBFs de même longueur en termes de nombres de trames accolées. Dans ce cas, le plus grand LSBF en octets est retenu.

3.2.6 Algorithme de la méthode de construction du LSBF basée sur les intervalles de réception et outil de simulation

Pour mettre en œuvre notre méthode basée sur les intervalles de réception permettant la construction du LSBF sur un flux de trames de n VLs, un simulateur écrit en langage C est réalisé. Mises à part les fonctions d'initialisation, nous avons dû développer un jeu complet de nouvelles fonctions par rapport au simulateur employé dans la première approche.

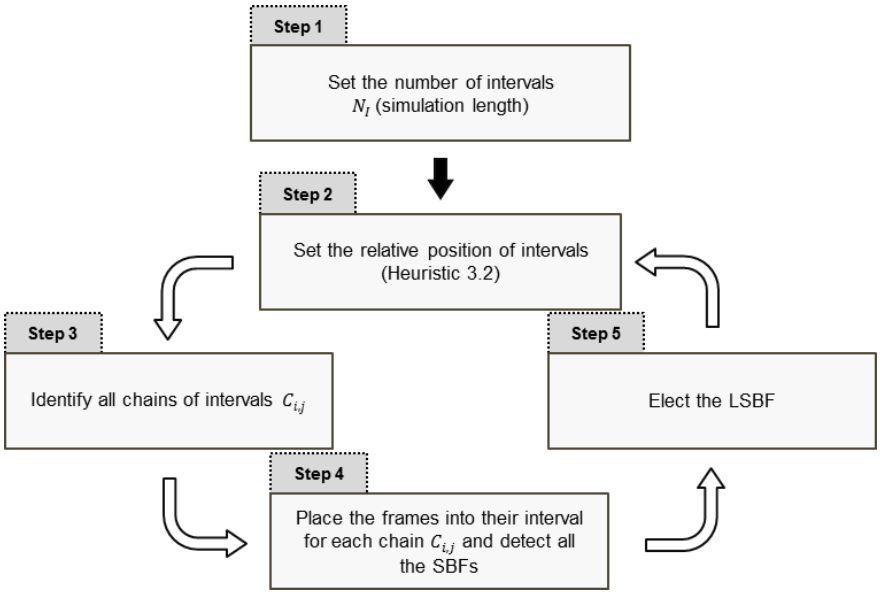


Figure 40 – Algorithme général de la méthode de construction du LSBF basée sur le modèle par intervalles.

Le but de ce simulateur est le même que pour la première approche : il doit calculer le LSBF à partir des paramètres d'une CTRES et des variations maximales des délais de bout en bout $\Delta\Gamma^{R}_{i,\max}$ pour chaque VL_i. Les paramètres CTRES sont une liste de n VL_i associés chacun à leur BAG_i et à la longueur de trame maximale $L_{i,\max}$. La Figure 40 propose un aperçu de l'algorithme et de la procédure de calcul.

L'algorithme comprend quatre phases principales qui sont : le calcul du nombre d'intervalles appartenant à chaque VL_i, le placement des intervalles $I_{i,j}$ sur chaque VL_i, la recherche de chaînes de chevauchement, et enfin le placement des trames sur leur intervalle respectif.

Avant l'étape 1, le démarrage est le même que pour le précédent simulateur : les paramètres CTRES et les variations maximales des délais de bout en bout $\Delta\Gamma^{R}_{i,\max}$ sont téléchargés depuis un fichier texte. Lors de l'étape 1, ils sont utilisés pour allouer la mémoire nécessaire au stockage des résultats en calculant le nombre d'hyperpériodes N_{hyp} et de trames nécessaires à la simulation.

L'étape 2 est consacrée au placement des intervalles $I_{i,j}$ sur chaque VL_i, c'est-à-dire par le calcul des dates $\tau^{s}_{i,j}$ et $\tau^{e}_{i,j}$ suivant l'heuristique 3.2.

Une fois le premier placement d'intervalles choisi, les chaînes d'intervalles $C_{i,j}$ qui pourraient induire un SBF sont identifiées de manière itérative lors de l'étape 3.

L'étape 4 consiste simplement à élire le plus grand SBF avant de reboucler sur l'étape 2 pour un deuxième placement. À la fin de la boucle « étapes 2, 3 et 4 », le LSBF de la CTRES est choisi comme le plus grand SBF détecté et ses caractéristiques temporelles sont choisies.

3.3 Évaluation du LSBF par simulation

Dans cette section, nous présentons les résultats de notre méthode de construction du LSBF implémentée avec un simulateur codé en C. L'influence des paramètres CTRES sur le LSBF est mesurée pour un grand nombre de CTRESs comme pour la première approche.

Comme les paramètres CTRES ont une influence sur le flux de trames entrant dans la mémoire de réception, nous observons les impacts de variation des paramètres sur le calcul du LSBF. Le but est de déterminer le poids de chaque paramètre CTRES sur le calcul du LSBF.

3.3.1 Variation du nombre de VLs

Commençons par l'influence du nombre de VLs sur le LSBF. Pour cela, nous réalisons une série de simulations où à la fois le nombre de VLs et le BAG moyen de tous les VLs de la CTRES varient.

Les résultats sont représentés sur la Figure 41 sous la forme d'une nappe dont les points indiquent la valeur du LSBF pour un couple (nombre de VLs ; BAG moyen). Chaque point est construit à partir d'une seule CTRES et c'est pourquoi nous pouvons observer des irrégularités locales sur la nappe. En

effet, le passage d'un jeu de paramètres CTRES à un autre peut créer des variations un peu abruptes sur la mesure du LSBF.

De plus, la moyenne est utilisée pour le BAG comme paramètre de position. Sur la Figure 41, la dispersion des valeurs de BAG autour de la moyenne est faible. Les autres paramètres CTRES sont maintenus à un niveau constant pour toutes les mesures : le $L_{i,\max}$ moyen est fixé à 1 500 octets pour un écart-type nul et le $\Delta\Gamma^{R}_{i,\max}$ moyen est fixé à 750 µs pour un écart-type égal à 50. Le $\Delta\Gamma^{R}_{i,\max}$ moyen est proche de sa valeur maximale autorisée par l'ARINC 664 car nous voulons amplifier autant que possible la valeur du LSBF. En effet, des valeurs plus faibles pour le $L_{i,\max}$ moyen impliqueraient un LSBF plus petit et l'influence du nombre de VLs sur le LSBF serait plus difficile à observer.

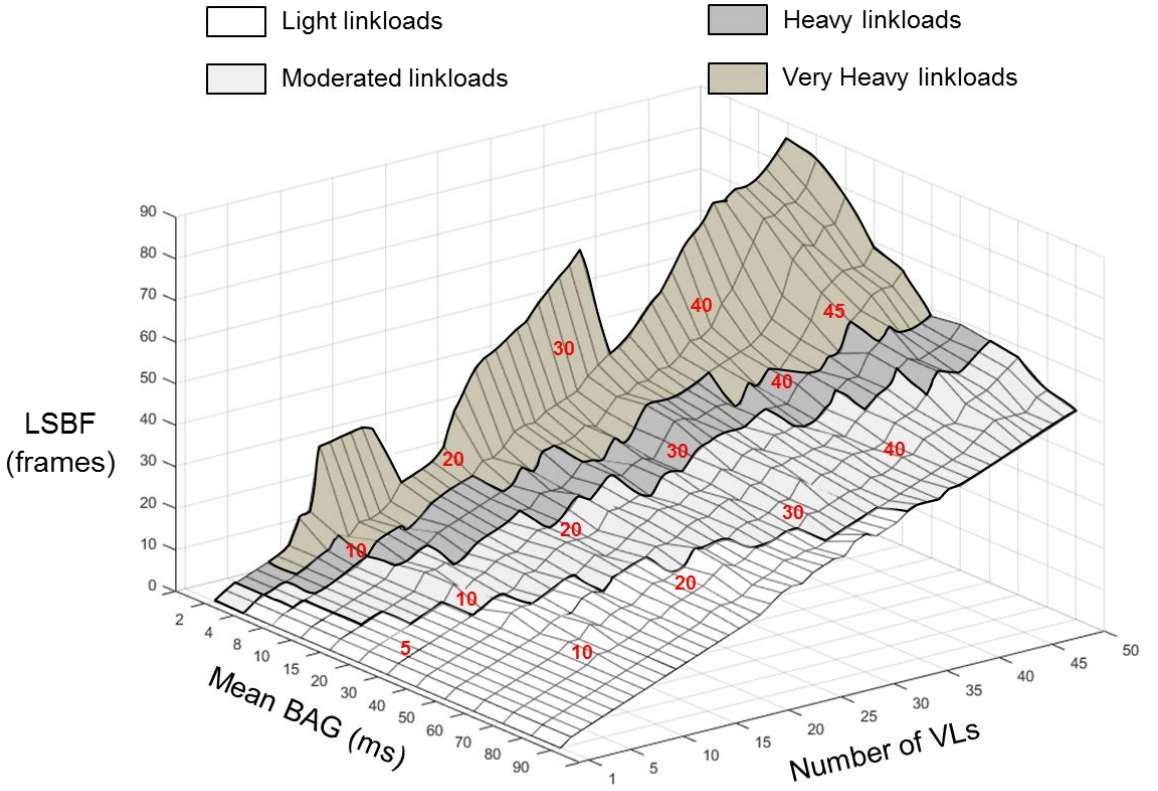


Figure 41 – LSBFs en fonction du nombre de VLs et du BAG moyen.

Pour chaque couple (nombre de VLs, BAG moyen) indiqué sur la Figure 41, nous calculons la charge du lien physique. Naturellement, les charges légères (moins de 10%) sont situées vers le BAG moyen le plus élevé et vers les plus petits nombres de VLs. En revanche, la charge augmente lorsque le BAG moyen diminue et lorsque le nombre de VLs augmente. Les résultats montrent que pour les charges légères et pour les charges modérées (entre 10% et 20%), le LSBF est à peu près égal au nombre de VLs. Cependant, une augmentation de la charge du lien vers une charge lourde (entre 20% et 40%) ou très lourde (plus de 40%) entraîne une augmentation significative et non linéaire du LSBF.

Pour une charge inférieure à 20%, nous pouvons établir qu'une modification de la CTRES conduisant à une variation du BAG moyen a un impact très faible sur le LSBF (en général, plus ou moins une

trame). Cependant, l'impact du nombre de VLs sur le LSBF est plus important car la relation entre le nombre de trames et le nombre de VLs est quasi-proportionnelle.

Pour les charges lourdes et très lourdes, une généralisation est plus hasardeuse. Le LSBF augmente pour atteindre des valeurs extrêmes et il serait difficile d'anticiper l'impact d'une modification sur une CTRES lourde ou très lourde sur le LSBF.

3.3.2 Variation du BAG moyen

Maintenant, nous essayons d'isoler l'influence du BAG sur le LSBF. Dans ce but, deux ensembles de paramètres CTRES sont choisis (le premier avec dix VLs et le second avec quarante-cinq VLs) et nous modifions la moyenne calculée sur l'ensemble des BAGs, pour une variation entre 2,0 ms et 125 ms. Le $L_{i,\max}$ moyen et le $\Delta\Gamma^{R}_{i,\max}$ moyen conservent les mêmes valeurs que dans les simulations précédentes.

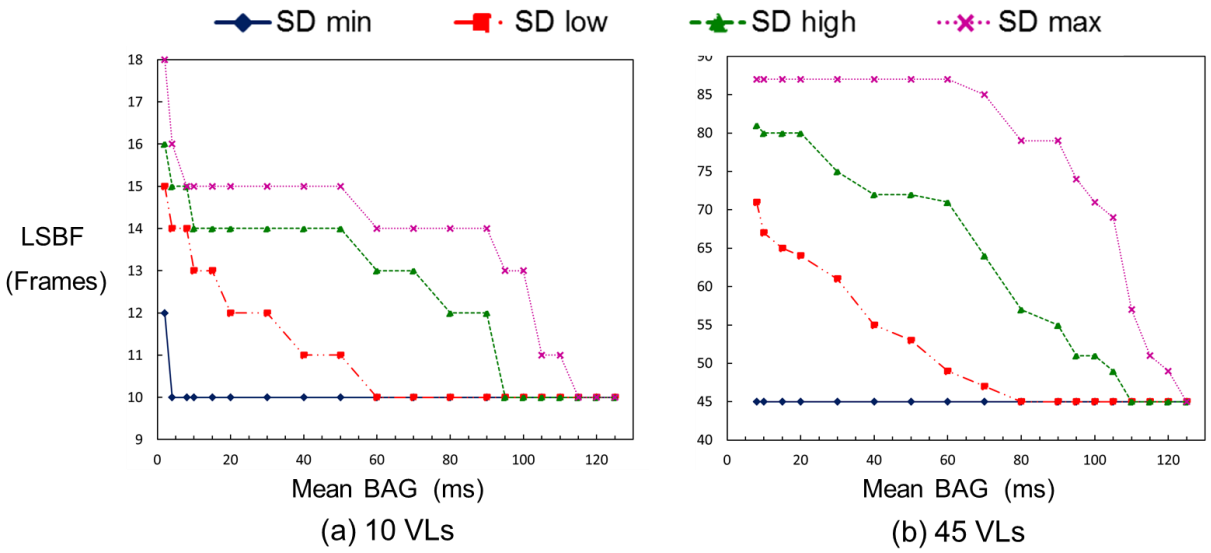


Figure 42 – Taille du LSBF en fonction du BAG moyen pour 10 VLs (a) et 45 VLs (b), pour quatre niveaux de dispersion des valeurs de BAG autour de la moyenne.

En outre, il est intéressant d'estimer la dispersion des valeurs de BAG autour de la moyenne avec un paramètre de dispersion. L'*écart-type* (en anglais : **SD (Standard Deviation)**) est un paramètre de dispersion simple à calculer et souvent utilisé. Cependant, les valeurs de BAG ne varient pas linéairement mais en puissance de 2. Donc, l'écart-type n'est pas représentatif de la dispersion des valeurs de BAG autour du BAG moyen de la CTRES mais il peut au moins indiquer une tendance au resserrement ou à la dispersion des valeurs autour de la moyenne. En effet, avec l'augmentation de la moyenne des BAGs, l'écart-type varie de façon chaotique à cause de la définition du BAG. Nous proposons donc quatre niveaux qualitatifs de dispersion des valeurs de BAG autour de la moyenne : **SD min**, **SD low**, **SD high** et **SD max**.

La Figure 42 montre qu'une augmentation du BAG moyen implique une diminution du LSBF puisque la charge diminue également. De plus, un résultat intéressant apparaît : l'écart-type du BAG influence

considérablement le LSBF. Une dispersion minimale donne un LSBF dont la valeur est proche du nombre de VLs pour la majorité des simulations. Lorsque la dispersion des valeurs de BAG autour de la moyenne augmente, la proportion des VLs dont le BAG est petit augmente. Par conséquent, l'écart $\tau^{s}_{i,j+1} - \tau^{e}_{i,j}$ entre deux intervalles consécutifs diminue pour une partie des VLs jusqu'à ce qu'il soit assez petit pour permettre l'appartenance de deux trames consécutives de ce VL au LSBF.

Pour 45 VLs, nous notons que le LSBF varie de 93% pour un BAG moyen inférieur à 60 ms entre la dispersion minimale et la dispersion maximale. En outre, nous pouvons prédire une variation plus grande du LSBF avec une augmentation du nombre de VLs. Ainsi, il convient de mesurer précisément la dispersion des valeurs de BAG autour du BAG moyen de la CTRES pour déterminer s'il sera possible d'anticiper l'impact de l'ajout d'un VL dont le BAG est éloigné du BAG moyen de la CTRES avant la modification.

3.3.3 Variation du $L_{i,\max}$ moyen

Le $L_{i,\max}$ moyen agit sur la durée de réception effective d'une trame. Ainsi, il est possible d'estimer à priori que plus les trames sont petites, plus leur durée de réception est courte, et donc plus le nombre de trames du LSBF est amené à se réduire.

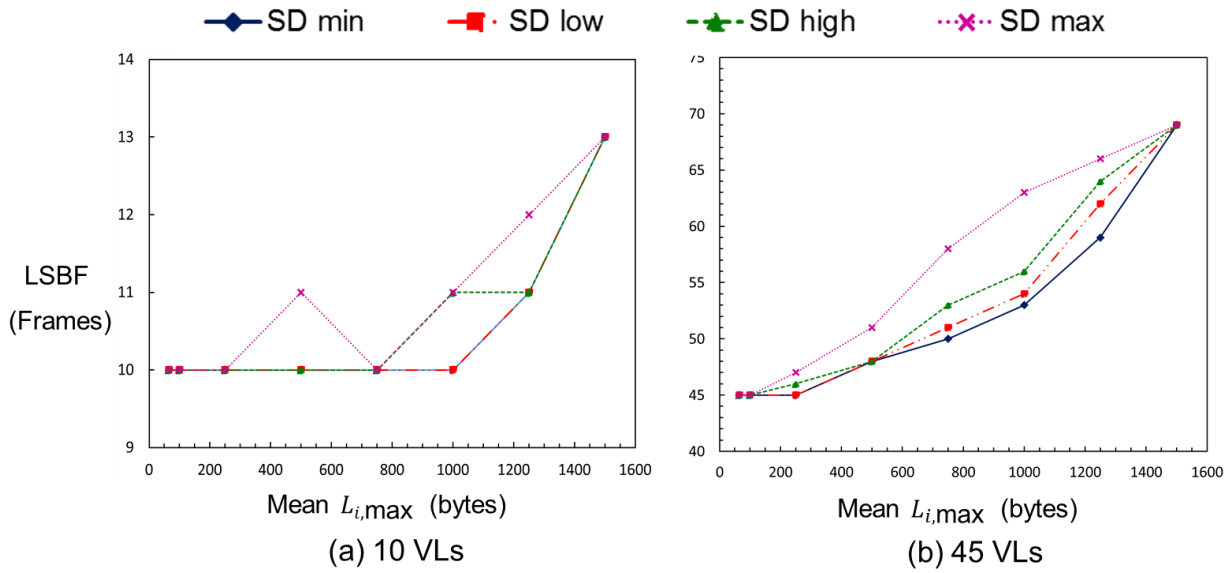


Figure 43 – Taille du LSBF en fonction du $L_{i,\max}$ moyen pour 10 VLs (a) et 45 VLs (b), pour quatre niveaux de dispersion des valeurs de $L_{i,\max}$ autour de la moyenne.

Les résultats présentés sur la Figure 43 confirment cette intuition. Le $L_{i,\max}$ moyen varie de 65 à $1\,500$ octets et la dispersion des $L_{i,\max}$ est présentée sur quatre niveaux de dispersion, comme précédemment. Le BAG moyen est fixé à 2,0 ms pour 10 VLs (a) et à 10,0 ms pour 45 VLs (b) et son niveau de dispersion est SD low. Ce choix est motivé par la volonté de concentrer nos mesures sur une zone où nous sommes susceptibles d'observer d'importantes variations du LSBF. Enfin, le $\Delta\Gamma^{R}_{i,\max}$ reste à une moyenne de 750 ms et à un niveau SD low pour la dispersion.

Quelle que soit le niveau de dispersion des $L_{i,\max}$ autour de la moyenne, les résultats montrent que si le $L_{i,\max}$ moyen est inférieur à environ 600 octets, alors le nombre de trames du LSBF est proche du nombre de VLs. Comme le $L_{i,\max}$ est généralement petit (plus de 80% des VLs ont un $L_{i,\max}$ inférieur à 300 octets sur un réseau AFDX industriel (36)), nous considérons que le $L_{i,\max}$ est un facteur qui a peu d'influence sur la taille du LSBF.

En outre, lorsque le $L_{i,\max}$ moyen dépasse 600 octets, seule une dispersion importante des valeurs de $L_{i,\max}$ entraîne un écart important entre les LSBFs calculés. Enfin, nous observons une convergence du LSBF pour un $L_{i,\max}$ moyen proche de 1 500 octets pour tout niveau de dispersion car le $L_{i,\max}$ moyen atteint la valeur maximale de la taille de trame autorisée par l'ARINC 664 et les dispersions autour de la valeur maximale sont restreintes.

3.3.4 Variation du $\Delta\Gamma^{R}_{i,\max}$ moyen

Le dernier paramètre dont la variation peut impacter le LSBF est la variation maximale du délai de bout en bout $\Delta\Gamma^{R}_{i,\max}$. Le $\Delta\Gamma^{R}_{i,\max}$ se différencie des trois paramètres CTRES précédents dans le sens où une variation du $\Delta\Gamma^{R}_{i,\max}$ moyen ou de la dispersion des $\Delta\Gamma^{R}_{i,\max}$ autour de la moyenne ne conduit pas à une variation de la charge du lien physique. Cela signifie que seule la taille des intervalles varie et non le nombre d'intervalles dans l'hyperpériode T_{hyp}, ni leur position relative, ni la durée de réception effective des trames.

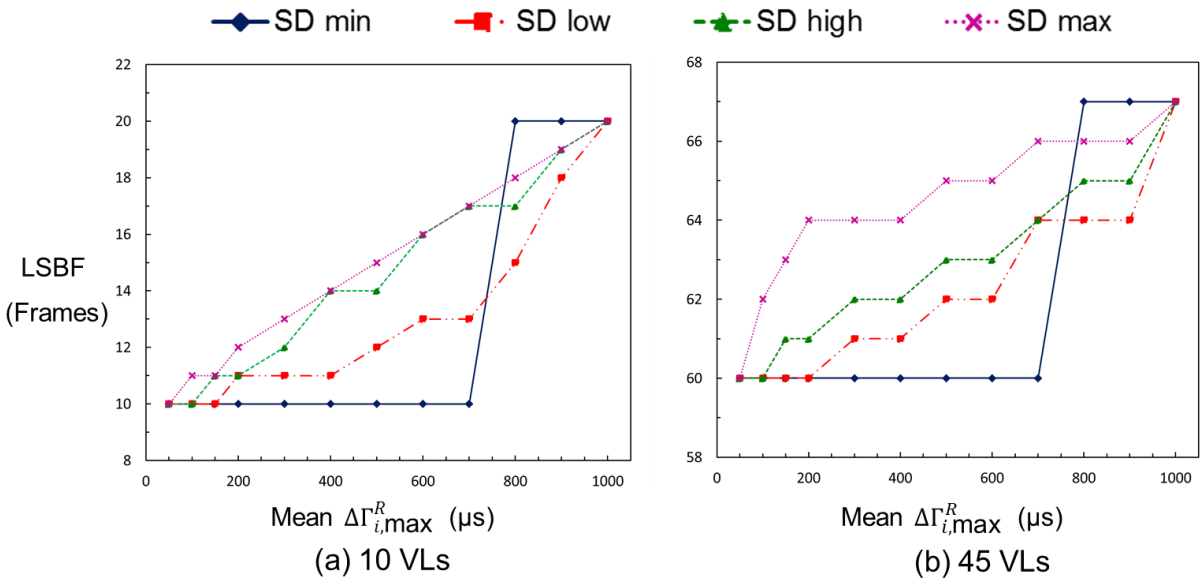


Figure 44 – Taille du LSBF en fonction du $\Delta\Gamma^{R}_{i,\max}$ moyen pour 10 VLs (a) et 45 VLs (b), pour quatre niveaux de dispersion des valeurs de $\Delta\Gamma^{R}_{i,\max}$ autour de la moyenne.

Comme précédemment, nous choisissons un ensemble de paramètres CTRES constants conduisant à une observation aisée de l'effet de la variation du $\Delta\Gamma^{R}_{i,\max}$ moyen sur la taille du LSBF, comme le montre la Figure 44. Le BAG moyen est fixé à 2,0 ms pour 10 VLs (a) et à 10,0 ms pour 45 VLs (b) et son

niveau de dispersion est SD low. Le $L_{i,\max}$ moyen est fixé à 1 500 octets et pour un niveau de dispersion SD min. Ainsi, la charge est de 60% pour 10 VLs (a) et de 54% pour 45 VLs (b), ce qui correspond à des charges très lourdes.

Sur la Figure 44, nous notons une augmentation de 100% du LSBF pour 10 VLs et une augmentation de 12% pour 45 VLs lorsque le $\Delta\Gamma^R_{i,\max}$ moyen passe de 50 à 1 000 µs. Cela s'explique par le lien direct entre $\Delta\Gamma^R_{i,\max}$ et la taille de l'intervalle (propriété 3.1) sur chaque VL_i, mais aussi par l'impact du $\Delta\Gamma^R_{i,\max}$ sur la durée $\tau^s_{i,j+1} - \tau^e_{i,j}$. Par conséquent, une augmentation du $\Delta\Gamma^R_{i,\max}$ moyen implique une réduction générale des durées $\tau^s_{i,j+1} - \tau^e_{i,j}$ pour chaque VL_i et donc des chaînes d'intervalles $C_{i,j}$ comportant davantage d'intervalles, ce qui entraine à son tour une augmentation du nombre potentiel de VLs ayant plusieurs trames dans le LSBF.

Ceci est particulièrement vrai sur la Figure 44 (a) pour la courbe bleue correspondant à une dispersion SD min des valeurs $\Delta\Gamma^R_{i,\max}$ autour de leur moyenne. Entre 700 et 800 µs, nous observons un doublement du nombre de trames composant le LSBF. Expliquons ce saut. Pour un niveau de dispersion SD min, les valeur de $\Delta\Gamma^R_{i,\max}$ sont toutes égales, i.e. les intervalles des 10 VLs sont de la même taille. En-dessous de 800 µs, le LSBF est composé d'une séquence de 10 trames de 1 500 octets chacune, chaque VL_i contribuant à hauteur d'une trame au LSBF. Nous avons donc 10 trames accolées en cascade, d'une durée $\lambda_{\mathrm{LSBF}} = 1\,200\ \mu s$. Lorsque $\Delta\Gamma^R_{i,\max}$ vaut 800 µs, la durée $\tau^s_{i,j+1} - \tau^e_{i,j}$ vaut 1 200 µs et donc vaut λ_{LSBF}. Ceci signifie qu'il est possible de placer la première séquence de 10 trames de sorte à couvrir l'espace entre deux intervalles consécutifs sur un VL_i. Ainsi, il devient possible de placer deux trames de chaque VL_i dans le LSBF, d'où le basculement de 10 trames vers 20 trames à partir de 800 µs pour le $\Delta\Gamma^R_{i,\max}$, compte tenu des conditions particulières évoquées.

3.3.5 Synthèse des résultats de simulation

Pour résumer, nous avons évalué l'influence des trois principaux paramètres CTRES et de la variation maximale du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ sur la taille du LSBF dont l'estimation est indispensable pour mesurer le backlog de la mémoire de réception.

Le nombre de VLs et les valeurs de BAG sont les paramètres qui ont le plus de poids sur le calcul du LSBF. Le plus fort impact sur la taille du LSBF est mesuré lorsqu'un VL est ajouté à la CTRES et que sa valeur de BAG est faible (2,0 ou 4,0 ms). Ceci est particulièrement vrai lorsque le nombre de VLs est déjà grand (charge plus élevée).

Le $L_{i,\max}$ pèse également sur le LSBF mais seulement si le $L_{i,\max}$ moyen est élevé. Or, le $L_{i,\max}$ moyen est généralement inférieur à 300 octets dans les CTRESs industrielles où les courtes trames sont privilégiées. Ainsi, le $L_{i,\max}$ sera considéré comme un paramètre de seconde importance pour le calcul du LSBF.

Enfin, le $\Delta\Gamma^R_{i,\max}$ pris seul a peu d'impact sur le LSBF puisqu'il doit être suffisamment grand pour modifier la contribution d'un VL au LSBF tout en ayant une faible valeur du BAG associée. Si ces deux conditions ne sont pas respectées, la taille du LSBF ne sera pas modifiée.

L'ensemble de ces résultats répondent à un besoin de prédictibilité de l'évolution du pire scénario du flux de trames entrant dans la mémoire de réception. En effet, une augmentation de la taille du LSBF peut conduire à un redimensionnement de la mémoire et donc à une recertification de l'architecture matérielle. Au contraire, une réduction de la taille du LSBF n'implique pas de redimensionnement de la mémoire. Or, les résultats obtenus permettent d'identifier les modifications partielles des paramètres CTRES ou de la variation maximale du délai de bout en bout entraînant une diminution ou à une augmentation du LSBF. Ainsi, il est possible par exemple de définir une marge sur la taille de la mémoire de réception autorisant des modifications partielles des paramètres de la CTRES sans redimensionnement mémoire.

3.4 Estimation du WFB à l'aide du modèle basé sur les intervalles de réception

Compte tenu de ce qui précède, nous avons analysé avec précision le flux entrant et nous disposons d'une méthode de construction pour calculer le LSBF pour toute CTRES.

Dans cette section, l'attention est portée au backlog de la mémoire de réception comme la différence entre le flux de trames entrant et le flux de trames sortant. Les modèles d'écriture et de lecture dans la mémoire du chapitre précédent sont repris pour l'estimation du backlog dans deux scénarios de réception. Puis, nous comparons les estimations du LSBF entre le modèle pessimiste (chapitre 2) et le modèle basé sur les intervalles de réception (chapitre 3).

3.4.1 Réception du LSBF

Dans le premier scénario, une CTRES de soixante-dix VLs est choisie et nous traitons la réception du LSBF pour déterminer la taille optimale de la mémoire de réception. Soixante-treize trames (109 500 octets) composent le LSBF obtenu à l'aide de la méthode de construction basée sur les intervalles de réception.

Sur la Figure 45, les flux d'entrée (bleu) et de sortie (rouge) sont représentés par des fonctions de en escaliers qui mesurent les flux entrants et sortants en octets. Pour rappel, les fonctions en escalier sont une alternance de segments ascendants et horizontaux. Un segment ascendant correspond à la durée de réception effective d'une trame et un segment horizontal correspond au temps d'attente ε (ε étant la durée minimale entre la réception de deux trames consécutives). Le flux de sortie a les mêmes segments ascendants que le flux d'entrée car la vitesse d'écriture et la vitesse de lecture sont égales, mais les segments horizontaux sont plus grands à cause du délai θ_{READ}.

Le backlog est mesuré sur la Figure 45 (a) pour un délai θ_{READ} de 10,0 µs sur le flux de sortie, et sur la Figure 45 (b) pour un délai θ_{READ} de 20,0 µs. La réception du LSBF commence à la date 0 ms et se termine à la date 8,875 ms pour la réception des soixante-treize trames du LSBF.

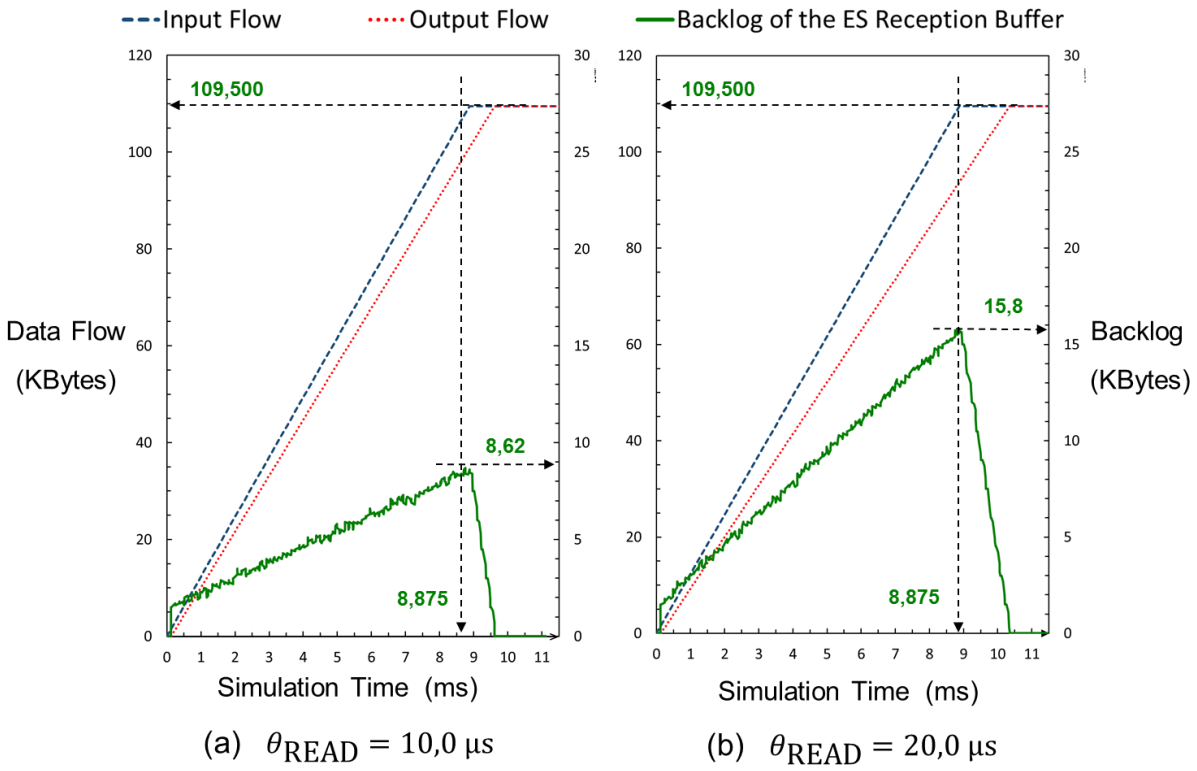


Figure 45 – Mesure du backlog de la mémoire de réception lors de la réception du LSBF pour un retard de lecture θ_{READ} de 10,0 µs (a) et de 20,0 µs (b).

Comme le suggère l'intuition, la durée du vidage total de la mémoire est plus longue lorsque le retard θ_{READ} sur le flux de sortie est plus grand. Le WFB est atteint lorsque la dernière trame du LSBF est reçue. Pour un délai de 10,0 µs, le WFB est de 8,62 Koctets et pour un délai de 20,0 µs, le WFB est de 15,8 Koctets. Nous mesurons donc un doublement du WFB lorsque le délai de lecture double.

3.4.2 Réception du LSBF suivi d'un SBF

Le second scénario est la réception du LSBF suivi de la réception d'un SBF. Ainsi, nous souhaitons vérifier que la réception d'un SBF suivant le LSBF ne conduit pas à un dépassement de la taille de la mémoire et identifier les facteurs pouvant conduire à un dépassement.

Les conditions de simulation sont les mêmes que pour la simulation précédente avec un délai à la lecture θ_{READ} de 20,0 µs. Sur la Figure 46, le LSBF est formé de soixante-treize trames (ou 109 500 octets) et le SBF suivant est formé de trente-et-une trames (49 100 octets). Il s'agit d'un scénario de réception dont l'occurrence est très improbable puisque qu'une telle réception ne pourrait se produire qu'en cas de congestions répétées du port de sortie du commutateur.

À la date 8,87 ms, le premier maximum pour le backlog est mesuré à 15,8 Koctets. Il s'en suit une phase de décroissance avant la réception du SBF suivant dont la réception démarre à la date 9,69 ms. À partir de la date 9,69 ms, le backlog croît de nouveau jusqu'à la fin de la réception du SBF à la date 13,58 ms. Un second maximum est alors atteint et vaut 12,0 Koctets. Nous constatons que le premier

maximum de 15,8 Koctets n'est pas dépassé, ce qui implique que la réception du premier LSBF conduit à la mesure du WFB égal au premier maximum.

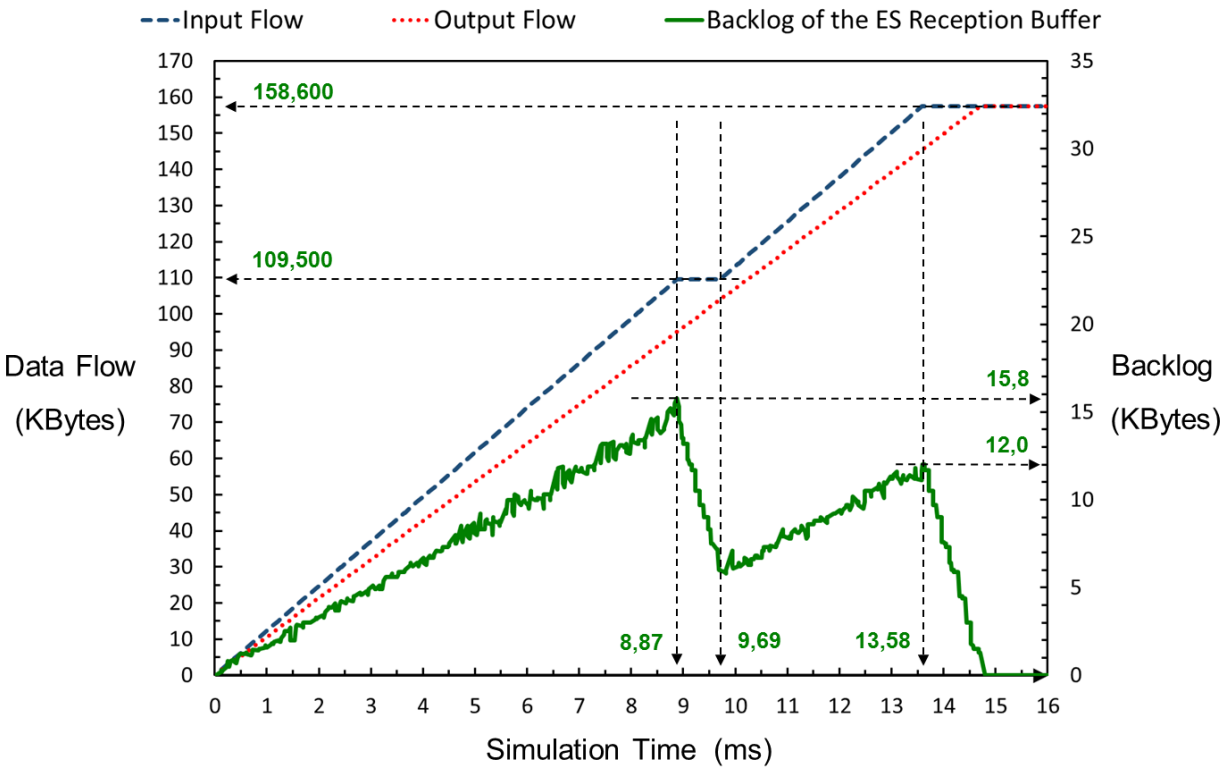


Figure 46 – Mesure du backlog de la mémoire de réception lors de la réception d'un LSBF suivi d'un SBF pour un délai de lecture θ_{READ} de 20,0 µs.

La condition de non-dépassement du WBF obtenu pour la réception du LSBF est liée à la durée d'attente entre la fin de réception du LSBF (date 8,87 ms) et du SBF (date 9,69 ms). Si cette durée est trop petite, un dépassement du premier maximum est possible si le SBF suivant présente un nombre suffisant de trames.

Dans la simulation présentée sur la Figure 46, nous choisissons une faible valeur pour cette durée (0,82 ms) en considérant qu'un tel scénario peut se produire uniquement si le port de sortie du commutateur final contient déjà un grand nombre de trames en cas de congestion et de pic de trafic sur l'ensemble du réseau AFDX. La probabilité du dépassement du maximum du backlog obtenu lors de la réception du LSBF est supposée extrêmement faible et nous restons dans le cadre de l'heuristique 2.1.

3.4.3 Comparaison entre les approches proposées

Pour comparer l'approche pessimiste avec l'approche basée sur les intervalles de réception, nous devons calculer avec chacune le nombre de trames composant le LSBF pour plusieurs CTRES.

Pour rappel, la méthode pessimiste calcule le LSBF à partir d'un flux périodique de trames de manière à former le SBF initial. Ensuite, la prise en compte de la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{\max}$ sur l'ensemble des VL_i brise la périodicité du flux de réception en retardant les trames proches du SBF initial et les trames du SBF initial. Ainsi, le LSBF est construit par l'addition des trames proches du SBF initial au SBF selon l'heuristique pessimiste : *Pour la construction du LSBF, la plus grande variation maximale des délais de bout en bout $\Delta\Gamma^{R}_{max}$ est appliquée à l'ensemble des VL_i de la CTRES.*

La principale différence entre les méthodes pessimistes et basées sur le modèle par intervalles tient en la façon dont les variations des délais de bout en bout $\Delta\Gamma^{R}_{i,\max}$ ont une incidence sur le flux entrant. En effet, le modèle par intervalles considère les $\Delta\Gamma^{R}_{i,\max}$ sur les VLs pris individuellement. Ceci est à la fois plus réaliste et plus précis parce que les VLs suivent des chemins physiques différents sur le réseau. Ils sont donc soumis à des variations de délai dont les valeurs maximales varient. De plus, la notion d'intervalle de réception permet une transition plus fine lors du passage de la modélisation d'un VL à n VLs.

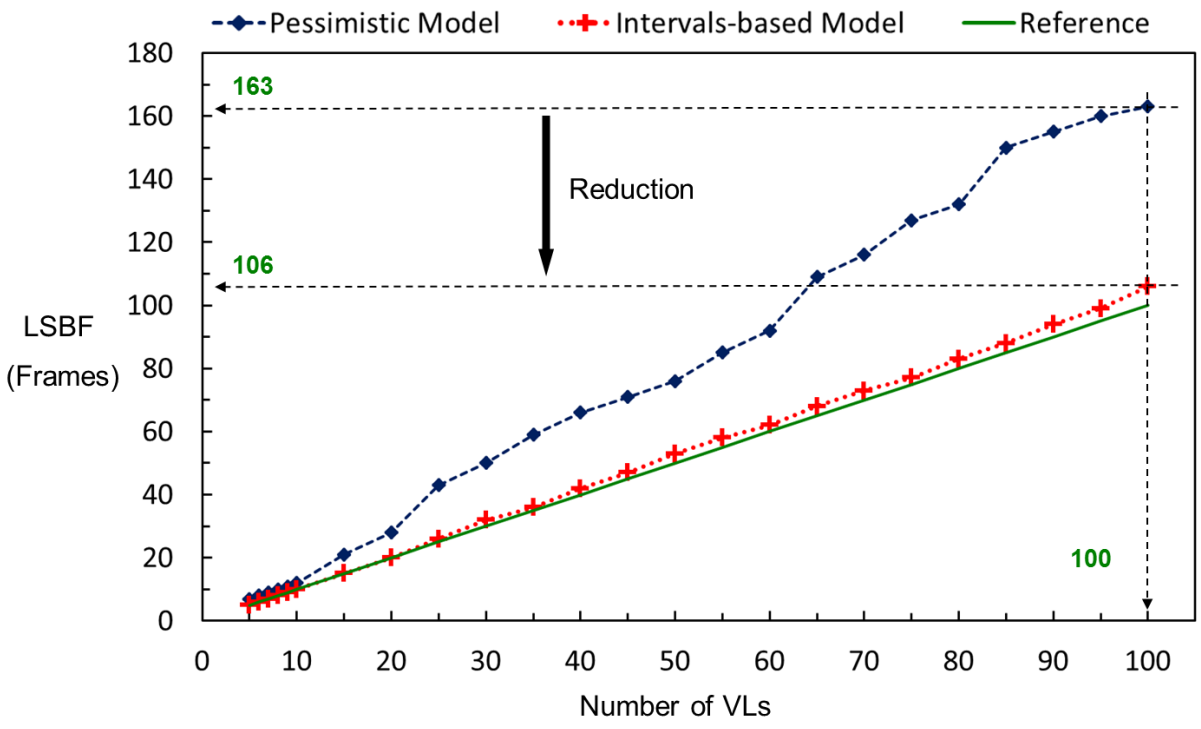


Figure 47 – Étude comparative : taille du LSBF en fonction du nombre de VLs obtenue avec le modèle pessimiste (en bleu) et obtenue avec le modèle par intervalles (en rouge).

La Figure 47 représente la taille du LSBF en fonction du nombre de VLs pour les deux approches. Nous avons choisi de faire varier le nombre de VLs car il s'agit de l'un des paramètres CTRES (avec le BAG moyen) dont la variation a le plus d'influence sur le plus la taille du LSBF. La droite de référence est simplement le plus petit LSBF dont nous pouvons prouver l'existence pour chaque CTRES. Il s'agit d'un LSBF comprenant un nombre de trames égal au nombre de VLs.

Sur la Figure 47, la première constatation est que les LSBFs obtenus avec le modèle par intervalles sont plus petits en nombre de trames que les LSBFs obtenus avec le modèle pessimiste. L'écart tend à s'accroître au fur et à mesure que le nombre de VLs augmente. La réduction du nombre de trames du LSBF atteint 65% pour 100 VLs.

En conclusion, cette étude montre que le modèle basé sur les intervalles de réception apporte une réduction du nombre de trames des LSBFs calculés par rapport au modèle pessimiste. Le modèle par intervalles est en effet plus réaliste par rapport à la première approche basée sur une heuristique pessimiste. Ainsi, l'estimation du pire scénario de réception pour le flux de trames entrant dans la mémoire de réception ES pourrait être plus réaliste en se basant sur le modèle basé sur les intervalles de réception. Le dimensionnement de la mémoire résultant de la considération de la seconde approche amènerait à des tailles plus réduites.

3.5 Conclusion

En supposant que le LSBF conduit au WFB dans la mémoire de l'ES de réception, une analyse précise du flux de trames entrant dans la mémoire est requise. Dans ce chapitre, nous avons proposé un modèle dit *par intervalles* basé sur la notion d'intervalles de réception dans lesquels les trames sont reçues au niveau de l'ES.

Ces intervalles existent puisque les délais de bout en bout sont bornés et variables. Ils varient en taille d'un VL de la CTRES à l'autre, en fonction du chemin statique de chaque VL_i. Par une méthode de construction basée sur le modèle par intervalles, nous avons constitué le flux de trames entrant de l'ES à partir des intervalles de réception de chaque VL_i de la CTRES. Puis, en plaçant les trames à l'intérieur des intervalles sur la base d'hypothèses de placement favorables à l'accolement de trames, le LSBF est obtenu. Un simulateur codé en C a été développé pour mettre en œuvre cette méthode de construction et fournir des estimations du WFB.

Comme résultats, l'influence des paramètres CTRES sur le calcul du LSBF est précisée. Nous avons déduit les tendances générales pour anticiper l'impact d'une évolution de la CTRES sur le LSBF et donc sur la taille de la mémoire. Les résultats suivants abordent la mesure du WFB à partir des LSBFs calculés sous différents délais θ_{READ} sur le flux de sortie.

Puis, le WFB est mesuré lors de la réception d'un LSBF suivant d'un SBF de façon à mettre en évidence les facteurs pouvant conduire à une sortie du cadre posé par l'heuristique 2.1. Ces facteurs sont le délai d'attente entre la réception du LSBF et du SBF et la quantité d'octets composant le SBF. Si ces facteurs restent inférieurs à des seuils calculés suivant le LSBF, l'heuristique 2.1 est respectée. Nous avons supposé qu'un dépassement de ces seuils était très improbable, compte tenu des conditions de réalisation d'un tel scénario de réception (charge élevée).

Enfin, une étude comparative avec le modèle pessimiste a mis en évidence une réduction du nombre de trames composant le LSBF allant jusqu'à 35% pour une même CTRES. Ceci indique que le

modèle par intervalles donne des LSBFs plus petits que le modèle pessimiste du fait qu'il considère les variations de délai de bout en bout sur les VLs individuels et non la plus grande variation des délais de bout en bout de l'ensemble des VLs de la CTRES.

Troisième partie

Étude probabiliste de l'occurrence de SBFs dans le flux entrant

Chapitre 4 : Probabilités d'occurrence de SBF : méthode probabiliste

La partie précédente a présenté deux approches dont le but était de déterminer la taille du LSBF pouvant être reçu au niveau d'un ES pour une CTRES donnée. En effet, nous avons supposé que le LSBF conduit au WFB dans la mémoire de réception, le WFB permettant de déduire lui-même la taille optimale de la mémoire. Ces approches ont été élaborées à partir de méthodes de construction qui s'appuient à la fois sur une analyse précise des paramètres CTRES et sur la variation maximale du délai de bout en bout $\Delta\Gamma_{i,\max}^{R}$ pour chaque VL_i. Or, la construction de LSBF ne donne pas d'indication sur leur fréquence d'apparition, c'est-à-dire leur probabilité d'occurrence.

Le principal objectif de ce chapitre est d'apporter un éclairage sur la probabilité d'occurrence du LSBF pour une CTRES, et ainsi de calculer des probabilités d'occurrence de SBF. La modélisation probabiliste du flux de trames en réception est un problème difficile et nous proposons dans ce chapitre une méthode pour la recherche de SBFs dans le flux entrant. Pour cela, nous reprenons la base théorique du modèle par intervalles et nous remplaçons la méthode de construction du LSBF par une méthode basée sur des distributions de probabilité afin de créer un flux de trames entrant dans la mémoire réaliste. La méthode probabiliste rend compte de l'incertitude sur la date de réception des trames.

Déterminer les probabilités d'occurrence de SBFs sur le flux entrant généré de façon aléatoire pour différentes CTRESs est l'objectif de ce chapitre. La Section 4.1 présente les heuristiques et les hypothèses nécessaires à la création d'un flux d'entrée réaliste. Le cadre théorique est centré sur le modèle par intervalles. Le cœur de la méthode probabiliste est présenté dans la Section 4.2. Le but est de construire un algorithme qui sera ensuite implémenté en langage C, et de justifier chaque étape de celui-ci. Nous soulignons l'emploi de distributions de probabilité à des étapes clefs de la méthode, notamment lors du placement des trames sur les intervalles de réception. La Section 4.3 présente des résultats de simulation liant les paramètres CTRES à la charge du lien, puis des mesures d'occurrence de SBF pour plusieurs CTRESs. Une étude comparative entre la méthode de construction pessimiste et la méthode probabiliste est introduite dans la dernière partie de la Section 4.3. Enfin, la Section 4.4 propose une conclusion de ce chapitre.

4.1 Cadre théorique de la méthode probabiliste

4.1.1 Intérêt de la méthode

Pour une CTRES donnée, le dimensionnement de la mémoire de réception est lié au calcul du WFB pouvant apparaître dans la mémoire, et donc au calcul de la différence maximale entre le flux de trames entrant et le flux de trames sortant. Comme étudié dans la partie précédente, le flux de sortie dépend de la vitesse de traitement de la couche UDP + IP. Le flux d'entrée dépend des paramètres CTRES et des aléas technologiques responsables de la variation du délai de bout en bout $\Delta\Gamma_{i,j}^{R}$.

Sporadiquement, des SBFs se constituent au sein du réseau par la coïncidence de plusieurs phénomènes temporels et conduire à la réception d'un SBF au niveau de l'ES. Le modèle pessimiste et le modèle basé sur les intervalles de réception exploitaient l'ensemble des paramètres CTRES pour créer artificiellement les conditions d'apparition du LSBF sur la base d'heuristiques à l'aide de méthodes de construction. Le LSBF correspondant à une estimation du pire scénario de réception pour le flux entrant, un modèle simple du flux de sortie basé sur un délai constant de lecture θ_{READ} a permis le calcul du WFB. À la suite de ces méthodes de construction du LSBF basées sur des modèles estimant le pire scénario de réception et sur la base d'heuristiques et d'hypothèses communes, nous proposons une méthode probabiliste de recherche de SBFs dans le flux entrant dans la mémoire de réception.

L'intérêt principal d'une modélisation probabiliste est de calculer les probabilités d'occurrence des SBFs (et, le cas échéant, du LSBF) pour une CTRES donnée, et d'ouvrir une discussion sur la prise en compte uniquement du LSBF pour le dimensionnement de la mémoire de réception (heuristique 2.1). L'idée est que si la probabilité d'occurrence du LSBF est très faible, peut être pourrons-nous prendre en compte un SBF plus petit que le LSBF pour dimensionner la mémoire de réception, tout en acceptant une perte de trame rarissime au niveau de l'ES de réception.

4.1.2 Rappel des heuristiques et hypothèses

D'une manière générale, la méthode probabiliste s'appuie sur les mêmes heuristiques et hypothèses pour l'estimation du pire scénario du flux de trames entrant que le modèle par intervalles présenté dans le chapitre 3. Dans cette partie, nous rappelons ces différents points théoriques.

4.1.2.1 LSBF en réception

Heuristique 2.1 : *Le scénario de réception conduisant au backlog maximal (WFB) dans la mémoire de réception ES correspond à la réception du LSBF sur le flux de trames entrant dans la mémoire*.

De l'heuristique 2.1 découlent les hypothèses 2.1 et 2.2 :

Hypothèse 2.1 : *Tous les ESs sources du réseau transmettent leurs trames BAG-périodiquement sur leurs VLs respectifs.*

Hypothèse 2.2 : *Pour un VL_i donné, la longueur de toutes les trames de ce VL est égale à la longueur maximale $L_{i,\max}$.*

4.1.2.2 Charge du lien physique

La méthode probabiliste de recherche de SBFs repose sur le fait que les paramètres de la CTRES (nombre de VLs, BAGs et $L_{i,\max}$) conduisent à une charge limitée sur le lien physique entre le commutateur final et l'ES de réception.

Hypothèse 2.3 *: La charge du lien physique est limitée à 20%.*

En effet, la charge doit être limitée pour assurer pour garantir le caractère déterministe du réseau.

4.1.2.3 Périodicité des intervalles

Commençons par rappeler la définition d'un intervalle de réception, basée la variabilité de $\Delta\Gamma_{i,j}^{R}$ et son encadrement :

Définition 3.1 : *Les dates de début et de fin de réception de toute trame $F_{i,j}$ au niveau de l'ES de réception appartiennent à un intervalle fini non nul noté $I_{i,j}$.*

$\boldsymbol{\tau_{i,j}^{s}}$ est la date la plus précoce de réception du premier bit de la trame $F_{i,j}$, et $\boldsymbol{\tau_{i,j}^{e}}$ est la date la plus tardive de réception totale de la trame $F_{i,j}$ telles que:

$$\forall\, F_{i,j} \in \mathrm{VL}_i, \qquad \exists\, I_{i,j} = \left[\tau_{i,j}^{s}\,;\, \tau_{i,j}^{e}\right] \text{ tq } F_{i,j} \in I_{i,j}$$

Les propriétés sur la taille maximale des intervalles $I_{i,j}$ et sur la périodicité de ceux-ci sont également rappelées :

Propriété 3.1 : $$\forall\, i \in [1\,;n], \quad \tau_{i,j}^{e} - \tau_{i,j}^{s} = \Delta\Gamma_{i,\max}^{R}$$

Propriété 3.2 : $$\forall\, i \in [1\,;n], \quad \tau_{i,j+1}^{s} = \tau_{i,j}^{s} + \mathrm{BAG}_i$$

4.2 Méthode probabiliste de recherche de SBFs

La méthode probabiliste se base sur le modèle par intervalles et sa méthode de construction du LSBF. Les différences majeures avec la méthode de construction basée sur le modèle par intervalles tiennent en la manière dont la position relative des intervalles est choisie et dont les trames sont placées sur les intervalles. En effet, le placement des intervalles sur chaque VL_i et les dates de réception des trames sont tirés suivant des lois de probabilité et ne sont pas choisis selon des heuristiques. Une fois le flux entrant généré, les SBFs sont simplement comptabilisés par la méthode.

Pour souligner ces différences, nous commençons par proposer l'algorithme général de la méthode probabiliste. Les parties suivantes reprennent ces étapes et les détaillent avec soin, en particulier le placement des trames sur leur intervalle respectif qui fait l'objet d'une attention soutenue.

4.2.1 Algorithme de la méthode probabiliste de recherche de SBFs et outil de simulation

À partir d'une CTRES, la méthode probabiliste consiste en la génération d'un flux de trames entrant basée sur des distributions de probabilité, puis en la recherche de SBFs. Cette méthode repose sur cinq étapes comme l'illustre la Figure 48. Les étapes 2 et 3, en pointillés bleu sont celles qui diffèrent de la méthode de construction du LSBF présentée dans le chapitre 3. Bien que cela ne soit pas indiqué sur la figure, les fonctions liées aux étapes 1, 4 et 5 du simulateur codé en C ont dû subir quelques modifications pour s'adapter au grand nombre de trames générées.

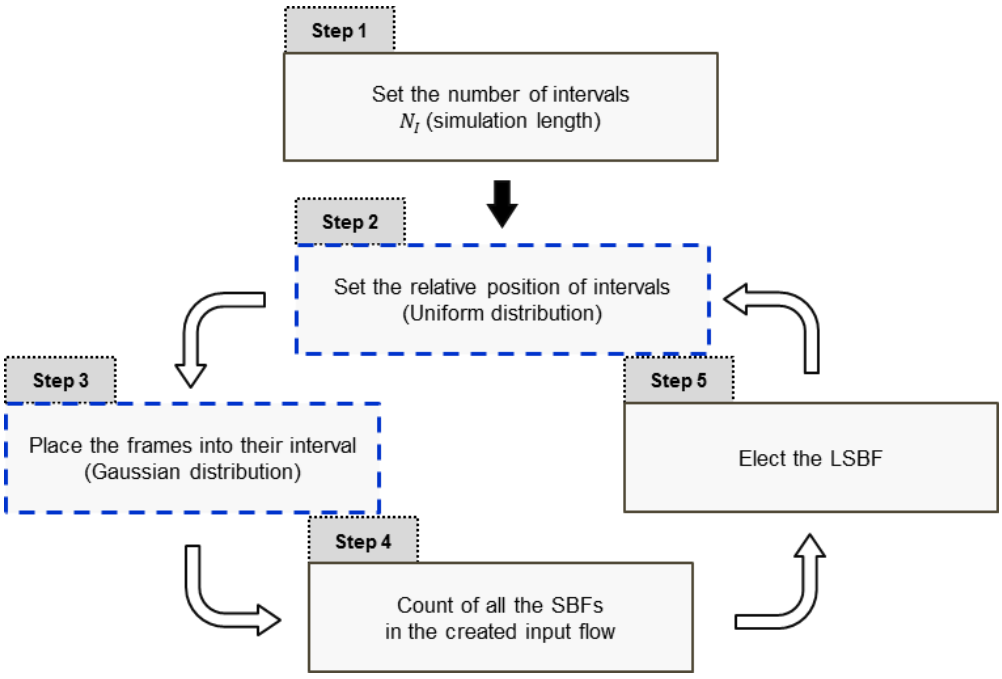


Figure 48 – Algorithme général de la méthode probabiliste de recherche de SBFs.

En entrée, le simulateur dispose, comme précédemment, des paramètres CTRES et des variations maximales du délai de bout en bout $\Delta\Gamma^R_{i,\max}$ de chaque VL_i, mais en plus le nombre d'hyperpériodes N_{hyp} sur lesquelles porte la simulation et le nombre de cycles de simulation, noté $\boldsymbol{k}$. La précision des probabilités d'occurrence des SBFs est directement lié au choix de N_{hyp} et de k mais cela impacte la durée de la simulation. L'utilisateur devra donc établir un compromis entre la précision souhaitée et la durée de la simulation.

En sortie, nous obtenons le nombre d'occurrences de SBFs classés en fonction de leur nombre de trames accolées, par rapport au nombre total de trames générées. Cela nous indique directement les probabilités d'occurrence des SBFs pour un ensemble de flux générés.

À présent, revenons sur la Figure 48 et détaillons les différentes étapes de l'algorithme. L'étape 1 consiste à déterminer le nombre d'intervalles de réception N_I sur lequel porte la recherche de SBFs, à partir du nombre d'hyperpériodes N_{hyp} choisi par l'utilisateur. Ainsi, le nombre d'intervalles N_I est un nombre fini comme pour la méthode de construction basée sur le modèle par intervalles. En revanche, le but de la méthode probabiliste est de calculer des probabilités basées sur un flux de trames entrant réaliste. Par conséquent, le nombre d'hyperpériodes N_{hyp} est choisi de manière à disposer d'un grand nombre d'intervalles (plusieurs millions) pour que les résultats de simulation soient représentatifs.

L'étape 2 a pour nature le choix de la position relative des intervalles d'un VL par rapport aux autres. Cela revient à choisir la date la plus précoce de réception du premier bit de la première trame du premier intervalle de chaque VL_i, qui se note pour rappel $\boldsymbol{\tau^S_{i,1}}$. Dans la méthode de construction du LSBF, le placement des intervalles dans le flux entrant est réalisé sur la base d'heuristiques favorisant l'apparition du LSBF. Avec la méthode probabiliste, nous réalisons un tirage suivant une distribution uniforme de la date $\tau^S_{i,1}$ pour chaque VL_i. Il est important d'effectuer le tirage de la position relative des intervalles un grand nombre de fois pour couvrir autant de cas que possible.

L'étape 3 est le placement des trames à l'intérieur des intervalles de réception. Il s'agit de tirer la date de réception totale de chaque trame suivant une loi normale en prenant en compte les variations de délai de bout en bout $\Delta\Gamma^R_{i,j}$ sur chaque VL_i. De même que pour l'étape 2, effectuer un grand nombre de tirages permet de se confronter à autant de cas de réception. Au terme de cette troisième étape, le flux entrant est totalement généré.

L'étape 4 consiste à traiter le flux de trames entrant, et ainsi à identifier les SBFs en passant en revue les dates de réception de toutes les trames du flux et en vérifiant la condition d'accolement. Les SBFs sont alors caractérisés en nombre de trames, en durée et en taille en octets.

Enfin, l'étape 5 consiste simplement à élire le plus grand SBF et à trier les SBFs détectés avant de reboucler sur l'étape 2 pour un deuxième placement des intervalles, puis les étapes 3, 4 et 5 sont répétées. Ce cycle est lui-même répété un grand nombre de fois (plusieurs millions) pour fournir des statistiques précises sur l'occurrence des SBFs.

Pour résumer, dans la méthode probabiliste de recherche de SBFs, deux étapes sont stratégiques pour compter les occurrences de SBF : le choix de la position relative des intervalles (étape 2) et le

placement des trames au sein des intervalles (étape 3). Des distributions de probabilité interviennent à ces deux étapes afin d'émuler des cas réels de fonctionnement. Le but n'est pas de déterminer le LSBF comme pour les méthodes de construction du LSBF mais de générer des flux entrants de façon aléatoire et de comptabiliser les SBFs.

4.2.2 Nombre fini d'intervalles N_I

Dans cette partie, nous détaillons l'étape 1 de l'algorithme qui consiste à calculer le nombre d'intervalles de réception N_I sur lequel porte un cycle de simulation à partir du nombre d'hyperpériodes N_{hyp} choisi par l'utilisateur et de la CTRES.

4.2.2.1 Formule

Le nombre d'intervalles de réception $\boldsymbol{N_{\text{hyp}}^{I}}$ dans une hyperpériode de durée T_{hyp} est la somme du nombre d'intervalles de réception de chaque VL_i sur une hyperpériode T_{hyp} pour n VLs de la CTRES, ce qui s'écrit :

$$N_{\text{hyp}}^{I} = \sum_{i=1}^{n} \frac{T_{\text{hyp}}}{\text{BAG}_i}$$

Le nombre d'intervalles de réception N_I se déduit immédiatement :

$$N_I = N_{\text{hyp}} \times N_{\text{hyp}}^{I} = N_{\text{hyp}} \times \sum_{i=1}^{n} \frac{T_{\text{hyp}}}{\text{BAG}_i}$$

Nous disposons donc d'une formule simple pour calculer le nombre d'intervalles (i.e. de trames) N_I nécessaire à la réalisation d'un cycle de simulation, qui dépend de la CTRES et du choix de l'utilisateur. En guise d'illustration, la Figure 49 représente le flux entrant fini sous la forme d'un nombre d'hyperpériodes N_{hyp}.

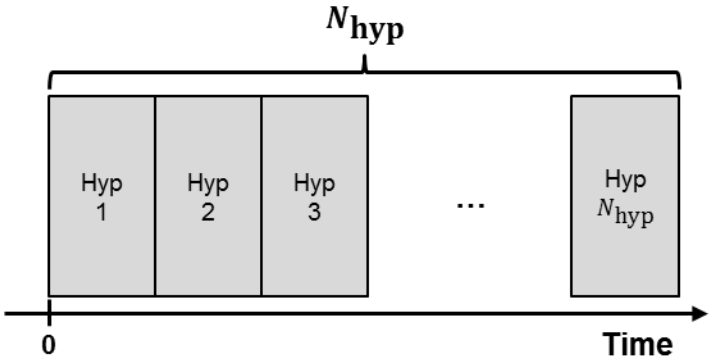


Figure 49 – Flux de trames entrant représenté comme un nombre N_{hyp} d'hyperpériodes.

4.2.2.2 Choix du nombre d'hyperpériodes N_{hyp} et du nombre de cycles de simulation k

L'utilisateur a la responsabilité du choix du nombre d'hyperpériodes N_{hyp}. Ce choix doit être guidé par la volonté d'obtenir un flux d'entrée contenant suffisamment d'intervalles pour être représentatif. Or, chaque hyperpériode comporte un nombre d'intervalles N_{hyp}^{I} qui dépend des paramètres CTRES, et en particulier des valeurs de BAG.

À cela s'ajoute le nombre de cycles de simulation k choisi par l'utilisateur qui n'a pas de conséquence directe sur l'étape 1 de l'algorithme de la méthode probabiliste. En effet, un nouveau cycle de simulation implique de refaire les étapes 2, 3, 4 et 5. Essentiellement, il s'agit de tirer de nouvelles positions relatives pour les intervalles, ce qui modifie les conditions de génération du flux de trames entrant.

Ainsi, le nombre total de trames générées à la fin de la simulation de k cycles est $k \times N_{\text{hyp}} \times N_{\text{hyp}}^{I}$. Par conséquent, la question est de choisir un nombre $k \times N_{\text{hyp}} \times N_{\text{hyp}}^{I}$ d'intervalles ni trop grand pour éviter des temps de simulation élevés, ni trop petit pour que les tirages aléatoires effectués dans les étapes suivantes portent sur un échantillon significatif d'intervalles et donc de trames.

Le nombre d'intervalles $k \times N_I$ pour plusieurs cycles de simulation est fixé à plusieurs millions quelle que soit la CTRES choisie, de façon à générer plusieurs millions de trames pour chaque simulation. L'utilisateur doit donc adapter les nombres N_{hyp} et k en fonction de la CTRES simulée. Ce choix de plusieurs millions de trames permet de calculer des probabilités avec une précision de 10^{-6} tout en limitant le temps de calcul à quelques minutes pour la simulation de k cycles.

4.2.3 Position relative des intervalles basée sur la distribution uniforme

Pour une CTRES donnée, l'étape 2 de la méthode probabiliste consiste à placer les intervalles sur chaque VL_i, c'est-à-dire, à choisir la position relative des intervalles d'un VL à l'autre.

Nous avons vu dans le chapitre 3 qu'il n'est pas possible de considérer toutes les positions relatives puisque le nombre de placements possible est très grand, d'autant plus lorsque le nombre de VLs est grand et lorsque le pas entre deux positions est petit.

Dans la méthode probabiliste, nous souhaitons générer des flux de trames entrant dans la mémoire de réception réalistes et courants. Or, le placement de tous les intervalles d'un VL_i quelconque dépend du placement de son premier intervalle $I_{i,1}$ par rapport à la date 0 (date de référence) comme le montre la Figure 50 où la date $\tau_{i,1}^{s}$ de l'intervalle $I_{i,1}$ est choisie telle que :

$$\tau_{i,1}^{s} \in [\, 0 \,;\, \text{BAG}_i \,]$$

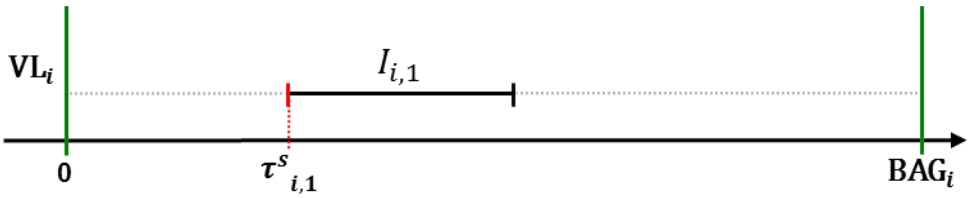


Figure 50 – Exemple de tirage de la borne inférieure $\tau^s_{i,1}$ du premier intervalle $I_{i,j}$ d'un VL_i quelconque.

La date $\tau^s_{i,1}$ mérite donc une attention particulière dans la manière elle est choisie pour chaque VL_i.

Analysons la transmission des trames sur le réseau AFDX. Au démarrage des ESs sources, l'absence de synchronisation globale et les délais de démarrage différents conduisent à des dates d'émission aléatoires pour les premières trames de l'ensemble des VLs. Ainsi, nous ne pouvons avoir aucun à priori sur les dates $\tau^s_{i,1}$ qui peuvent donc prendre n'importe quelle valeur dans l'intervalle $[\,0\,;\,\mathrm{BAG}_i\,]$. Par conséquent, la distribution uniforme continue $U(0\,;\mathrm{BAG}_i)$ est choisie pour fixer les dates $\tau^s_{i,1}$ de chaque VL_i de façon équiprobable telle que :

$$\tau^s_{i,1} \sim U(0\,;\mathrm{BAG}_i)$$

Ainsi, une fois la date $\tau^s_{i,1}$ fixée, tous les intervalles du VL_i peuvent être placés facilement d'après la propriété 3.2 sur une durée aussi longue que nécessaire. Dans notre méthode, nous commençons par placer les intervalles de chaque VL_i au sein de la première hyperpériode notée $T_{\mathrm{hyp},1}$ sur la Figure 51, puis cette hyperpériode est répliquée N_{hyp} fois.

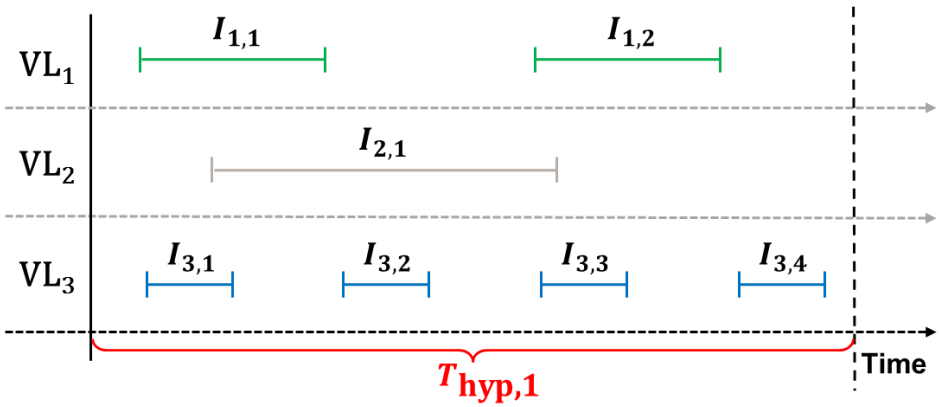


Figure 51 – Exemple de positions relatives des intervalles pour la première hyperpériode $T_{\mathrm{hyp},1}$ pour une CTRES de trois VLs.

Ainsi, comme indiqué sur la Figure 48 de l'algorithme de la méthode probabiliste, l'étape 2 de tirage des positions relatives est répétée plusieurs fois. Dans nos simulations, nous réalisons entre $1\,000$ et $10\,000$ tirages de jeux de positions relatives pour une CTRES.

4.2.4 Placement des trames basé sur la distribution de Laplace-Gauss (loi normale)

4.2.4.1 Formalisation du problème

La troisième étape de la méthode algorithmique consiste à placer les trames à l'intérieur des intervalles de réception de manière à finaliser la génération du flux de trames entrant de la mémoire. Pour réaliser cela, nous commençons par rappeler quelques notations de la caractérisation temporelle des intervalles et des trames. La Figure 52 présente les principales dates utiles pour placer une trame $F_{i,j}$ sur son intervalle de réception $I_{i,j}$.

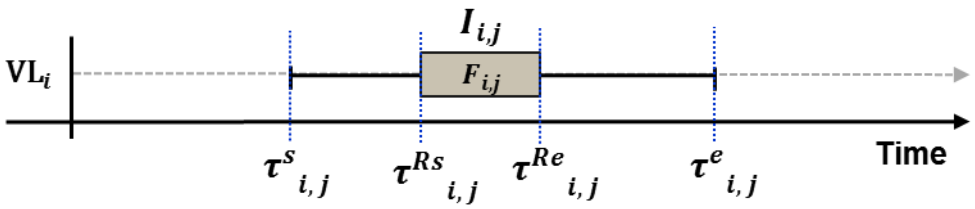


Figure 52 – Caractérisation temporelle de la réception d'une trame $F_{i,j}$ sur son intervalle $I_{i,j}$ pour un VL_i.

Ces dates sont :

- $\boldsymbol{\tau^{s}_{i,j}}$: la date *la plus précoce de réception du début* de la trame $F_{i,j}$.
- $\boldsymbol{\tau^{Rs}_{i,j}}$: la date *de début de réception* de la trame $F_{i,j}$.
- $\boldsymbol{\tau^{Re}_{i,j}}$: la date *de fin de réception* (réception totale) de la trame $F_{i,j}$.
- $\boldsymbol{\tau^{e}_{i,j}}$: la date *la plus tardive de réception totale* de la trame $F_{i,j}$.

Le placement des trames sur leur intervalle de réception est la pierre angulaire de la méthode probabiliste. Il s'agit de concevoir un modèle de placement de trames basé sur une distribution de probabilité qui prenne en compte les contraintes de la réception de trames sur un réseau AFDX. La variable aléatoire considérée est la date $\boldsymbol{\tau^{Re}_{i,j}}$ de réception totale de la trame $F_{i,j}$ qu'il faut donc choisir de façon aléatoire pour tous les intervalles du flux d'entrée. Nous pourrions tout aussi bien considérer la date $\tau^{Rs}_{i,j}$ de début de réception, ce qui reviendrait au même.

Étudions de plus près la réception d'une trame $F_{i,j}$ sur un VL_i quelconque afin de déterminer par la suite la loi de probabilité la plus adaptée au tirage des dates $\tau^{Re}_{i,j}$.

Le premier point est qu'une et une seule trame $F_{i,j}$ est reçue sur l'intervalle $I_{i,j}$. Par conséquent, une seule valeur $\tau^{Re}_{i,j}$ est associée à chaque intervalle $I_{i,j}$. Cela peut sembler trivial mais dans le

domaine des probabilités mathématiques, il est important d'identifier le mécanisme de tirage (Combien de tirages ? Avec ou sans remise ? etc…).

Le deuxième point est que le tirage de $\tau_{i,j}^{Re}$ s'effectue sur un intervalle de temps continu et borné (définition 3.1). À priori, notre choix s'orientera donc plutôt vers une distribution continue également. Une distribution est continue lorsque la ou les variables aléatoires peuvent prendre n'importe quelle valeur sur un intervalle fini ou infini. Les distributions continues se représentent à l'aide d'une fonction de densité de probabilité, et parmi ces distributions figurent les distributions de Laplace-Gauss (dites loi normale), exponentielle et χ^2. Aux distributions continues s'opposent les distributions discrètes dont les variables aléatoires ne peuvent prendre que des valeurs entières positives ou nulles. Elles se représentent à l'aide d'un diagramme en barres et il s'agit par exemple des distributions binomiales ou de Poisson.

Le troisième point est que pour respecter la propriété du déterminisme, la charge globale du réseau AFDX est limitée (hypothèse 2.3). Par conséquent, une trame a plus de chances d'être reçue après un délai de bout en bout $\Gamma_{i,j}^{R}$ proche du délai minimum $\Gamma_{i,\min}^{R}$ calculé avec un réseau vide et donc sans rencontrer de congestions dans les ports de sortie des commutateurs (56) (54). Ceci conduit à la formulation de l'heuristique 4.1 :

Heuristique 4.1 : *Pour tout VL_i, une trame $F_{i,j}$ est plus probablement reçue au début de son intervalle de réception $I_{i,j}$.*

De la même manière, il est moins probable de recevoir le début d'une trame $F_{i,j}$ à une date $\tau_{i,j}^{Rs}$ relativement éloignée de la date $\tau_{i,j}^{s}$ puisque cela implique qu'une congestion s'est probablement produite et/ou que la trame $F_{i,j}$ a subi un jitter de transmission important. En raison de la limitation de la charge du réseau, nous supposons qu'une congestion d'un port de sortie d'un commutateur est un phénomène relativement rare.

Les différentes mécaniques de la réception des trames ont été exposées dans les points précédents, cela dans le but d'orienter le choix d'une distribution de probabilité pour $\tau_{i,j}^{Re}$ adaptée.

4.2.4.2 Distribution de Poisson

La loi de Poisson (57) s'applique à des situations où l'on étudie la réalisation d'un événement dans un intervalle de temps fini. Les phénomènes ainsi étudiés sont des phénomènes d'attente comme par exemple le nombre de pannes d'un système informatique sur une année, un nombre de personnes reçues à un guichet, le comptage de désintégration radioactive d'un échantillon de Césium 137 ou encore pour faire du contrôle de qualité sur des composants électroniques. La loi de Poisson peut être interprétée comme le cas limite d'une loi binomiale.

L'application de la loi de Poisson à une situation est conditionnée par trois hypothèses portant sur la réalisation de l'événement « intéressant ». Tout d'abord, les nombres de réalisations de l'événement au cours d'intervalles de temps disjoints sont des variables aléatoires indépendantes. En d'autres

termes, le nombre de réalisations au cours d'un intervalle de temps est indépendant du nombre de réalisations au cours d'intervalles de temps antérieurs. Ensuite, la probabilité pour que l'événement se réalise une fois au cours d'un petit intervalle de temps Δt est proportionnelle à la taille de l'intervalle. Cette probabilité vaut $\alpha\Delta t$, où α est une valeur positive que l'on suppose constante. Enfin, il est très rare d'observer plus d'une fois la réalisation de l'événement au cours d'un petit intervalle de temps Δt, c'est-à-dire que la probabilité pour que l'événement se réalise plus d'une fois au cours de l'intervalle de temps Δt est négligeable.

Ces trois hypothèses caractérisent un processus de Poisson. α est une constante du processus qui représente le nombre moyen de réalisations par unité de temps et que l'on appelle *intensité du processus*. Sous ces hypothèses, la variable aléatoire X = « nombre de fois où l'événement considéré se réalise au cours d'un intervalle de temps de durée t » est distribuée suivant une loi de Poisson de paramètre $\lambda = \alpha t$. La fonction de densité de la variable X s'écrit :

$$f(k) = p(X = k) = \frac{e^{-\lambda}\lambda^k}{k!} \quad \text{où } k \in \mathbb{N}$$

L'étude de la loi de Poisson peut surprendre puisqu'il s'agit d'une distribution discrète dénombrable alors que nous venons de préciser que nous avions besoin d'une distribution continue pour le tirage de la date $\tau_{i,j}^{Re}$. De plus, dans le cas de la réception de trames, les hypothèses 2.1 et 2.3 sont effectivement respectées mais pas l'hypothèse 3.4 puisque qu'une trame est forcément reçue dans chaque intervalle. La distribution de Poisson ne peut donc pas s'appliquer à notre modèle de réception de trames par intervalles.

En revanche, si l'hypothèse 2.1 ne contraignait pas l'étude, la loi de Poisson pourrait tout à fait s'appliquer pour modéliser les dates de réception des trames, à condition de refondre le modèle de réception par intervalles. Dans ce cas, nous pourrions effectivement étudier le temps d'attente entre la réception de deux trames consécutives sur un VL particulier, puis généraliser sur l'ensemble des VLs de la CTRES. La loi de Poisson est d'ailleurs communément utilisée dans l'analyse de performance de réseaux **LAN (Local Area Network)** pour mesurer les latences dans des réseaux de type *queuing* (58), (59), ou encore pour évaluer la distribution de la température sur un réseau complexe (60).

4.2.4.3 Distribution de Laplace-Gauss (loi normale)

Nous devons cette loi aux travaux de S. Laplace (61) et de C. F. Gauss respectivement en 1809 et en 1812 d'où son nom de loi de Laplace-Gauss (aussi communément nommée *loi normale*). La loi normale revêt une importance capitale dans bon nombre de méthodes statistiques car elle apparaît comme loi limite dans des conditions très générales. La loi de Laplace-Gauss s'applique à des phénomènes complexes dont les résultats ont des causes nombreuses, dont les effets sont faibles et plus ou moins indépendants. Un exemple d'utilisation est celui de l'estimation de l'erreur commise sur la mesure d'une grandeur physique. Cette erreur résulte d'un grand nombre de facteurs tels que : vibration de l'appareil de mesure, conditions de température, de pression, d'humidité, présence de champs électromagnétiques, etc… Chacun de ces facteurs a un effet faible mais leur combinaison peut

conduire à une erreur non négligeable. Ainsi, deux mesures faites dans des conditions que l'expérimentateur juge comme identiques pourront donner des résultats différents.

La distribution normale s'applique donc aux situations dont les causes sont nombreuses et indépendantes, dont les effets s'additionnent et dont aucune n'est prépondérante. Cela correspond plutôt bien à la réception des trames dans le contexte d'un réseau AFDX puisque les délais et les latences que subissent les trames sont causés par de nombreuses sources de perturbation parmi lesquelles figurent les jitters de transmission et les retards de commutation. Chacune de ces sources contribue au délai de bout en bout d'une trame sur un VL donné. Si nous mesurons les délais de bout en bout de deux trames consécutives, nous constatons tout de suite que ces délais sont différents bien que les mesures soient effectuées dans les mêmes conditions (29).

La distribution normale semble donc adaptée à la réception des dates et nous allons appliquer à la variable aléatoire $\tau_{i,j}^{Re}$ la distribution normale, ce qui se note :

$$\tau_{i,j}^{Re} \sim N(\mu, \sigma)$$

La fonction de densité de la loi normale pour une variable aléatoire quelconque X s'écrit sous la forme :

$$f(X) = \frac{1}{\sigma\sqrt{2\pi}} \times \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right)$$

Les deux réels μ et σ sont les paramètres de la loi normale. Le paramètre μ est la moyenne de la distribution. Elle représente la position centrale de la distribution. Le paramètre σ est l'écart-type de la distribution et il influence la dispersion des valeurs autour de la moyenne. σ^2 est donc la variance.

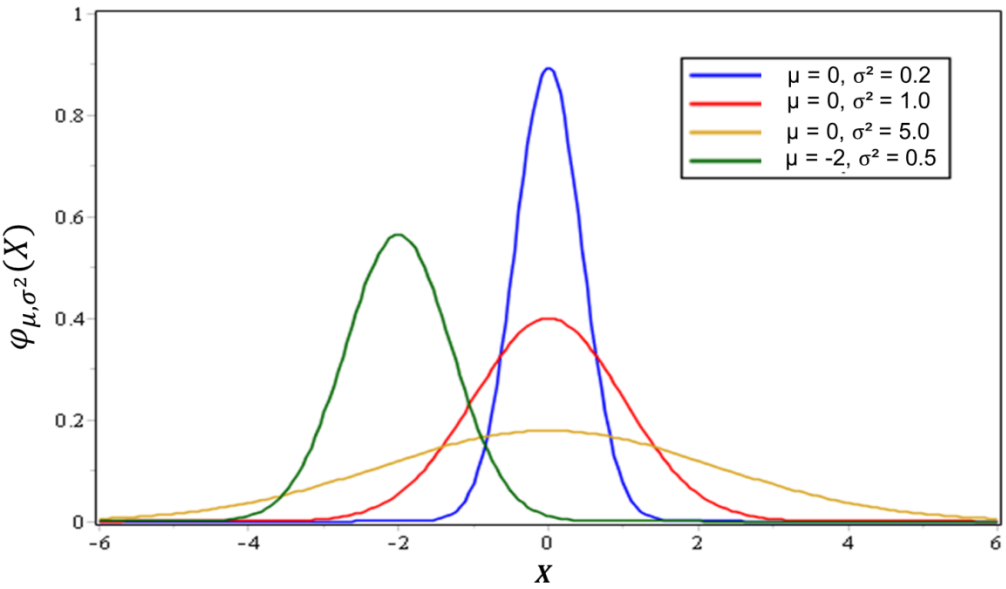


Figure 53 – Exemple de tracés de densités de probabilité pour la loi normale.

À titre d'illustration, quelques densités de probabilité ont été tracées avec Matlab et sont présentées sur la Figure 53. Quatre couples (σ ; μ) sont tracés en autant de couleurs. Il est très net que le paramètre μ agit sur la position de la courbe et le paramètre σ sur l'étalement des valeurs autour de la valeur moyenne.

À présent que le choix d'une distribution adéquate est arrêté, nous devons l'appliquer à la situation de réception des trames sur des intervalles de réception bornés et étudier son implémentation en tant qu'algorithme pour générer un grand nombre de dates $\tau_{i,j}^{Re}$. Cela passe également par la définition des paramètres σ et μ pour adapter la distribution aux contraintes AFDX.

4.2.4.4 Application de la distribution normale au contexte de la réception des trames AFDX

L'application de la loi normale au contexte de la réception des trames demande d'abord de se concentrer sur les valeurs prises par les paramètres μ et σ. Pour interpréter le fait qu'il est plus probable de recevoir une trame $F_{i,j}$ au début de l'intervalle de réception, c'est-à-dire que la date $\tau_{i,j}^{Rs}$ de réception est proche de la date $\tau_{i,j}^{s}$ (heuristique 4.1), nous fixons μ à la valeur nulle pour tous les VLs de la CTRES :

$$\mu = 0$$

Pour un VL_i donné, le paramètre σ agit sur la dispersion des délais de bout en bout $\Gamma_{i,j}^{R}$ que les trames peuvent subir lors de leur transit sur le réseau. Une faible valeur σ implique une faible dispersion autour de la moyenne et donc des délais de bout en bout $\Gamma_{i,j}^{R}$ qui varient peu et donc qui restent proches du délai minimum $\Gamma_{i,\min}^{R}$. Au contraire, une valeur de σ grande implique une forte dispersion autour de la moyenne et donc des délais de bout en bout $\Gamma_{i,j}^{R}$ qui varient *modérément* par rapport au délai minimum $\Gamma_{i,\min}^{R}$.

La question qui se pose alors est : « Comment présupposer la dispersion des délais de bout en bout $\Gamma_{i,j}^{R}$ sans avoir à disposition de nombreuses mesures de ces délais ? ». Dans le chapitre 3, nous avons supposé connue la plus grande variation maximale des délais de bout en bout $\Delta\Gamma_{i,\max}^{R}$ de chaque VL_i, ce qui correspond dans le modèle par intervalles à la taille des intervalles de réception (définition 3.1). La taille des intervalles indique les extrema des délais de bout en bout mais pas la répartition des dates de réception des trames à l'intérieur des intervalles. Il faut donc explorer une autre piste.

La dispersion des dates de réception $\tau_{i,j}^{Re}$ dans les intervalles d'un VL_i dépend de son chemin physique statique. La principale différence d'un VL à l'autre est le nombre de commutateurs traversés et la charge rencontrée sur les différents liens physiques. Ainsi, nous formulons l'heuristique suivante :

Heuristique 4.2 : *Pour tout VL_i, plus le nombre de commutateurs traversés est grand, plus il est probable pour une trame $F_{i,j}$ de rencontrer une forte charge conduisant à une congestion dans un port de sortie.*

Une congestion a pour effet principal d'augmenter le délai de bout en bout $\Gamma_{i,j}^{R}$ et donc de retarder la réception de la trame au niveau de l'ES de réception.

Par conséquent, nous attribuons un écart-type à chaque VL_i, noté $\boldsymbol{\sigma_i}$, puisque chaque VL_i a un chemin physique différent. Dans notre méthode probabiliste, nous considérons qu'un VL_i peut traverser un maximum de trois commutateurs. Ceci est réaliste compte tenu de l'architecture réseau de l'Airbus A380 (33), et ce maximum pourrait facilement être augmenté à un nombre plus grand de commutateurs.

Ainsi, trois intervalles de valeurs sont fixés pour σ_i suivant le nombre de commutateurs traversés et nous proposons une distinction entre deux catégories de VLs : les VLs dits *courts* (C-VL) et les VLs dits *longs* (L-VL) :

- $\sigma_i \in \ ]0;1]$ lorsque le VL_i traverse un commutateur (C-VL).
- $\sigma_i \in \ ]1;2{,}5]$ lorsque le VL_itraverse deux commutateurs (L-VL).
- $\sigma_i \in \ ]2{,}5;5]$ lorsque le VL_i traverse trois commutateurs (L-VL).

À ce stade, nous connaissons la manière de régler les paramètres μ_i et σ_i pour chaque VL_i. Étudions à présent comment obtenir une date $\tau_{i,j}^{Re}$ pour chaque intervalle $I_{i,j}$, dont le tirage est basé sur les distributions normales paramétrées comme évoqué précédemment.

4.2.4.5 Implémentation en C d'une distribution normale

Sous l'hypothèse qu'une trame est émise tous les BAG_i sur chaque VL_i (hypothèse 2.1), une valeur pour la variable aléatoire $\tau_{i,j}^{Re}$ doit être tirée suivant la loi $N(0,\sigma_i)$ sur tous les intervalles de réception $I_{i,j}$ de VL_i. Ainsi, comme la génération d'un flux d'entrée réaliste comporte plusieurs millions de tirages, nous avons besoin d'une méthode relativement rapide pour effectuer les tirages. La fonction densité de probabilité ne peut pas être directement implémentée en C en raison de la durée nécessaire pour réaliser un seul tirage (plusieurs secondes). Pour un très grand nombre de tirages, le temps de calcul augmente linéairement ce qui rend cette fonction inexploitable (plusieurs heures de simulation pour une seule CTRES).

Pour résoudre ce problème, une méthode d'approximation de la loi normale bien connue est mise en œuvre : l'algorithme de Box-Muller (62). L'algorithme de Box-Muller se base sur des tirages uniformes pour renvoyer une valeur X qui suit une distribution Laplace-Gauss de paramètres 0 et 1 (loi normale centrée réduite) :

$$X \sim N(0;1)$$

L'algorithme de Box-Muller s'exprime comme suit :

$$X \ = \ \sqrt{-2\ln(U_1)} \times \ \cos(2\pi.U_2)$$

Où les variables aléatoires U_1et U_2 suivent une distribution uniforme $U(0,1)$.

À partir de la variable aléatoire X, nous avons une seconde relation qui est l'expression de la variable aléatoire Y comme combinaison linéaire de X et des paramètres de la distribution de Laplace-Gauss pour le VL_i considéré :

$$Y = \mu_i + \sigma_i X$$

Dans ce cas, l'algorithme de Box-Muller précise que la variable aléatoire Y suit une distribution normale $N(\mu_i, \sigma_i)$. Le temps de génération de Y est très rapide (quelques µs) puisque basé sur des distributions uniformes qui sont bien maitrisées en langage C. L'algorithme de Box-Muller offre ainsi une méthode rapide pour générer un grand nombre de valeurs aléatoires suivant différents paramétrages de loi normale.

4.2.4.6 Génération algorithmique des dates $\tau_{i,j}^{Re}$

Les valeurs de la variable aléatoire Y ne sont pas directement exploitables comme dates de réception des trames $F_{i,j}$ sur les intervalles $I_{i,j}$ principalement pour deux raisons. La première est que la loi normale $N(0, \sigma_i)$ donne des résultats réels et donc potentiellement négatifs. La deuxième est que la loi de Laplace-Gauss donne des valeurs sur un intervalle infini, tandis que les dates des intervalles de réception sont bornés. Ainsi, l'utilisation des valeurs de Y nécessite quelques adaptations détaillées dans ce qui suit.

Nous devons établir une correspondance entre les valeurs de Y générées par l'algorithme de Box-Muller et les valeurs des dates $\tau_{i,j}^{Re}$ sachant que les conditions de génération et de correspondance varient d'un VL à l'autre. Pour éliminer les valeurs négatives, il suffit d'appliquer une fonction valeur absolue sur les valeurs de Y, ce qui ne modifie pas les probabilités d'occurrence puisque la fonction densité de probabilité est symétrique par rapport à l'axe des ordonnées. Ensuite, pour ramener les valeurs de Y sur un intervalle fini, nous appliquons une fonction de seuillage ou fonction porte. La valeur du seuil est choisie à posteriori de manière à être supérieure à la plupart des valeurs tirées. Les valeurs supérieures à ce seuil sont remplacées par des valeurs inférieures. À ce stade, nous disposons d'une liste de valeurs de Y en valeur absolue et ramenées sur un intervalle fini. De plus, le nombre de valeurs de Y générées est égal au nombre d'intervalles N_I des N_{hyp} hyperpériodes considérées pour la constitution d'un flux entrant. Bien sûr, les paramètres de la loi normale varient d'un VL à l'autre pour assurer une certaine homogénéité dans les résultats.

Enfin, un simple rapport de proportionnalité de la taille de l'intervalle $I_{i,j}$ par rapport à l'amplitude des valeurs de Y permet de « ramener » les valeurs de Y sur l'intervalle $I_{i,j}$ et donc de générer des dates $\tau_{i,j}^{Re}$. La Figure 54 propose une représentation faisant le lien entre la génération des Y et les dates $\tau_{i,j}^{Re}$ sur deux VLs dont les paramètres σ_i ont des valeurs différentes. La partie positive de la densité de probabilité est représentée en bleu pour suggérer la répartition des probabilités de tirage des dates $\tau_{i,j}^{Re}$ sur l'intervalle $I_{i,j}$. Le décalage entre la date $\tau_{i,j}^{s}$ de début de l'intervalle et la position du maximum

de la courbe de densité de probabilité est égal à la durée de réception effective d'une trame $F_{i,j}$ sur le VL_i.

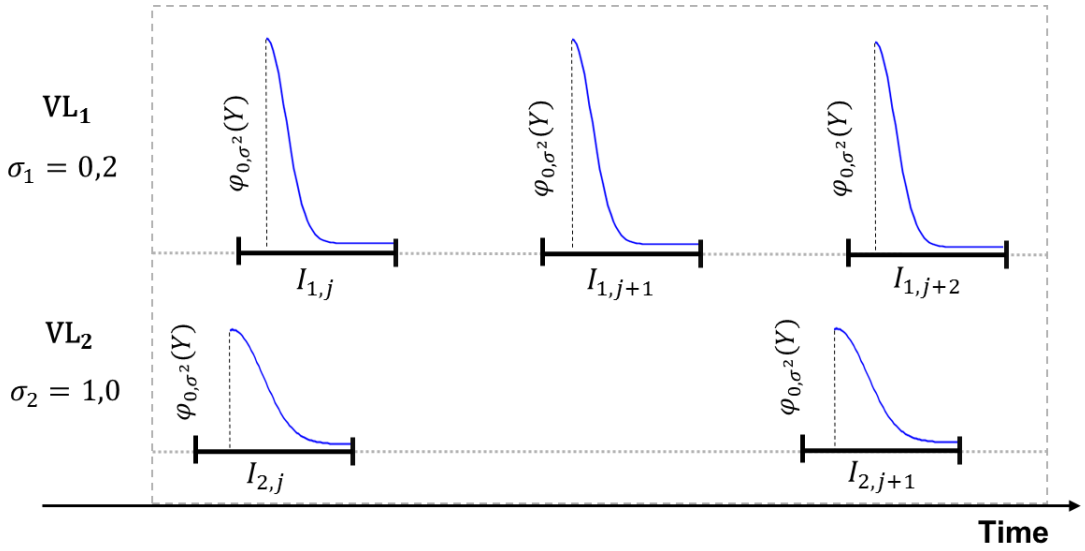


Figure 54 – Représentation des densités de probabilité des tirages de $\tau_{i,j}^{Re}$ de suivant la loi de Laplace-Gauss pour deux VLs et pour $\sigma_1 = 0{,}2$ et $\sigma_2 = 1{,}0$.

Enfin, avant de valider le flux de trames généré, il faut s'assurer que l'ordonnancement des trames proposé ne contient pas de chevauchements entre des trames de différents VLs puisqu'une seule trame à la fois peut être reçue sur le lien physique. L'algorithme parcourt la liste des dates $\tau_{i,j}^{Rs}$ et des dates $\tau_{i,j}^{Re}$, et effectue des comparaisons de proche en proche pour détecter les cas de chevauchement entre les trames successives.

Si un tel cas est détecté, un retard supplémentaire est appliqué à la trame suivante pour assurer une durée minimale ε entre les deux trames consécutives sur le lien physique, comme spécifié dans l'ARINC 664 (3).

4.2.5 Comptage des SBFs

Une fois le flux de trames généré, il ne reste plus qu'à traiter ce flux pour identifier tous les SBFs. Pour un jeu de positions relatives et un jeu de tirages des dates $\tau_{i,j}^{Re}$ donnés, le flux de trames comprend quelques centaines de trames. Une comparaison de proche en proche des dates $\tau_{i,j}^{Re}$ et $\tau_{i,j+1}^{Rs}$ par rapport à la durée minimale ε entre deux trames successives indique si ces trames sont accolées :

$$\tau_{i,j+1}^{Rs} - \tau_{i,j}^{Re} \begin{cases} > \varepsilon & \Rightarrow \text{non accolées} \\ = \varepsilon & \Rightarrow \quad \text{accolées} \end{cases}$$

Où ε vaut précisément 1,60 µs (3). Les SBFs sont comptabilisés sous la forme d'un tableau avec le nombre de trames accolées et le nombre d'occurrences relevées.

Une fois un flux de trames traité pour un jeu de positions relatives et un jeu de tirages des dates $\tau_{i,j}^{Re}$ donnés, le flux est supprimé et l'algorithme recommence à tirer des positions relatives pour les intervalles. Ce rebouclage se produit un nombre de fois égal au nombre de cycles de simulation k spécifié par l'utilisateur.

4.3 Recherche de SBFs sur les flux de trames générés avec la méthode probabiliste

La méthode probabiliste de recherche de SBFs basée sur le modèle par intervalles a été présentée dans la section précédente. À présent, nous nous intéressons à la mesure des probabilités d'occurrence de SBFs. Pour cela, nous commençons par étudier l'influence des paramètres CTRES (BAG et $L_{i,\max}$) sur la charge du lien physique. Puis, nous réalisons plusieurs simulations pour comptabiliser les SBFs de différentes CTRESs dont les charges varient, à partir de l'implémentation de la méthode probabiliste en langage C.

Enfin, la dernière partie introduit une étude comparative entre la méthode de construction du LSBF pessimiste du chapitre 2 et la méthode probabiliste afin d'estimer la pertinence de la prise en compte du LSBF pour le dimensionnement de la mémoire de l'ES de réception.

4.3.1 Influence du BAG et du $L_{i,\max}$ sur la charge

Nous commençons par rappeler la définition de la charge du lien physique (linkload) entre le commutateur final et l'ES de réception :

$$\text{linkload} = \frac{\sum_{i=1}^{n} \left(\frac{\text{BAG}_{\max}}{\text{BAG}_i} \times L_{i,\max} \times C \right)}{\text{BAG}_{\max}}$$

Il faut considérer la définition de la charge comme une couche d'abstraction sur les paramètres de la CTRES (nombre de VLs, BAG et $L_{i,\max}$). La charge permet à l'utilisateur de se rendre rapidement compte si la quantité de trames reçues par unité de temps est important ou non.

D'après la définition mathématique de la charge, celle-ci dépend des BAG_i et des $L_{i,\max}$ associés à chaque VL_i de la CTRES (la vitesse C de transmission du réseau est constante). Ainsi, nous voyons immédiatement que de faibles valeurs de BAG_i et de grandes valeurs de $L_{i,\max}$ conduisent à une charge importante.

Pour confirmer cela, nous reprenons la CTRES industrielle présentée dans le chapitre 2 (Tableau 5) dont la charge est de 15,0%.

BAG (ms)	Number of VLs
2	**2**
4	**3**
8	**5**
16	**9**
32	**10**
64	**11**
128	**10**

$L_{i,\max}$ (bytes)	Number of VLs
[64 ; 150]	**22**
[151 ; 300]	**13**
[301 ; 600]	**8**
[601 ; 900]	**3**
[901 ; 1200]	**1**
[1201 ; 1518]	**3**

Tableau 5 – Répartition des paramètres CTRES (BAG et $L_{i,\max}$) pour une CTRES de cinquante VLs.

Pour les premiers calculs, le $L_{i,\max}$ des trames varient tandis que le nombre de VLs et les BAGs associés demeurent constants. La même valeur $L_{i,\max}$ est associée à tous les VLs de la CTRES et elle varie entre 80 octets pour de petites trames à 1 500 octets pour des trames plus grandes.

Le Tableau 6 indique les résultats des calculs de charge. Comme attendu, plus les $L_{i,\max}$ sont grands, plus la charge est importante. L'influence de $L_{i,\max}$ sur la charge est importante puisque nous passons d'une charge légère pour 80 octets à une charge très lourde pour 1 500 octets.

$L_{i,\max}$ (octets)	Linkload
80	**2,24%**
400	**11,20%**
800	**22,40%**
1 500	**42,00%**

Tableau 6 – Valeurs de la charge du lien pour une variation du $L_{i,\max}$.

Les valeurs de BAG_i sont ensuite modifiées pour les cinquante VLs et les $L_{i,\max}$ initiaux. De même que pour la valeur des $L_{i,\max}$, la même valeur de BAG est appliquée à tous les VLs de la CTRES et elle varie de la valeur minimale (2,0 ms) à la valeur maximale (128 ms) autorisée par l'ARINC 664. Les résultats des calculs de charge sont indiqués dans le Tableau 7.

BAG_i (ms)	Linkload
2,0	**61,50%**
8,0	**15,37%**
32	**3,84%**
128	**0,96%**

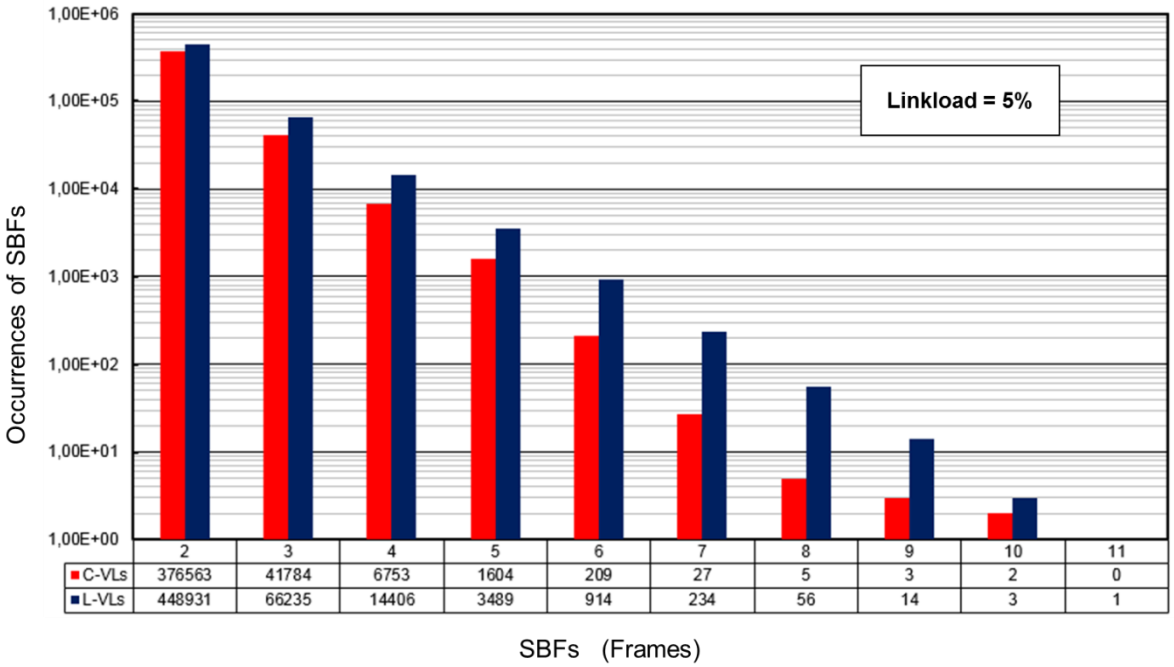
Tableau 7 – Valeurs de la charge du lien pour une variation du BAG.

Pour une variation du BAG, la charge a une amplitude considérable de près de 64%. Pour le cas extrême d'un BAG général de 2,0 ms, la charge est de 64,0%, bien que cette CTRES entre en contradiction avec l'hypothèse 2.3. Il faut retenir qu'une diminution des valeurs de BAGs a pour effet une augmentation de la charge.

4.3.2 Occurrence de SBF pour différentes charges

Dans cette partie, nous présentons des résultats de simulation obtenus à partir de l'implémentation de la méthode probabiliste dans un simulateur codé en C. Les simulations sont réalisées dans les conditions décrites dans les sections précédentes, pour rappel : un nombre d'hyperpériodes N_{hyp} et un nombre de cycles de simulation k choisis de manière à générer plusieurs millions de trames dans le flux entrant pour une CTRES donnée.

La première CTRES étudiée génère un flux de trames entrant dont la charge est de 5%, soit une charge légère. Pour cette CTRES (et les suivantes), nous étudions deux jeux de valeurs d'écart-type σ_i. En effet, pour la génération du flux, chaque VL_i est associé à une valeur σ_i arbitraire. Nous rappelons que, suivant la valeur de σ_i, le VL_i est considéré comme *court* (C-VL) ou *long* (L-VL) en raison du nombre de commutateurs traversés et de la charge des liens parcourus. Les deux jeux utilisés comprennent pour l'un : 90% de C-VLs et 10% de L-VLs (en rouge), et pour l'autre : 10% de C-VLs et 90% de L-VLs (en bleu), de manière à considérer les extrema des dispersions des dates $\tau_{i,j}^{Re}$.



	2	3	4	5	6	7	8	9	10	11
C-VLs	376563	41784	6753	1604	209	27	5	3	2	0
L-VLs	448931	66235	14406	3489	914	234	56	14	3	1

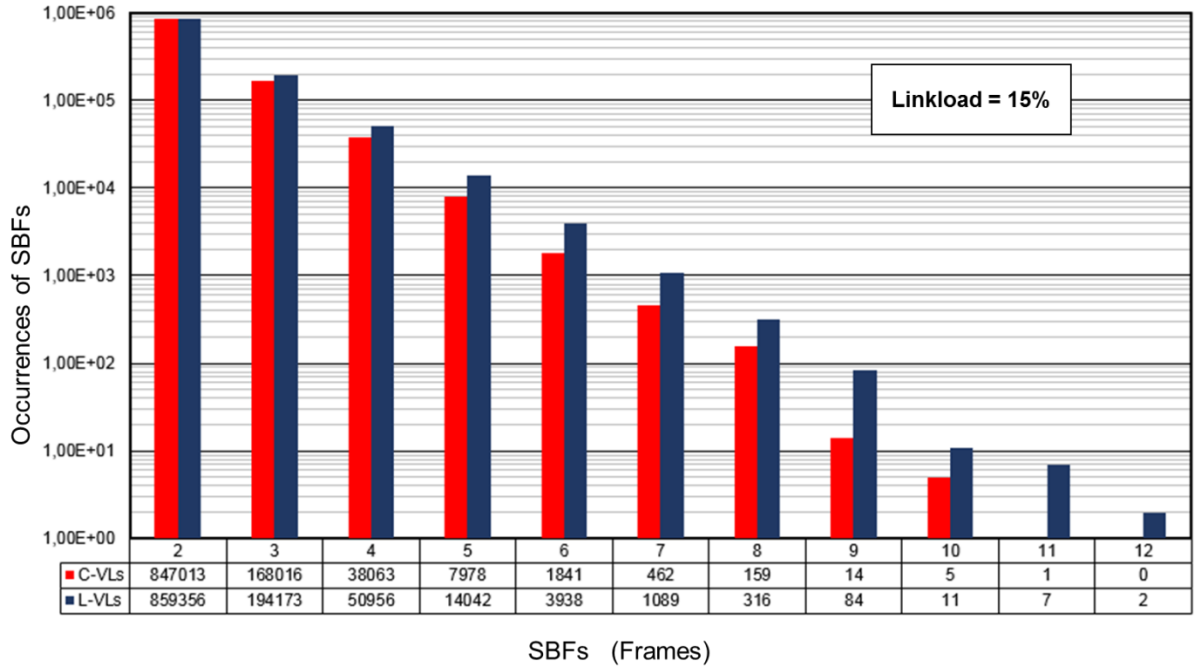
Figure 55 – Occurrences de SBF pour une charge de 5%.

La Figure 55 présente les occurrences de SBF pour la première CTRES avec une charge de 5% et deux jeux de σ_i. De façon générale, le nombre de SBFs est plus élevé pour la configuration avec une majorité de L-VLs. Cela s'explique par le fait que les dates $\tau_{i,j}^{Re}$ sont réparties de façon plus dispersée

sur leur intervalle de réception et elles sont donc moins concentrées sur le début des intervalles. Les flux sont donc moins réguliers et davantage de SBFs peuvent apparaître.

Les occurrences des SBFs sont représentées sous la forme d'histogramme sur une échelle logarithmique. Les SBFs les plus fréquents sont constitués de deux trames (plus de 10 000 occurrences), ce qui est assez intuitif. Les occurrences décroissent ensuite et nous remarquons que le LSBF, soit la plus longue séquence de trames accolées, est de dix trames. Cependant, le nombre d'occurrences du LSBF est seulement d'une trame pour la configuration basée sur les C-VLs et de deux pour la configuration basée sur les L-VLs. Ramené aux millions de trames de l'ensemble des flux d'entrée, il résulte une probabilité d'occurrence de l'ordre de 10^{-6}.

La deuxième CTRES génère un flux de trames en réception dont la charge est de 15%, soit une charge modérée. La Figure 56 présente les occurrences de SBF pour cette CTRES pour deux jeux de σ_i (les mêmes que pour la configuration précédente). À nouveau, nous observons que le nombre de SBFs est plus élevé pour la configuration avec une majorité de L-VLs, pour la même raison de dispersion des dates $\tau_{i,j}^{Re}$.

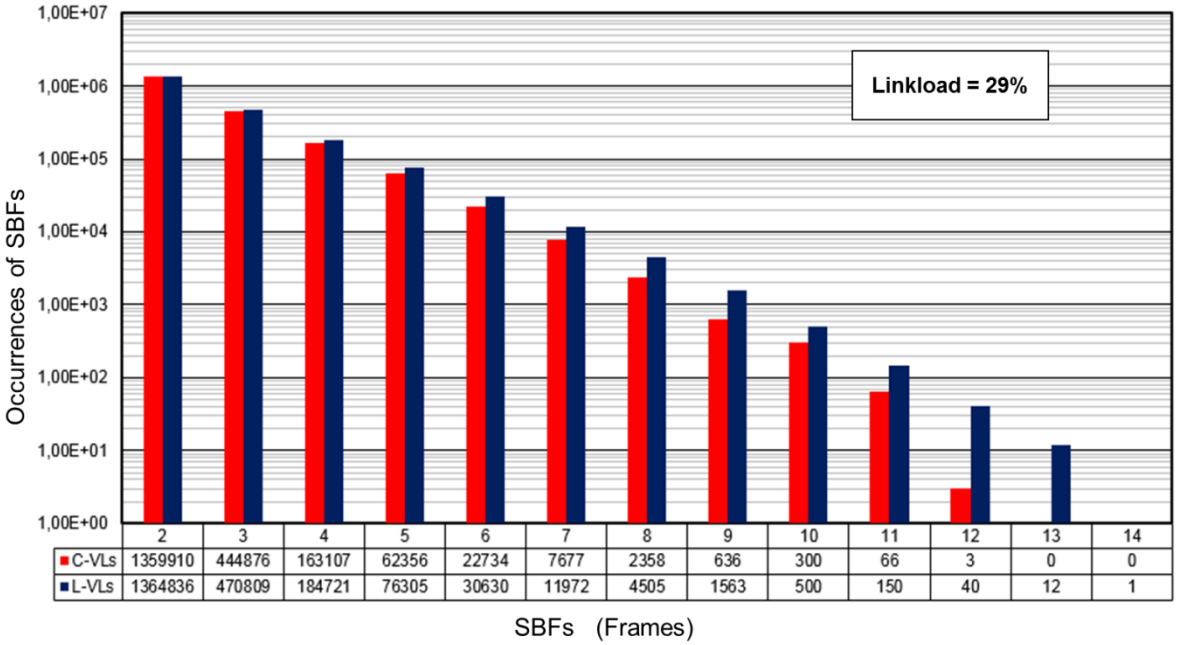


	2	3	4	5	6	7	8	9	10	11	12
C-VLs	847013	168016	38063	7978	1841	462	159	14	5	1	0
L-VLs	859356	194173	50956	14042	3938	1089	316	84	11	7	2

Figure 56 – Occurrences de SBF pour une charge de 15%.

L'allure générale de l'histogramme est similaire à celui de la Figure 55 : les SBFs les plus fréquents sont constitués de deux trames et les occurrences de SBF décroissent lorsque le nombre de trames dans le SBF augmente. Le LSBF de cette CTRES est de douze trames mais uniquement pour le jeu de σ_i ayant une majorité de L-VLs. Le nombre d'occurrences du LSBF est à nouveau très faible puisqu'il est seulement d'une occurrence pour des millions de trames. Sa probabilité d'occurrence est donc de l'ordre de 10^{-6}.

La troisième et dernière CTRES génère un flux de trames entrant dont la charge est de 29%, soit une charge lourde. La Figure 57 présente les occurrences de SBF de différente taille pour cette CTRES pour les deux jeux de σ_i habituels. Le nombre d'occurrences de SBF, toutes tailles confondues, est plus élevé pour la configuration ayant une majorité de L-VLs à cause de la plus grande dispersion des dates $\tau_{i,j}^{Re}$.



	2	3	4	5	6	7	8	9	10	11	12	13	14
C-VLs	1359910	444876	163107	62356	22734	7677	2358	636	300	66	3	0	0
L-VLs	1364836	470809	184721	76305	30630	11972	4505	1563	500	150	40	12	1

Figure 57 – Occurrences de SBF pour une charge de 29%.

Pas de grands changements par rapport aux deux figures précédentes si ce n'est que le LSBF de cette CTRES est de treize trames pour le jeu de σ_i ayant une majorité de L-VLs. Le nombre d'occurrences du LSBF est un peu plus élevé puisqu'il est d'une dizaine d'occurrences pour des millions de trames. Sa probabilité d'occurrence est donc de l'ordre de 10^{-5}, ce qui est un peu supérieur aux probabilités précédentes.

Pour résumer, le nombre d'occurrences de SBF par rapport au nombre total de trames des flux générés est assez élevé et la distribution des valeurs de σ_i a un impact modéré sur ces occurrences. Le Tableau 8 indique le pourcentage de trames incluses dans un SBF pour les différentes CTRESs proposées par rapport à l'ensemble des trames.

Linkload	**Number of frames in SBFs compared to the total number of frames**	
	C-VLs	L-VLs
5%	**4,8%**	**5,9%**
15%	**11,9%**	**12,5%**
29%	**17,4%**	**18,1%**

Tableau 8 – Taux de trames appartenant à un SBF pour les CTRESs simulées.

Nous constatons que, quel que soit le jeu de σ_i choisi, le pourcentage de trames incluses dans un SBF est relativement important. D'après notre méthode probabiliste de recherche de SBFs, il est donc fréquent de rencontrer des SBFs dans le flux entrant et ceux-ci sont généralement de petite taille. Les LSBFs observés comprennent également un nombre de trames faible par rapport à ce que nous avons pu établir avec les méthodes de construction du LSBF dans la grande partie précédente.

L'objet de la partie suivante est d'ailleurs de comparer les LSBFs obtenus avec la méthode de construction du LSBF basée sur le modèle pessimiste et ceux obtenus avec la méthode probabiliste de recherche de SBFs.

4.3.3 Occurrence des LSBFs pour différentes CTRES

Cette dernière partie présente une synthèse de nombreuses simulations réalisées sur différentes CTRESs dans le but d'établir une comparaison entre les LSBFs construits sur la base du modèle pessimiste et les LSBFs obtenus avec la méthode probabiliste.

L'intérêt de cette comparaison est de provoquer une discussion sur la pertinence de la prise en compte du LSBF construit sur la base de modèles estimant le pire scénario de réception pour le flux entrant pour le dimensionnement de la mémoire de réception.

4.3.3.1 Influence du nombre de VLs sur le LSBF

Cette première vague de simulations porte sur le nombre de trames composant le LSBF et sa probabilité d'occurrence en fonction du nombre de VLs de la CTRES. Ces résultats sont établis avec la méthode probabiliste, sur la base de la CTRES industrielle.

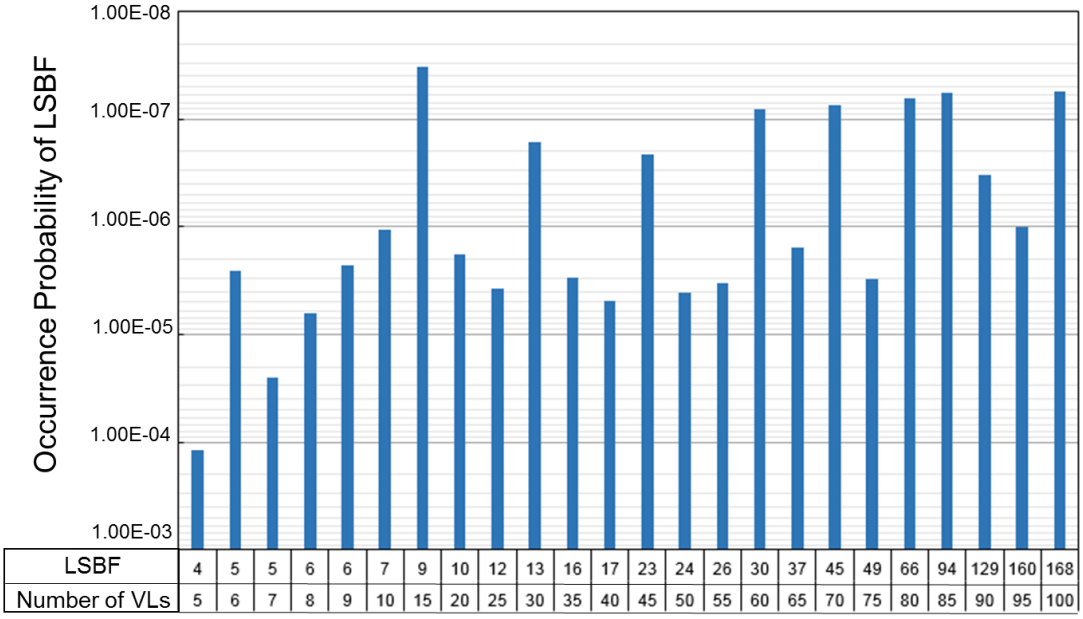


Figure 58 – Probabilité d'occurrence de LSBFs en fonction du nombre de VLs.

La Figure 58 indique le nombre de trames dans le LSBF dans le tableau sous la figure, ainsi que les probabilités d'occurrence des LSBFs sous la forme d'un histogramme en barres. Le paramètre variable choisi pour les CTRESs est le nombre de VLs puisqu'il s'agit du paramètre CTRES qui influence le plus le calcul du LSBF. Par ailleurs, pour la CTRES industrielle de cinquante VLs, le BAG moyen de l'ensemble des VLs est de 50,1 ms pour un écart-type de 43,9 et le $L_{i,\max}$ moyen est de 310,2 octets pour un écart-type de 317,8. Les simulations portent sur des dizaines de millions de trames afin d'être représentatives.

En dessous de huit VLs, il est relativement fréquent de rencontrer un LSBF dont la taille est légèrement inférieure au nombre de VLs. À partir de huit VLs, les probabilités d'occurrence sont inférieures à 10^{-6} mais nous ne pouvons pas dégager de tendance à la hausse ou à la baisse avec l'augmentation du nombre de VLs. L'allure des probabilités est plutôt en dents de scie. C'est là toute la difficulté inhérente à l'estimation du LSBF qui dépend fortement des CTRESs utilisées et donc des couples $\left(\mathrm{BAG}_i\ ;\ L_{i,\max}\right)$.

4.3.3.2 Comparaison des LSBFs obtenus avec la méthode pessimiste et avec la méthode probabiliste

Passons à présent à la comparaison des méthodes. Les résultats de la méthode probabiliste sont comparés avec ceux obtenus par la méthode de construction du LSBF basée sur le modèle pessimiste. Sachant que la méthode probabiliste repose sur le modèle par intervalles, il aurait été intéressant de la comparer aussi la méthode de construction du LSBF basée sur le modèle par intervalles. Mais ceci n'a pas pu être réalisé parce que ces travaux ont été faits avant que la méthode de construction basée sur le modèle par intervalles soit mise au point. En revanche, il pourrait être intéressant de mener cette comparaison dans de futurs travaux.

La Figure 59 propose une comparaison des LSBFs calculés par la méthode probabiliste (histogramme bleu) et par la méthode de construction pessimiste (courbe rouge). Pour chaque CTRES, nous calculons la charge (courbe violette) dont l'échelle est représentée sur l'axe vertical de droite.

Nous observons des charges légères (moins de 10%) pour un nombre inférieur à dix VLs et des charges très lourdes à partir de quarante VLs. Nous avons poussé la valeur de la charge jusqu'à quasiment 90%, ce qui est en dehors du cadre de l'hypothèse 2.3 mais qui donne une idée sur l'évolution de la charge, notamment pour des CTRESs comprenant un nombre important de VLs (quatre-vingt et plus).

La première observation est que les LSBFs obtenus avec la méthode probabiliste de recherche de SBFs sont de taille inférieure aux LSBFs établis avec la méthode de construction pessimiste pour toutes les CTRESs. Cela conforte l'idée que les LSBFs construits se produisent très rarement, puisque nous ne sommes même pas parvenus à les observer sur des millions de trames.

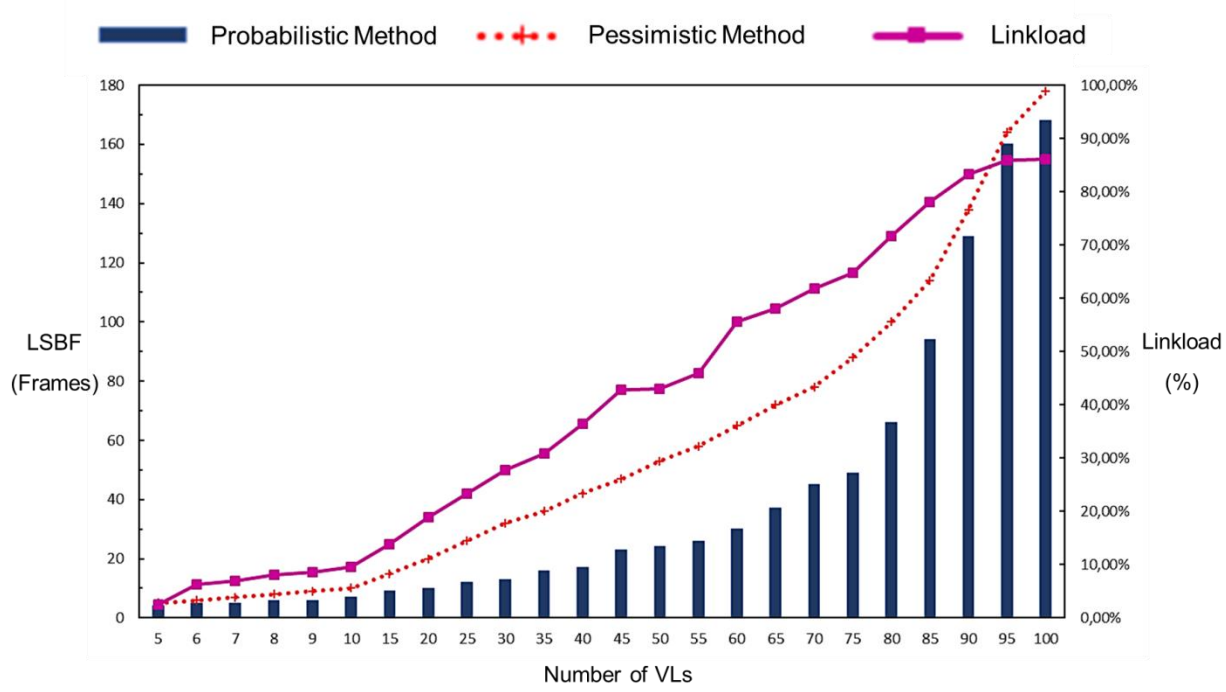


Figure 59 – Taille des LSBFs calculés avec la méthode de construction pessimiste et avec la méthode probabiliste en fonction du nombre de VLs.

Pour moins de dix VLs, les résultats sont proches : le nombre de trames dans les LSBFs obtenus par chacune des méthodes est quasiment égal.

Entre dix et quatre-vingt VLs, les probabilités d'occurrence des LSBFs mesurées sur la Figure 58 sont inférieures à 10^{-6}. De plus, sur la Figure 59 pour cet intervalle de nombre de VLs, la taille du LSBF issue de la méthode probabiliste atteint seulement 50% de la taille du LSBF issue de la méthode de construction pessimiste. Ce résultat est intéressant parce qu'il remet en question la pertinence de la prise en compte du LSBF pessimiste pour établir la taille de la mémoire de réception. En effet, si la probabilité d'occurrence du LSBF pessimiste est très faible, nous pourrions considérer un cas « moins pire » pour dimensionner la mémoire.

La question est alors d'estimer la probabilité à partir de laquelle nous considérons que le LSBF a une probabilité suffisamment faible pour ne pas être pris en compte. Bien entendu, l'intérêt de considérer un cas moins pire est la réduction de la taille de la mémoire de réception. Le seuil de probabilité n'est pas tranché dans cette étude puisqu'il relève de critères quantitatifs de fiabilité pour la certification de l'ES.

4.4 Conclusion

Se limiter à des estimations du pire scénario pour le flux de trames entrant dans la mémoire de réception, et donc à des méthodes de construction du LSBF, ne permet pas de se rendre compte de la fréquence d'apparition des LSBFs construits. Si le LSBF résulte de la concomitance de nombreux phénomènes temporels, il est possible qu'il se produise de manière extrêmement rare. La quantification de la rareté de cet événement requiert un changement de paradigme dans lequel nous prendrions en considération l'aspect « probabilités de réalisation des SBFs ». Ainsi, nous avons proposé une méthode probabiliste basée sur le modèle par intervalles pour générer un flux de trames réaliste en entrée de l'ES de réception et pour rechercher toutes les occurrences de SBFs.

Les étapes clefs de cette méthode sont le tirage uniforme des positions relatives des intervalles sur les différents VLs de la CTRES et le tirage selon la loi de Laplace-Gauss des dates de réception totale des trames sur chaque intervalle. Le choix de ces distributions de probabilité a été justifié avec soin. De plus, la définition de la charge a permis une meilleure appréhension des CTRESs en donnant un repère quantitatif sur la taille des flux générés à partir d'une CTRES.

Les résultats obtenus par l'implémentation en C de la méthode probabiliste et par simulation de nombreuses CTRESs sur des millions de trames, ont fait apparaître que les LSBFs obtenus avec la méthode probabiliste sont généralement plus petits que ceux obtenus à l'aide de la méthode de construction pessimiste (de l'ordre de 50%).

En outre, les probabilités d'occurrence des LSBFs sont inférieures à 10^{-6}. Par conséquent, en supposant que le LSBF obtenu pour une CTRES particulière conduit au WFB dans la mémoire de réception, nous avons ouvert une discussion sur la pertinence de la prise en compte du LSBF pessimiste pour le dimensionnement de la mémoire.

Quatrième partie

Compression sans perte de trame AFDX dans la mémoire de réception ES

Chapitre 5 : Réduction de taille mémoire par implémentation matérielle d'un code de compression sans perte

Dans les parties précédentes, nous avons proposé des méthodes de construction du LSBF et une méthode probabiliste de recherche de SBFs dans le flux de trames entrant dans la mémoire de l'ES de réception. L'objectif de ces méthodes était d'estimer les pires scénarios de réception pour le flux entrant afin d'en déduire le WFB et la taille optimale de la mémoire de réception, et ce à partir de n'importe quelle CTRES.

Dans ce dernier chapitre, la problématique du dimensionnement de la mémoire est abordée sous un nouvel angle. Nous explorons une possible réduction de la taille mémoire par compression sans perte des trames placées à l'intérieur de celle-ci. Pour cela, l'idée consiste à implémenter un ensemble encodeur/décodeur respectivement en amont et en aval de la mémoire, directement au cœur de l'ES de réception. Dans le contexte AFDX, un code de compression pourrait apporter à la fois un gain pour la réduction de la mémoire de réception ES, mais aussi un gain pour la bande passante requise pour la transmission des trames sur le réseau. Toutefois, compte tenu du positionnement du problème, nous présentons uniquement une solution de réduction de la taille de la mémoire de réception.

Déterminer la solution adéquate pour faire de la compression de trames AFDX est l'objet de ce chapitre. Dans la Section 5.1, nous commençons par fixer le choix d'un mode de représentation des trames AFDX ainsi qu'une méthode de génération de trames basée sur une distribution non-uniforme. Quelques éléments fondamentaux de la théorie des sources de codages sont présentés. La Section 5.2 dresse un état de l'art des méthodes de compression et critique leur utilisation potentielle dans le contexte AFDX. La Section 5.3 se concentre sur un code prometteur, le code LZW, dont nous proposons une implémentation matérielle de l'ensemble encodeur/décodeur afin d'évaluer le gain de compression dans la Section 5.4. Enfin, la Section 5.5 présente une conclusion de ce chapitre.

5.1 Éléments théoriques de la compression de données et trame AFDX

La compression de données est une opération de codage qui consiste à transformer une séquence de bits A en une séquence de bits B plus courte et contenant les mêmes informations ou des informations voisines. La séquence A peut être restituée à partir de la séquence B au moyen d'une opération de décompression, qui est l'opération inverse de la compression. Le principe de la compression repose sur l'exploitation de la redondance de bits de la séquence A. En réduisant cette redondance, le code de compression peut créer une séquence B plus courte que la séquence A.

La compression de données est très largement utilisée dans des technologies courantes comme la compression de fichiers audio, de photos, de vidéos, l'archivage de données, la transmission de flux streaming, etc… Dès lors qu'il s'agit d'un système embarqué (appareils photos numériques haute résolution, caméras vidéo HD, baladeurs MP3), la mémoire de stockage est limitée, et l'amélioration permanente de la qualité des prises de vue ou des fichiers audio conduit à des besoins de stockage très élevés. Une méthode de compression adéquate permet de stocker davantage d'information dans un même espace mémoire que si l'information n'est pas compressée. De même, en transmission de données, le facteur limitant est la bande passante et une méthode de compression permet de réduire la bande passante requise pour la transmission de données.

Dans cette section, quelques définitions et propriétés théoriques de la compression de données sont adaptées au contexte des trames AFDX, et nous proposons une analyse des trames afin d'identifier les sources de redondance et de proposer une méthode de génération de trames basée sur des tirages aléatoires.

5.1.1 Codage de trames

Avant de s'attaquer à la formalisation théorique de la compression de données, l'information manipulée doit être formalisée. Nous ne nous intéressons pas au contenu des trames mais à la forme et à la taille des informations afin de pouvoir la stocker dans la mémoire de réception.

Pour ce faire, il convient de choisir un *alphabet* pour représenter l'information contenue dans une trame. L'ARINC 664 indique que la représentation unitaire d'une trame est le bit. Une trame est donc composée de plusieurs centaines de bits et les longueurs de trame $L_{i,\max}$ autorisées se situent entre 64 et 1 518 octets, soit entre 512 et 12 144 bits. Ce premier alphabet est l'alphabet de la logique booléenne composé de ce que nous appelons les *symboles sources* $\{0\ ;\ 1\}$ regroupés dans un ensemble non vide Ω tel que $\Omega = \{0\ ;\ 1\}$.

Or, représenter une trame avec l'alphabet booléen est peu pratique. Pour faciliter la représentation des trames, nous proposons plutôt l'utilisation de l'alphabet hexadécimal (en base 16) dont chaque symbole source est associé à un code unique de quatre bits. Dans le domaine de la compression, l'alphabet hexadécimal est couramment utilisé pour illustrer les méthodes de compression (63) parce

qu'il est plus facile à appréhender par l'utilisateur que l'alphabet booléen. Ainsi, l'ensemble des symboles sources représentant l'information contenue dans une trame est noté Ω_{HEX} et il contient les symboles $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

5.1.2 Définition d'un code de compression

Nous disposons de l'alphabet hexadécimal pour représenter les trames. Pour rappel, chaque symbole source de l'alphabet hexadécimal est codé sur quatre bits. Dans cette partie, nous définissons mathématiquement un code de compression en nous basant sur la représentation hexadécimale choisie.

Définition 5.1 : *Tout code de compression peut être considéré comme une application C de Ω vers U où Ω est l'ensemble des symboles sources et U un ensemble de mots de code basé sur un alphabet binaire.*

La définition 5.1 s'écrit donc :

$$\Omega \longmapsto U$$

$$\mathrm{x} \longmapsto \mathrm{C}(x)$$

Cette première définition est très générale et il faut l'adapter à la représentation des trames en séquences de symboles sources hexadécimaux. En effet, une trame n'est pas considérée comme un symbole source unique mais comme une séquence finie de symboles sources hexadécimaux. Ainsi, la compression s'applique non pas sur la trame entière mais sur des séquences plus ou moins longues de symboles sources.

Nous prenons Ω_{HEX} comme l'ensemble fini des séquences de symboles sources hexadécimaux composant une trame, et U^+ comme l'ensemble des mots de code associés à chaque séquence de symboles sources.

Nous obtenons l'application C^+ de Ω_{HEX} vers U^+ :

$$\Omega_{\mathrm{HEX}} \longmapsto \mathrm{U}^+$$

$$\mathrm{x} \longmapsto \mathrm{C}^+(x)$$

Pour résumer, un code de compression C^+ associe à chaque symbole source ou séquence de symboles sources x, un mot de code $C^+(x)$ de longueur variable $l(x)$, et ce sur l'ensemble des symboles sources de la trame.

La performance du code de compression C^+ peut être mesurée à l'aide de l'espérance $L(C)$. L'espérance $L(C)$ est la longueur moyenne des symboles de l'ensemble U^+, c'est-à-dire la longueur moyenne des mots de code $l(x)$ après compression multipliée par leur probabilité d'occurrence $p(x)$:

$$L(C) = \sum_{x \epsilon \Omega} p(x) l(x)$$

Enfin, la Figure 60 rassemble les termes liés aux codes de compression :

Figure 60 – Termes des codes de compression.

Tout l'intérêt du code de compression est de proposer des séquences de mots de code globalement plus courts en nombre de bits que les séquences de symboles sources. Dans ce cas, l'espérance $L(C)$ de la longueur des mots de code après compression est plus petite que celle des symboles sources avant compression. Le gain de compression est alors positif.

5.1.3 Propriétés d'un code de compression

Nous présentons dans cette partie des propriétés fondamentales des codes de compression : la propriété de bijection de Ω_{HEX} vers U^+et le caractère préfixe d'un code de compression.

D'abord, l'utilisation d'un code est possible si et seulement si chaque mot de code de l'ensemble U^+ possède un unique antécédent dans l'ensemble Ω_{HEX}. On parle alors de bijection de Ω_{HEX} vers U^+ :

Propriété 5.1 : *Un code de compression est uniquement décodable si et seulement si C^+ est une bijection de Ω_{HEX} vers U^+.*

Ce qui revient à écrire :

$$\mathrm{C}^+ \textit{est une bijection de } \Omega_{\text{HEX}} \textit{ vers } U^+ \quad \Leftrightarrow \quad \forall\, x \,\epsilon\, \Omega_{\text{HEX}} \;, \exists\, !\, y \,\epsilon\, U^+ \text{ tel que } y = \mathrm{C}^+(x)$$

Ensuite, la propriété 5.2 présente le caractère préfixe d'un code de compression :

Propriété 5.2 : *Un code de compression est préfixe si et seulement si aucun mot de code n'est le préfixe d'un autre mot de code. Cela revient à dire qu'aucun mot de code ne peut se prolonger pour donner un autre mot.*

Cette propriété est très utile pour décoder des mots de code créés par des codes de compression à longueur variable puisqu'il ne s'agit pas de décoder un seul symbole mais une séquence de symboles placés les uns à la suite des autres, sans avoir recours à des séparateurs. Par ailleurs, les codes non préfixes à l'origine peuvent devenir préfixes par le placement de séparateurs entre deux mots de code consécutifs. Les codes de compression à longueur fixe sont intrinsèquement préfixes.

Enfin, nous présentons l'inégalité de Kraft. Il s'agit d'une condition nécessaire et suffisante pour qu'un code de compression C non préfixe puisse être transformé en un code préfixe équivalent au sens de la longueur des mots.

Propriété 5.3 : *Inégalité de Kraft :*

$$\sum_{i \in \Omega}^{n} \left(\frac{1}{r}\right)^{l_i} \leq 1$$

Où :

- i désigne l'$i^{\text{ème}}$ symbole source de l'ensemble Ω.
- l_i est la longueur de l'$i^{\text{ème}}$ symbole source (en bits).
- r est la taille de l'alphabet des symboles sources.

L'inégalité de Kraft permet de s'assurer qu'il existe un code préfixe pour un code de compression donné. Si l'inégalité est satisfaite, il est possible de construire le code préfixe en utilisant un arbre dont la profondeur maximale est égale à la longueur maximale des mots dans la nouvelle base d'encodage. Un arbre n'est à priori pas unique : différentes constructions peuvent conduire à des codes différents mais dont la distribution des longueurs reste la même.

5.1.4 Modèle pour la génération de trames

À présent, nous analysons la façon dont les trames AFDX sont formées afin d'exploiter les sources de redondances de symboles sources. Comme nous ne disposons pas de statistiques sur la distribution des symboles sources au sein des trames basées sur des cas pratiques industriels, l'évaluation de la distribution des symboles sources est basée sur une étude de l'ARINC 664 et des champs de trame.

5.1.4.1 Analyse de la structure des trames AFDX

Comme l'indique la Figure 61, la structure des trames AFDX présente certaines similitudes, en particulier au niveau des en-têtes (en anglais : *headers*). Ceci s'explique par le fait que les trames que nous étudions sont destinées au même ES de réception, et donc les champs des en-têtes contiennent des informations identiques comme par exemple l'adresse de destination dans l'en-tête MAC. En outre, les trames AFDX ont généralement une taille inférieure à 300 octets. Comme la taille des en-têtes est fixe et comprend 47 octets (3), les en-têtes constituent une part substantielle de la trame.

À cela s'ajoute le fait qu'un ES de réception traite des trames provenant toujours des mêmes ESs sources car la CTRES est définie de manière statique. Ceci exacerbe la redondance des symboles sources, et notamment pour les trames appartenant au même VL.

Enfin, la charge utile (en anglais : *payload*) des trames comprend également un certain degré de redondance et donc il semble raisonnable de supposer que la nature des données échangées sur un VL

reste la même lors des opérations du réseau. Par conséquent, la redondance de symboles sources peut être identifiée tant sur les en-têtes que sur les payloads de trame.

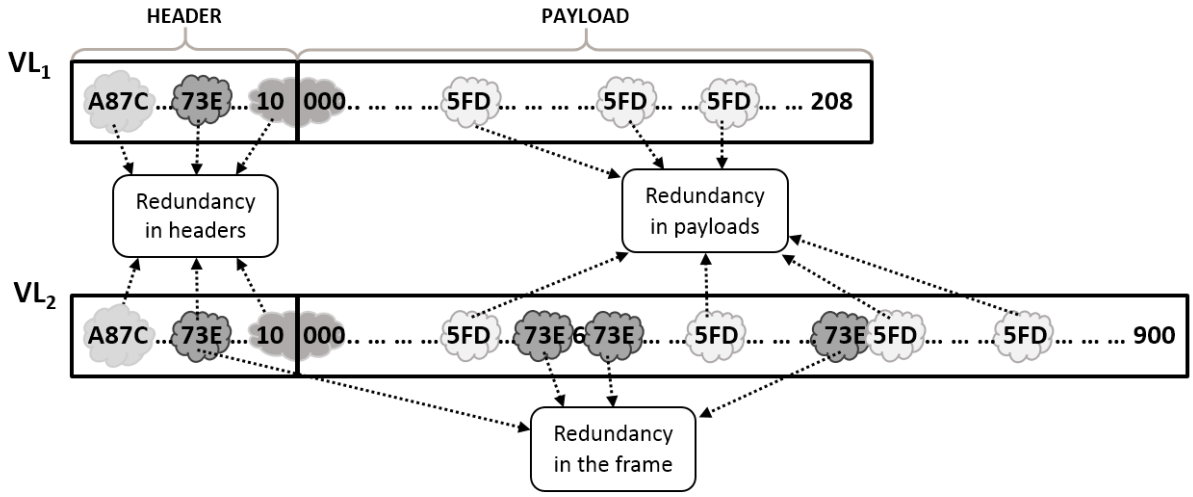


Figure 61 – Redondance de séquences des symboles sources dans une trame et entre deux trames dans les en-têtes et dans les payloads.

5.1.4.2 Distribution uniforme

Une distribution uniforme des symboles sources constituant une trame est la méthode présentant l'entropie de source la plus élevée. En d'autres termes, une trame est construite à partir d'une série de tirages uniformes de symboles sources parmi les seize chiffres hexadécimaux de l'alphabet Ω_{HEX}. La probabilité de tirer un symbole source particulier est donc de $\frac{1}{16}$. Ainsi, un grand nombre de tirages entraîne des nombres d'occurrence pour chaque symbole source assez proches, sans à priori sur l'ordre des symboles sources dans la trame.

La distribution uniforme est la manière la plus générale de générer une trame AFDX. L'inconvénient de la distribution uniforme est que l'entropie (ou le degré de désorganisation) est maximale, de sorte que le gain de compression est généralement petit ou même négatif, et ce quelle que soit la méthode de compression adoptée puisque la redondance est minimale. D'après l'analyse de la structure des trames AFDX, l'identification de points de redondance conduit à écarter le choix d'une distribution uniforme pour la génération des trames.

5.1.4.3 Distribution non-uniforme et génération de trames

Une distribution non-uniforme offre une entropie de source réduite par rapport à une distribution uniforme. Ainsi, le gain de compression potentiel augmente puisque la redondance des symboles sources augmente.

Les points précédents justifient l'utilisation d'une distribution non-uniforme des symboles sources. Cependant, même si l'écart entre une distribution uniforme et une distribution non-uniforme est

difficile à évaluer avec des critères statistiques, nous supposons que la distribution utilisée dans cette étude est relativement proche d'une distribution uniforme.

La méthode de génération des trames se base sur un nombre fini de listes qui contiennent des séquences aléatoires de symboles sources d'une longueur comprise entre un et quatre symboles sources, suivant le numéro de liste. Ces séquences aléatoires sont générées par tirage uniforme de chaque symbole source. Les listes sont au nombre de quatre et elles comportent toutes le même nombre de symboles sources, mise à part la première liste qui ne contient que les seize symboles sources de l'alphabet hexadécimal.

Pour générer une trame, nous devons choisir la valeur des quatre coefficients de pondération associés à chaque liste, qui définissent la probabilité d'aller piocher une séquence dans telle ou telle liste. De cette façon, plus les coefficients de pondération sont élevés pour les listes de séquences longues (trois et quatre symboles sources), plus la redondance de la trame est élevée.

Cette méthode de génération est mise en œuvre dans un générateur de trames AFDX codé en C. Ce générateur de trames permettra de tester l'efficacité du code de compression choisi.

5.2 Étude pratique de codes de compression

Le domaine de la compression de données est très vaste : depuis les premières formulations théoriques dans les années 1960, des milliers d'articles scientifiques ont été publiés à ce sujet. Dans cette section, nous présentons les deux catégories d'algorithme de compression avant de nous focaliser sur les algorithmes de compression sans perte. Les principes de compression des principaux algorithmes de compression sans perte sont expliqués à l'aide d'exemples, puis nous évaluons leur utilisation possible dans le contexte AFDX.

5.2.1 Code de compression avec et sans perte

Tout d'abord, les algorithmes de compression se distinguent selon une caractéristique de premier ordre : la perte ou non d'informations lors du processus de compression. La compression avec pertes offre des taux de compression qui sont généralement meilleurs que ceux obtenus avec la compression sans perte, tout en présentant des vitesses de compression et de décompression satisfaisantes par rapport au domaine d'utilisation.

Le Tableau 9 présente quelques algorithmes avec pertes. Il s'agit principalement de transformées en ondelettes, de transformée en cosinus discrète (en anglais : **DCT (Discrete Cosine Transform)**) et de transformée de Fourier discrète (en anglais : **DFT (Discrete Fourier Transform)**).

La compression par ondelettes consiste à décomposer une image en plusieurs images de résolution inférieure en considérant des espaces d'approximation de moins en moins précis à partir d'une ondelette mère. Un seuil de détails est choisi lors de l'étape de quantification qui conduit à une perte

de données. Cette transformée est particulièrement adaptée à la compression d'images et elle est employée notamment dans le format normalisé JPEG 2000.

La transformée de Fourier discrète (DFT) et la transformée en cosinus discrète (DCT) sont des méthodes de compression du son ou d'images assez similaires : elles reposent sur le fait que l'information est portée par des coefficients correspondant à des basses fréquences et que la plupart des hautes fréquences sont imperceptibles pour un utilisateur humain. Ainsi, en définissant correctement des pas de quantification, la précision des coefficients est réduite, ce qui conduit aussi à une perte d'information. L'avantage est que l'information nécessite moins de bits pour être codée.

Sans Perte	Codage entropique	Code de Golomb-Rice Code de Huffman Code Arithmétique Code Gamma / Delta / Omega
	Codage par plages	Code Run Length Encoding (RLE)
	Transformation	Transformée de Burrows-Weeler (BWT)
	Codage à dictionnaire	Code de Lempel-Ziv-Welch (LZW)
Avec Pertes	Transformations	Transformée en ondelettes Transformée en cosinus discrète (DCT) Transformée de Fourier discrète (DFT)

Tableau 9 – Exemple d'algorithmes de compression parmi les plus courants.

Le principal inconvénient des algorithmes avec pertes est la perte irréversible de données lors de la compression. Si cela est acceptable dans une certaine mesure pour la compression d'images, de vidéos ou de fichiers audio, le contexte critique du réseau AFDX interdit la perte d'information en termes de bits de trame AFDX. Les conséquences d'une altération d'une trame peuvent être catastrophiques et mettre à mal la sécurité des passagers.

Cela nous conduit à ne considérer dans cette étude que les méthodes de compression sans perte afin de respecter l'intégrité des trames reçues au niveau de l'ES de réception. Les méthodes de compression sans perte se déclinent en quatre familles : le codage entropique, le codage par plages, le codage par dictionnaire et des transformations.

5.2.2 Codage entropique

Développé dans les années 1940, le codage entropique regroupe un ensemble de techniques fondamentales de compression de données sans perte. Il est présent dans de nombreux codex de compression. Il repose sur le principe qu'un changement de représentation des symboles sources peut conduire à une réduction de l'espace mémoire nécessaire au stockage des données ou à une réduction de la bande passante requise pour la transmission de données.

Le codage entropique s'appuie sur des statistiques sur les symboles de la source pour encoder de nouveaux mots, dont la longueur dépend du nombre d'occurrences des dits symboles. Plus un symbole est fréquemment rencontré et plus son mot encodé devrait être court. Les méthodes de codage qui découlent de ce principe sont donc généralement à longueur variable. Dans les paragraphes suivants, nous présentons quelques codages entropiques et nous étudions leur utilisation possible dans le contexte de la réception de trames AFDX.

5.2.2.1 Code de Golomb-Rice

Le codage de Golomb-Rice est un code de compression entropique sans perte, préfixe et non-ambiguë. Solomon W. Golomb a initié en 1966 une première version de son codage (64) dont l'implémentation a été rendue plus efficace en 1971 grâce aux travaux de Robert F. Rice (65). Le codage Golomb-Rice est utilisé pour les compressions aux formats **ALAC (Apple Lossless Audio Codec)**, *MPEG-4 Audio Lossless Coding* et *Free Lossless Audio Codec* entre autres.

Principe : La première version du code requiert en premier lieu le choix d'un paramètre *m* où *m* est un entier naturel inférieur au plus grand entier naturel issue de la source. Encoder un entier naturel N consiste à effectuer la division euclidienne de N par m, ce qui donne une décomposition de la forme $N = q \times m + r$, où q est le quotient codé en base unaire et *r* le reste codé en base binaire tronqué. La Figure 62 présente la structure d'un mot de code dans la base de Golomb-Rice.

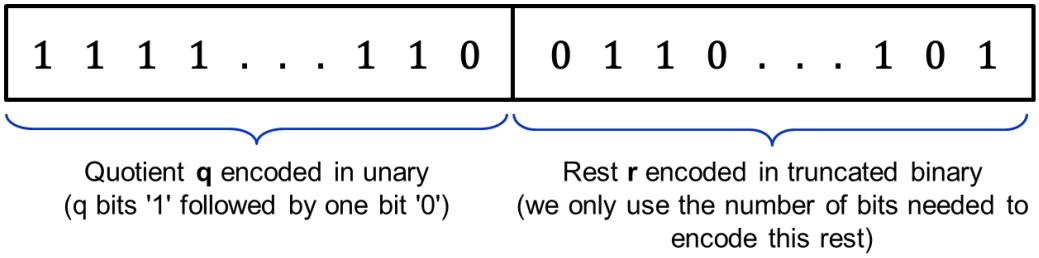


Figure 62 – Structure d'un mot de code suivant l'encodage de Golomb-Rice.

La deuxième version de ce codage élaborée par Rice impose de choisir un paramètre m puissance entière de 2. Autrement dit, si nous voulons coder des mots sources de p bits, nous devons choisir un $m = 2^k$, $k \in [\![0,\ p]\!]$. Cela permet d'obtenir les résultats de la division euclidienne par 2^k plus efficacement grâce à des opérations arithmétiques simples, sachant que le codage binaire de q s'obtient alors en effectuant un décalage de k bits vers la droite et r est obtenu en récupérant les k LSBs de la valeur à encoder.

Pour résumer, si nous considérons la compression d'un entier N codé sur p bits pour un paramètre k, le mot codé sera composé de la partie unaire de longueur $\lfloor N/2^k \rfloor + 1$ bits et du reste en binaire sur k bits. Un exemple d'application de cette méthode pour des valeurs de $N \in [\![0,\ 15]\!]$ sur 4 bits et des paramètres $k \in [\![0,\ 4]\!]$ est présenté dans le Tableau 10.

Il est alors manifeste que la longueur du mot codé est fortement affectée par la valeur du paramètre de compression k. L'enjeu est donc de choisir un paramètre k le plus petit possible suivant la distribution de probabilité d'occurrence des symboles sources.

N	Binary	$m = 2^0$ Codeword	$m = 2^1$ Codeword	$m = 2^2$ Codeword	$m = 2^3$ Codeword	$m = 2^4$ Codeword
0	0000	0	0 0	0 00	0 000	0 0000
1	0001	10	0 1	0 01	0 001	0 0001
2	0010	110	10 0	0 11	0 010	0 0010
3	0011	1110	10 1	10 00	0 011	0 0011
4	0100	11110	110 0	10 01	0 100	0 0100
5	0101	111110	110 1	10 11	0 101	0 0101
6	0110	1111110	1110 0	10 00	0 110	0 0110
7	0111	11111110	1110 1	10 01	0 111	0 0111
8	1000	111111110	11110 0	10 10	10 000	0 1000
9	1001	1111111110	11110 1	10 11	10 001	0 1001
10	1010	11111111110	111110 0	110 00	10 010	0 1010
11	1011	111111111110	111110 1	110 01	10 011	0 1011
12	1100	1111111111110	1111110 0	110 10	10 100	0 1100
13	1101	11111111111110	1111110 1	110 11	10 101	0 1101
14	1110	111111111111110	11111110 0	1110 00	10 110	0 1110
15	1111	1111111111111110	11111110 1	1110 01	10 111	0 1111

Tableau 10 – Table d'encodage pour différentes valeurs de k selon le code de Golomb-Rice.

Application à l'AFDX : Les trames AFDX que nous souhaitons compresser proviennent de différents VLs. Elles sont formées suivant le protocole propre à l'implémentation du réseau avionique. Par conséquent, nous supposons que la distribution des symboles sources est proche d'une distribution uniforme et leur génération indépendante. Cela permet de raisonner de la façon la plus générale possible. Autrement dit, si nous considérons des symboles sources de N bits, nous supposons que la probabilité de rencontrer un symbole particulier est égale à $1/2^N$.

Kiely (66) a introduit un indicateur évaluant l'espérance R_k de la longueur des mots encodés et donc la compression moyenne, en fonction du paramètre k et de la distribution statistique des mots à encoder. R_k représente un taux de compression défini ainsi :

$$R_k = \sum_{j=1}^{+\infty} \left(\left\lfloor \frac{j}{2^k} \right\rfloor + 1 + j \right) Prob(le\ mot\ \text{à}\ coder = j)$$

Sous nos hypothèses et pour des symboles de N bits, l'équation se réécrit de la manière suivante :

$$R_k = \sum_{j=1}^{2^N} \left(\left\lfloor \frac{j}{2^k} \right\rfloor + 1 + j \right) \times \frac{1}{2^N}$$

Il s'avère que R_k admet un minimum et il est atteint pour une certaine valeur du paramètre $k_{opt} = N - 1$. En choisissant un tel k, la moyenne de la longueur des mots codés est donc minimisée, et ainsi la compression est optimale, sans préjuger des symboles constituant les trames AFDX en entrée.

Prenons par exemple un symbole WORD de N bits avec le paramètre de compression optimal au sens de Kiely $k_{opt} = N - 1$. Si l'on regarde de plus près le résultat d'une compression, deux cas de figures sont possibles :

1er cas : $0 \leq WORD < 2^{k_{opt}}$, donc $WORD = 0 \times 2^{k_{opt}} + r$, avec $r \in [\![0,\ 2^{k_{opt}} - 1]\!]$. Le codage de WORD aura la forme représentée sur la Figure 63 :

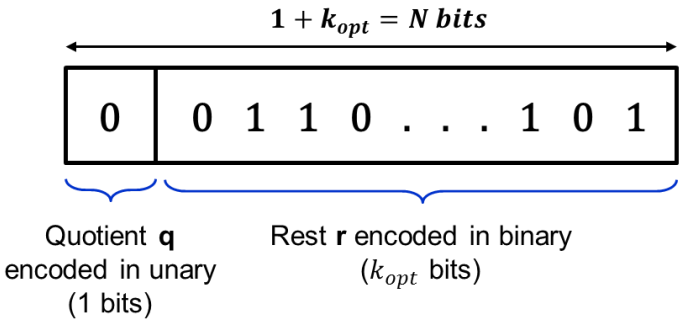


Figure 63 – Mot de code si $0 \leq WORD < 2^{k_{opt}}$ selon le code de Golomb-Rice.

2ième cas : $2^N > WORD \geq 2^{k_{opt}}$, donc $N = 1 \times 2^{k_{opt}} + r$, avec $r \in [\![0,\ 2^{k_{opt}} - 1]\!]$. Le codage de WORD aura la forme représentée sur la Figure 64 :

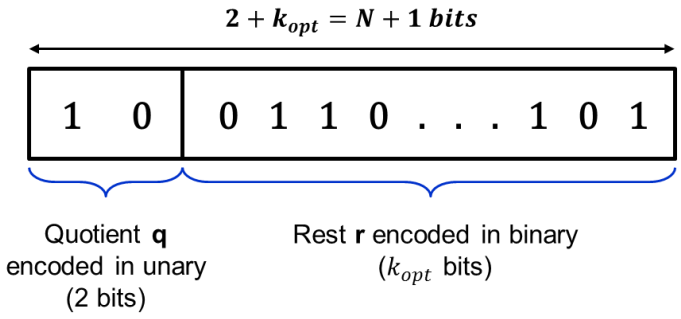


Figure 64 – Mot de code si $2^N > WORD \geq 2^{k_{opt}}$ selon le code de Golomb-Rice.

Nous constatons que dans les deux cas, la longueur du mot codé est supérieure ou égale à la longueur du mot source WORD : le gain en compression est nul et entraîne même un surcoût en termes de bits. Ce résultat s'explique par la distribution des bits dans les trames. La distribution est uniforme, ce qui implique une entropie maximale de la source et conduit à des statistiques uniformes sur l'occurrence des valeurs codées (fréquence des petites et grandes valeurs).

Il est donc difficile d'adapter l'algorithme pour les plages de valeurs fréquentes en dimensionnant le paramètre k. Un k plus petit permettrait notamment de réduire le nombre de bits nécessaires pour encoder le mot source lors de la compression des faibles valeurs mais entraînerait des mots de bits plus longs pour l'encodage des grandes valeurs. Pour être efficace, un tel code de compression doit utiliser des statistiques sur les symboles issus de la source : c'est une caractéristique fondamentale des algorithmes de compression entropique. Nous n'avons pas accès à de telles statistiques, ce qui rend le code de Golomb-Rice inadéquat pour compresser des trames AFDX.

5.2.2.2 Code de Huffman

Les propriétés du code de Huffman sont proches de celles du code de Golomb-Rice puisqu'il s'agit également d'un code préfixe et non-ambigu. David A. Huffman a publié une première version de son codage (67) en 1952. Le code de Huffman est souvent utilisé en complément d'une autre méthode de codage pour améliorer le gain de compression comme dans l'algorithme Deflate (68).

Principe : Le code de Huffman utilise un code préfixe à longueur variable pour représenter un symbole de la source. Il est optimal pour un codage par symbole, c'est-à-dire lorsque la source est figée comme pour la compression d'un fichier. Le code de Huffman repose sur la création d'un arbre de probabilité d'occurrence des symboles sources, un code court étant associé aux symboles les plus fréquents. Il est optimal au sens de la plus courte longueur pour un codage par symbole et pour une distribution de probabilité connue.

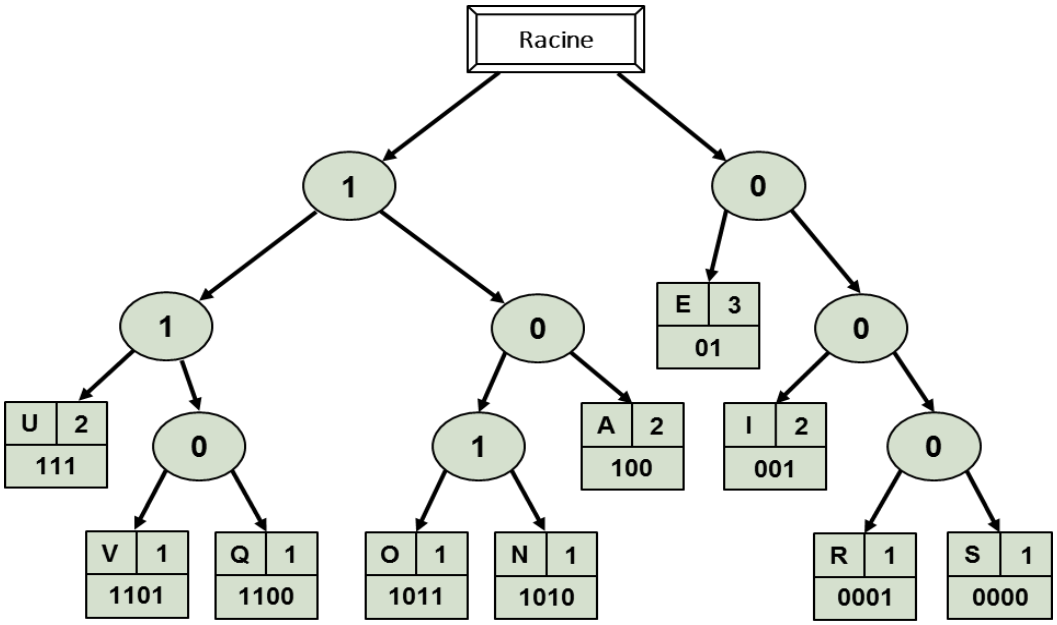


Figure 65 – Exemple d'arbre de Huffman.

Partons d'un exemple où la source de symboles est une chaîne de caractères « RESEAU AVIONIQUE ». La première étape consiste à compter le nombre d'occurrences de chaque lettre soit : 3 E, 2 A, 2 I, 2 U, 1 Q, 1 N, 1 O, 1 V, 1 S et 1 R. Chacun des caractères constitue une feuille de l'arbre de probabilité. Chaque feuille reçoit alors un poids qui vaut le nombre d'occurrences du caractère.

Ensuite, l'arbre est construit en associant les feuilles et nœuds de plus faible poids jusqu'à remonter à la racine (Figure 65). La dernière étape est l'attribution d'un codage à chaque branche de l'arbre en faisant le chemin inverse. En partant de la racine, nous associons à chaque branche une valeur binaire « 0 » pour la branche de droite et « 1 » pour la branche de gauche par exemple. Pour chaque feuille et donc pour chaque caractère, un mot de code est ainsi obtenu. Enfin, le décodage consiste à parcourir l'arbre de codage jusqu'à obtenir une feuille. Le caractère correspondant peut ainsi être reconstitué.

Le codage de « RESEAU AVIONIQUE » soit quinze caractères (espaces non compris) donne en code ASCII un total de 105 bits. Selon l'arbre de la Figure 65, le code de Huffman réduit la séquence de bits à quarante-huit. Nous mesurons un gain de compression de 54% dans cet exemple. Bien sûr, le gain de compression est d'autant plus significatif que les poids de l'arbre sont convenablement attribués. Il existe à ce propos différentes variantes pour la création de l'arbre : statique, semi-adaptative ou adaptative.

La variante statique impose un arbre dont les poids sont attribués hors ligne et qui ne sont pas modifiés en court de fonctionnement. Dans le cas d'un texte en français, le « e » est la lettre qui revient le plus souvent et elle aura donc un code plus court. Pour la phase de décodage, l'arbre n'a pas besoin d'être transmis.

La variante adaptative offre le meilleur taux de compression puisque les poids de l'arbre sont modifiés de façon itérative en cours de fonctionnement de l'algorithme. Ainsi, aucune étude préalable de la source n'est requise pour garantir un taux de compression élevé. En revanche, cette efficacité se paie par un temps de calcul considérablement plus élevé dû aux modifications à apporter à l'arbre durant la phase de compression.

Enfin, la variante semi-adaptative se situe entre les deux variantes précédemment décrites et réclame de lire le fichier à compresser afin d'établir des statistiques sur celui-ci, de calculer les poids de l'arbre puis de compresser le fichier. Le gain engendré par cette analyse n'est pas significatif puisque cette variante réclame la transmission de l'arbre ainsi constitué.

Application à l'AFDX : Le raisonnement reprend les hypothèses émises dans la partie précédente, à savoir que la génération des bits de la trame suit une distribution proche de la distribution uniforme. Ainsi, les variantes statiques et semi-adaptatives ne seraient pas performantes à priori, étant donné l'absence de statistiques sur la constitution des trames. L'entropie de la source contraint le gain à de très faibles valeurs. En revanche, la variante adaptative pourrait présenter un gain plus significatif mais la mise en œuvre d'un tel algorithme est difficile en langage matériel.

5.2.2.3 Code arithmétique

Le code arithmétique est un code statistique qui adapte la longueur du mot encodé en fonction de sa probabilité d'occurrence : plus un symbole source apparaît régulièrement, moins le nombre de bits nécessaires pour le coder est grand. La paternité du code arithmétique reviendrait à G. Langdon aux alentours des années 1980 (69). Le code arithmétique est notamment proposé pour améliorer la compression d'images en noir et blanc (70).

Principe : Le code arithmétique fonctionne suivant un principe proche du code de Huffman dans le sens où il s'agit d'un code préfixe à longueur variable basé sur une étude statistique de l'occurrence des symboles sources. Il est cependant plus efficace que le code de Huffman puisqu'il utilise moins de bits pour stocker un symbole source. Le code arithmétique encode une séquence de symboles « par morceaux » d'un certain nombre de symboles et attribue à chaque symbole un nombre flottant *N*, tandis que le code de Huffman attribue à chaque symbole source un code précis. À titre d'illustration, prenons l'exemple où un symbole représente 90% des symboles sources, la taille optimale du code serait de 0,15 bits. Le code de Huffman devrait encoder ce mot sur au moins un bit tandis que l'utilisation de code flottant permet de code sur des fractions de bits. On dit du code arithmétique qu'il est optimal niveau bit.

Source Symbol	Probability	Interval
F	1/4	[0 ; 0,25[
4	1/2	[0,25 ; 0,75[
A	1/4	[0,75 ; 1[

Tableau 11 – Code arithmétique : exemple de probabilité d'occurrence.

Considérons l'exemple de la séquence de symboles « F4A4 » basée sur un alphabet hexadécimal. L'analyse statistique donne immédiatement la probabilité d'occurrence de chaque symbole source comme indiqué dans le Tableau 11. De plus, chaque symbole se voit affecter un intervalle compris entre 0 et 1 qui dépend de sa probabilité associée.

L'étape suivante consiste à remplacer la séquence de symboles sources par un nombre flottant. Pour ce faire, la séquence de symboles doit être affectée à un intervalle I compris entre 0 et 1. Pour construire I, nous partons de l'intervalle $[0\ ;\ 1[$, puis nous appliquons un algorithme à chaque fois que l'on rencontre un symbole de la séquence comme suit :

- La borne inférieure $\mathrm{I}_{\mathrm{inf}}$ de la séquence source prend le résultat $\mathrm{I}_{\mathrm{inf}} + (\mathrm{I}_{\mathrm{sup}} - \mathrm{I}_{\mathrm{inf}}) \times \mathrm{I}_{\mathrm{inf\_symbole}}$.
- La borne supérieure $\mathrm{I}_{\mathrm{sup}}$ de la séquence source prend le résultat $\mathrm{I}_{\mathrm{inf}} + (\mathrm{I}_{\mathrm{sup}} - \mathrm{I}_{\mathrm{inf}}) \times \mathrm{I}_{\mathrm{sup\_symbole}}$.

Appliqué à l'exemple précédent, nous obtenons le Tableau 12 :

Source Symbol	$\mathrm{I}_{\mathrm{inf}}$	$\mathrm{I}_{\mathrm{sup}}$
	0,0	1,0
F	0,0	0,25
4	0,0625	0,1875
A	0,15625	0,1875
4	0,1640625	0,179668

Tableau 12 – Code arithmétique : création de l'intervalle I.

Ainsi, tout nombre flottant compris entre 0,164 062 5 et 0,179 668 correspond à la séquence de symboles sources « F4A4 », comme par exemple : 0,17.

Pour la phase de décodage, le Tableau 11 doit être connu du décodeur. Il s'agit ensuite de reconstituer la séquence de symboles sources à partir du nombre flottant X_{symb}. Pour cela, la prochaine lettre de la chaine est celle dont l'intervalle contient le nombre flottant. Pour la première itération, il est facile d'établir la correspondance à partir du nombre flottant transmis. Pour les suivantes, il faut appliquer la formule suivante, où P_{lettre} est la probabilité d'occurrence de la lettre précédemment identifiée :

$$\frac{\left(X_{\text{symb}} - I_{\text{inf}}\right)}{P_{\text{lettre}}} = X_{\text{symb}} + 1$$

Et ainsi de suite, jusqu'à reconstituer la totalité de la séquence de symboles sources.

Application à l'AFDX : Le code arithmétique et le code de Huffman sont assez proches en termes de gains optimaux basés sur des études statistiques de la source. De nouveau, nous mettons en avant l'argument de la distribution uniforme et donc une répartition à priori inconnue des symboles sources au sein des trames AFDX. Ceci ne permet pas d'exploiter efficacement les avantages du code arithmétique. De plus, la relative complexité de l'implémentation du code en langage de programmation matérielle est une contre-indication majeure à son implémentation aisée dans un ES.

5.2.2.4 Codes Gamma / Delta / Omega

Les codes Gamma, Delta et Omega sont des codes entropiques sans perte, préfixes et à longueur variable. Ils sont issus des travaux publiés en 1975 par P. Elias (71), l'un des pionniers de la théorie de l'information. Ces codes se distinguent des autres méthodes d'encodage par le fait qu'ils peuvent encoder des symboles codés sur un nombre de bits dont le nombre maximal est à priori inconnu, c'est-à-dire dont la représentation décimale est une valeur relativement grande. La force de ces codes est donc leur taille variable, efficace sur des valeurs petites et ils ne divergent pas trop vite si le nombre à encoder est très grand.

Par ailleurs, il n'est pas nécessaire de connaître les probabilités d'occurrence des symboles sources contrairement au code d'Huffman ou au code arithmétique. Ces codes font partie des codes universels. Dans la suite, nous détaillons brièvement les principes de chacun des codes.

Principe du codage Gamma (γ) : Le code Gamma est le plus simple des trois. La représentation des symboles sources sera une valeur entière naturelle x. Nous posons alors l'inégalité suivante : $2^N \leq x \leq 2^{N+1}$. L'encodage Gamma consiste à écrire N en base unaire (N « 0 » suivis d'un « 1 ») auquel est concaténé le reste $R = x - 2^N$ en binaire. Si nous considérons une séquence de valeurs entières {1 ; 13 ; 3}, le codage en binaire d'une telle séquence impliquerait la connaissance du plus grand nombre et donc du choix du nombre de bits nécessaires à l'encodage, c'est-à-dire {0 0001 ; 1 0000 ; 0 0011}. En appliquant le codage Gamma à cette séquence, nous obtenons : {1 ; 000 1101 ; 011}.

En termes de nombre de bits sur cet exemple simple, le codage Gamma permet un gain de quatre bits, soit de 27%.

Principe du codage Delta (δ) : Le codage Delta se base sur le codage Gamma et il a la propriété d'être asymptotiquement optimal. De même que précédemment, x est encadré par $2^N \leq x \leq 2^{N+1}$. *N+1* est encodé avec le codage Gamma. Au mot constitué s'ajoute les *N* premiers bits de x en binaire naturel. Par exemple, prenons $x = 2^{12} + 15 = 4\,111$. Alors $2^{12} \leq 4111 \leq 2^{13}$. Nous encodons *N* + 1 = 13 en codage Gamma ce qui donne **000 1101**, puis nous accolons les *N* premiers bits de x en binaire naturel soit **1 000 0000 1111**. La représentation Delta est donc **000 1101 0000 0000 1111**. Le constat immédiat est que le mot obtenu est plus long que le mot source. Ainsi, le code Delta ne fournit pas un gain positif pour une seule valeur mais il prouve son efficacité sur des séquences constituées de « petits » termes fréquents et de quelques « grands » termes peu fréquents.

Principe du codage Omega (ω) : Le codage ω est le plus complexe des trois codes présentés. Il est asymptotiquement optimal mais son efficacité se concentre sur les grandes valeurs tandis qu'il est moins efficace sur les petites valeurs. Le codage ω est également appelé codage récursif puisqu'il s'appuie sur un procédé récursif tel que décrit ci-après :

- On place un « 0 » à la fin du code.
- Si $N = 1$, FIN : l'encodage est fini.
- Ecrire N en binaire et le mettre au début du code existant.
- Poser $N = L - 1$, avec *L* la longueur du code binaire de l'étape précédente.
- Retourner à la seconde étape avec le nouveau N.

Ce codage est plus volumineux que les deux précédents. Il est intéressant pour de très grandes valeurs.

Application à l'AFDX : Les calculs de P. Elias donnent les dimensions de ces trois codes. Pour un entier *x*, il faut $L\gamma$, $L\delta$ et $L\omega$ bits afin de les encoder respectivement en code Gamma, Delta et Oméga.

$$L\gamma = 2\lfloor log_2(x) \rfloor + 1$$

$$L\delta = \lfloor log_2(x) \rfloor + 2\lfloor log_2(\lfloor log_2(x) \rfloor + 1) \rfloor + 1$$

$$L\omega \leq \frac{5}{2}\lfloor log_2(x) \rfloor + 1$$

Comme expliqué précédemment, la force de ces codes réside dans leur longueur variable ce qui permet d'encoder les petites valeurs sur relativement peu de bits par rapport à la taille fixe qu'il faudrait en binaire naturel pour coder la plus grande valeur. Néanmoins, dans un réseau AFDX, les valeurs très petites (ex : 0000 0000 0000 0011) ne sont à priori pas majoritaires puisque le tirage des bits suit la loi uniforme. La probabilité pour que les n premiers bits soient à « 0 » vaut $\frac{1}{2^n}$ et elle est donc très faible. À cela s'ajoute la difficulté à implémenter ces méthodes de codage en langage matériel.

En conclusion, les codes Gamma, Delta et Omega ne sont pas de bons candidats pour être utilisés tels quels au sein d'un ES AFDX.

5.2.3 Codage par plages (RLE)

Les origines du code **RLE (Run-Length Encoding)** sont incertaines mais elles émergent comme une méthode de compression efficace à partir des années 1960, les publications scientifiques à leur sujet augmentant de façon significative et coïncidant avec les travaux de D. A. Huffman. Son utilisation première fut pour la transmission par fax (72), puis pour le signal télévisé à partir de 1967 (73). Aujourd'hui, le code RLE est utilisé pour le formatage JPEG ou encore la compression de fichiers BZIP2 qui l'emploie pour deux étapes sur neuf de son complexe algorithme d'encodage.

Principe : Le code RLE est basé sur un principe de décomptage de *N* symboles identiques successifs. Puis, il indique le nombre *N* de symboles suivi du symbole lui-même. *N* peut être codé de différentes manières et conduire à des gains de compression variables. Prenons l'exemple de la séquence de symboles suivante :

AAAA BBBB BBBB BCCC CCCA AAAA AAAA

Si nous considérons chaque symbole codé sur huit bits (ASCII étendu), cette séquence conduit à N_1 = 256 bits pour une transmission ou un stockage en mémoire. Le code RLE appliqué à cette séquence conduit à la nouvelle séquence :

4A 9B 6C 13A

Pour cette séquence, le nombre d'occurrences de chaque symbole est codé sur cinq bits, soit N_2 = 52 bits nécessaires. Immédiatement, le gain de compression η_1 se déduit et donne :

$$\eta_1 = \frac{\mathrm{N}_1 - \mathrm{N}_2}{\mathrm{N}_1} = 81\%$$

À présent, si le nombre d'occurrences de chaque symbole est codé sur deux bits et non cinq, la séquence devient :

4A 4B 4B 1B 4C 2C 4A 4A 4A 1A

Le gain vaut alors :

$$\eta_2 = \frac{\mathrm{N}_1 - \mathrm{N}_2}{\mathrm{N}_1} = 61\%$$

Ainsi, la façon dont le code RLE est paramétré influence considérablement le gain de compression.

Enfin, nous présentons un dernier cas pour lequel la compression suivant le code RLE est défavorable. Prenons la séquence suivante :

ABCD ABCD ABCD ABCD ABCD ABCD ABCD ABCD

Les symboles ABCD se répètent périodiquement et cela conduit à un code RLE comme suit :

1A 1B 1C 1D … 1A 1B 1C 1D (x8)

Il apparaît que la séquence ainsi obtenue par codage RLE est plus longue que la séquence initiale. Le gain de compression résultant est donc négatif puisque la taille de la séquence encodée est plus grande que la taille de la séquence initiale. Le code RLE n'est donc performant que si le fichier ou le message est constitué de longues séquences de symboles identiques.

Application à l'AFDX : L'implémentation d'un système complet encodeur/décodeur RLE au sein d'un ES AFDX est techniquement réalisable puisqu'il présente une complexité algorithmique moindre par rapport à d'autres algorithmes tels que le code de Huffman adaptatif. Il nécessite peu de ressources de calculs et il est relativement rapide tant en compression qu'en décompression.

L'inconvénient majeur du code RLE est que son efficacité repose sur l'apparition de longues séquences de symboles identiques. Or, les symboles constituant la trame suivent une loi proche de la loi uniforme. Quelle que soit la représentation de la trame choisie, les probabilités d'obtenir de longues séquences de symboles identiques sont faibles. Comme la compression RLE peut conduire à une augmentation de la taille de la trame encodée, nous pouvons considérer que sur un grand nombre de trames (et donc de symboles), le gain global sera à priori négatif du fait de l'alternance des symboles sources. Ainsi, le code RLE employé tel quel au sein d'un ES ne permettra pas de gain de compression significatif, sauf cas particuliers. En revanche, le code RLE pourrait être utilisé comme étape intermédiaire ou finale dans un processus de compression complexe, tel qu'il est employé en compression JPEG.

5.2.4 Transformée de Burrows-Wheeler (BWT)

La transformée de Burrows-Wheeler (en anglais : **BWT (Burrows-Wheeler Transform)**) est une technique de réorganisation des données inventée par Michael Burrows et David Wheeler en 1994 (74). Elle est utilisée pour compléter les méthodes de compression telles que le code de Huffman ou le code RLE. Elle intervient notamment pour la compression au format BZIP2 offrant un meilleur taux de compression que la plupart des autres algorithmes de compression, ou encore dans les procédés de réalignement de séquences ADN informatisés (75).

Principe : La transformée BWT d'une séquence S de n bits se déroule en plusieurs étapes comme illustrée sur la Figure 66. D'abord, la séquence S est recopiée sur la première ligne du tableau T_1. Puis, une rotation sur la première ligne donne la deuxième ligne du tableau T_1 et ainsi de suite, jusqu'à obtenir un tableau carré T_1 d'ordre n. L'étape suivante consiste à trier les n lignes du tableau T_1 par ordre lexicographique afin d'obtenir un tableau T_2. Dans le tableau T_2, la séquence initiale S apparaît nécessairement sur au moins une des lignes. Notons l l'indice d'une de ces lignes. La transformée de la séquence S est alors la concaténation du codage de l en binaire avec la séquence formée par la lecture de la dernière colonne de haut en bas.

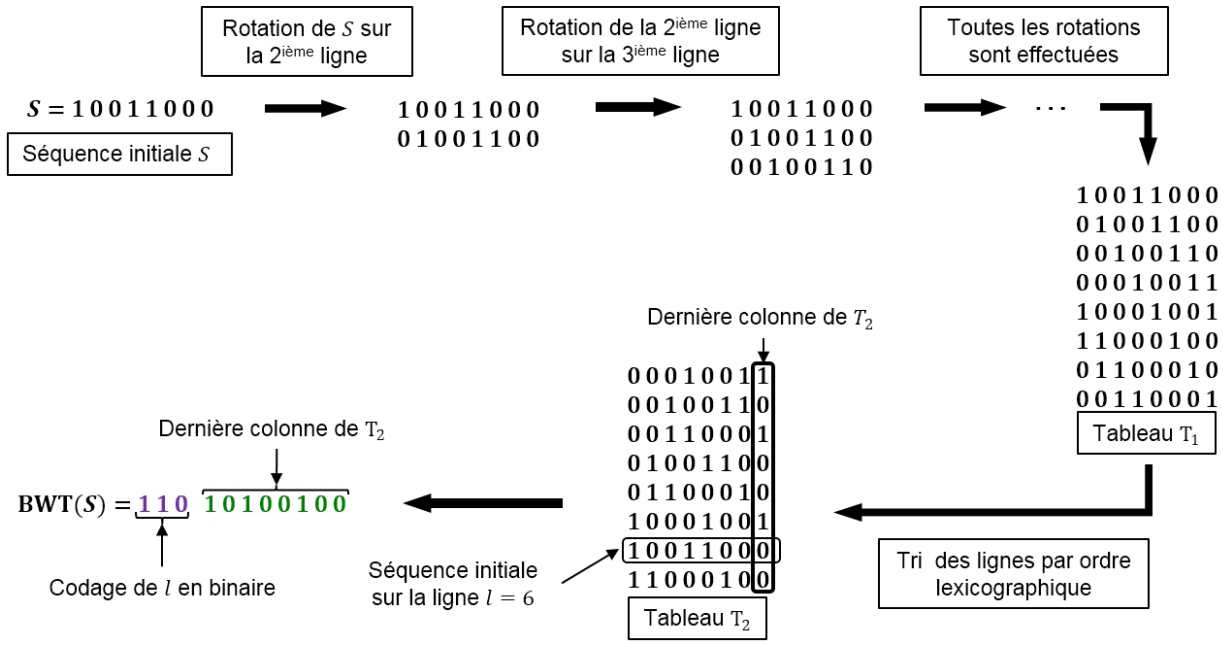


Figure 66 – Exemple de transformée de Burrows-Wheeler.

La transformée BWT n'offre pas de gain de compression direct mais son rôle est plutôt de faire apparaître des successions de symboles identiques de sorte à améliorer la compression d'un autre code (RLE par exemple). La transformée BWT est utilisée lorsqu'il y a des répétitions de plusieurs sous-séquences de symboles au sein de la séquence à encoder. Par exemple, nous considérons la séquence S_1 définie comme la concaténation de huit séquences S, soit :

$$S_1 = SSSSSSSS$$

La transformée BWT de la séquence S_1 donne :

$$\mathrm{BWT}(S_1) = 110000\ \ 11111111\ 00000000\ 11111111\ 00000000\ 00000000\ 11111111\ 00000000\ 0000000$$

Nous remarquons que la seconde partie de $\mathrm{BWT}(S_1)$ correspond à la réplication en huit exemplaires de chaque bit de la seconde partie de $\mathrm{BWT}(S)$. La répétition de la séquence S dans la séquence S_1 entraîne donc la répétition des bits dans sa transformée. Le code RLE appliqué à la séquence S_1 peut ainsi donner un gain de compression substantiel.

Application à l'AFDX : Plusieurs facteurs s'opposent à la mise en œuvre de la transformée BWT dans un ES. Le facteur prépondérant vient du fait que les symboles sources suivent une distribution proche de la distribution uniforme. Les longues répétitions de symboles se répèteront peu de fois, ce qui réduit l'efficacité de la compression qui pourrait suivre la transformée BWT. De manière générale, si un code RLE utilisant p bits pour coder une chaîne de bits identiques est employé après une transformée BWT, il faut que le nombre de chaînes distinctes de bits identiques c dans la séquence transformée vérifie $p \times c < N$ pour que la compression présente un gain positif.

La transformée inverse consiste en une succession de n tris lexicographiques sur des listes de séquences plus longues d'un bit à chaque nouveau tri. Ces algorithmes ont une complexité en

$O(n\log(n))$, augmentant le temps de traitement des séquences lorsque leur taille n augmente. C'est pour cela que la compression au format BZIP2 est connue pour être plus lente que les autres. Une telle lenteur peut nuire aux exigences de déterminisme imposées.

La transformée BWT offre un potentiel de compression plus important que les autres méthodes de compression présentées. Cependant, elle ne peut être utilisée seule puisqu'il s'agit seulement d'une méthode de réorganisation des symboles sources pour accentuer la redondance et non d'une méthode de compression. Si on ajoute à cela la complexité de la transformée BWT, cela en fait une solution peu pratique à mettre en œuvre pour la compression des trames.

5.2.5 Code de Lempel-Ziv-Welch (LZW)

Proposé en 1984, les codes LZW résultent de nombreuses améliorations basées sur les travaux de Lempel, Ziv (76) et Welch (77). Les codes LZW sont relativement légers et faciles à mettre en œuvre à la fois dans des architectures matérielles de bas niveau (78) et dans des logiciels de traitement de haut niveau (79). Différence notable avec les codes précédents : ils ne nécessitent pas d'analyse de symboles sources avant l'opération de compression.

Les codes LZW et leurs dérivés (LZ77, LZ4, LZMW, LZAP, LZJB, etc.) sont largement utilisés, et notamment dans des formats de fichiers comme PDF, GIF ou TIFF.

Principe : Le principe des algorithmes LZW est de compléter une table de codage appelée *dictionnaire.* Le dictionnaire code les séquences de symboles sources en des mots courts de longueur fixe correspondants aux adresses respectives des plages de dictionnaires où les séquences sont stockées.

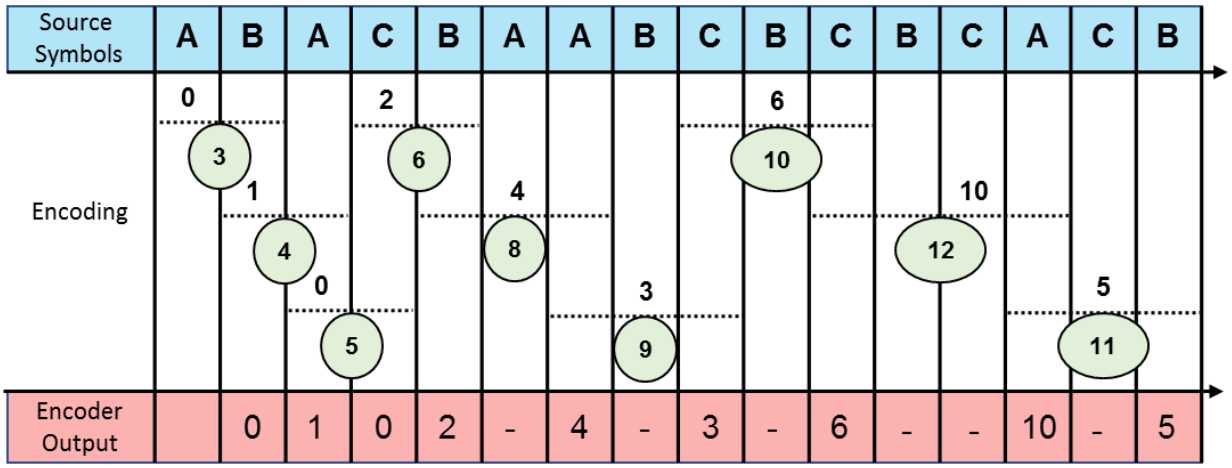


Figure 67 – Exemple simplifié de l'algorithme LZW : réception initiale de symboles sources et séquence encodée de mots de code.

La Figure 67 illustre les mécanismes de compression LZW grâce à un exemple simplifié du code LZW utilisé dans ce document. Les trois symboles sources (**A**, **B** et **C**) sont codés sur quatre bits comme les

caractères de l'alphabet hexadécimal que nous utilisons pour représenter le contenu des trames. La séquence de symboles sources reçues au niveau du compresseur LZW (ou encodeur LZW) est indiquée dans le cadre supérieur de couleur bleu : **{A B A C B A A B C B C B C A C B}**. La séquence de mots de code encodée est indiquée dans le cadre inférieur de couleur rouge : **{0, 1, 0, 2, 4, 3, 6, 10, 5}**.

La Figure 68 représente le dictionnaire divisé en quatre zones. Le dictionnaire s'étend sur une plage d'adresses de quatre bits ou seize adresses possibles et il contient quatre dictionnaires plus petits : le dictionnaire 1 (adresses 0 à 2) comprend les séquences de un symbole (donc l'alphabet initial), le dictionnaire 2 (adresses 3 à 7) comprend les séquences de deux symboles, et ainsi de suite jusqu'au dictionnaire 4 (adresses 12 à 15) (80).

	Dictionary Addresses	Source Symbols			
Dictionary 1	0	0000	A		
	1	0001	B		
	2	0010	C		
Dictionary 2	3	0011	A	B	
	4	0100	B	A	
	5	0101	A	C	
	6	0110	C	B	
	7	0111	...	...	
Dictionary 3	8	1000	B	A	A
	9	1001	A	B	C
	10	1010	C	B	C
	11	1011	A	C	B
Dictionary 4	12	1100	C	B	C	A
	13	1101	...	...	...	...
	14	1110	...	...	...	...
	15	1111	...	...	...	...

Figure 68 – Exemple simplifié de l'algorithme LZW : dictionnaires pour l'encodage des séquences de symboles sources.

Sur la base des Figures 67 et 68, nous détaillons la phase de remplissage des dictionnaires suivant l'algorithme LZW. Initialement, le dictionnaire 1 est pré-rempli avec les symboles sources de base.

La Figure 69 permet de suivre les étapes de l'algorithme LZW. Étape 1 : l'encodeur LZW reçoit le symbole source **A**. Il mémorise ce symbole (étape 2) puisque le symbole source A est déjà présent dans le dictionnaire 1.

Retour à l'étape 1, l'encodeur reçoit le deuxième symbole source **B**. Or, la séquence **AB** n'est pas pas présente dans le dictionnaire puisqu'elle n'a pas encore été rencontrée. Le dictionnaire n'est pas plein puisque nous sommes au démarrage, alors nous passons à l'étape 5 : la séquence **AB** est stockée dans le dictionnaire 2 associée au mot **0011** (adresse **3**) et le mot **0000** (adresse **0**, symbole source **A**) est envoyé à la sortie du codeur.

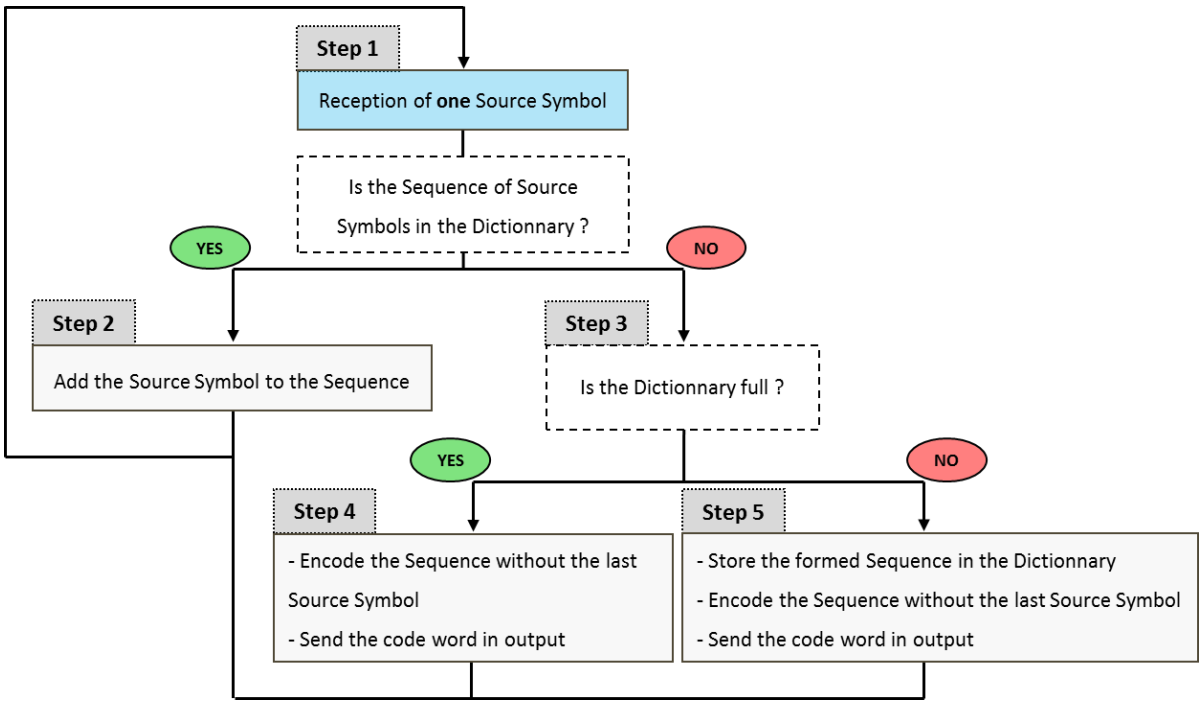


Figure 69 – Algorithme LZW.

Ces étapes sont répétées jusqu'à ce que toutes les adresses de dictionnaires disponibles soient associées à une séquence de symboles sources. Ainsi, la prochaine fois qu'une séquence de symboles sources mémorisée est rencontrée, le mot correspondant (ou l'adresse en binaire) est directement placé en sortie de l'encodeur.

Le gain mémoire provient de la différence entre le nombre de bits utilisés pour coder les symboles sources et le nombre de bits utilisés pour coder les mots de sortie. Dans cet exemple, un symbole source est codé sur quatre bits, donc deux symboles sources représentent huit bits. Or, un mot de code (ou une adresse du dictionnaire) est encodé sur quatre bits. Par conséquent, si une séquence de deux symboles est compressée, le gain est de $(8-4)/4 \times 100 = 50\%$. Plus le nombre de symboles est élevé dans une séquence, plus le gain de mémoire est important.

Application à l'AFDX : Le code LZW diffère des codes de compression sans perte présentés jusque-là par le fait que son gain de compression peut être positif, même sans étude préalable des symboles sources constituant les trames. Le remplissage des dictionnaires est réalisé au fil de l'eau sur le flux de symbole source et non pas hors ligne, de façon à s'adapter aux redondances des séquences de symboles sources.

Si le remplissage des dictionnaires en séquences de symboles sources ne dépend pas de l'utilisateur, il reste la difficulté du choix du nombre de dictionnaires et du paramétrage de la taille de ceux-ci. Les gains de compression résultants pourraient être amenés à varier significativement suivant ces choix et nous devrons étudier ces variations avec attention.

Enfin, dans sa version initiale, le code LZW est suffisamment simple pour permettre une implémentation matérielle au sein de l'ES de réception, en amont et en aval de la mémoire de réception. Sa représentation sous forme d'algorithme se prête bien à une implémentation sur FPGA en VHDL, en utilisant par exemple une machine d'état.

Pour ces différentes raisons, nous avons retenu le code LZW comme algorithme de compression sans perte pour compresser les trames stockées dans la mémoire de réception.

5.3 Implémentation matérielle du code LZW

Dans cette section, nous proposons une adaptation du code LZW pour compresser les trames stockées dans la mémoire de réception. La mise en œuvre du code LZW repose sur un encodeur et un décodeur placés respectivement avant et après la mémoire. Pour mesurer les gains de compression du code LZW adapté au contexte des trames AFX sur différents jeux de trames, nous avons entièrement codé en VHDL un encodeur et un décodeur LZW, que nous avons ensuite testé sur une plateforme matérielle spécialement conçue à cet effet. Nous présentons les architectures conçues dans les parties suivantes.

5.3.1 Encodeur LZW

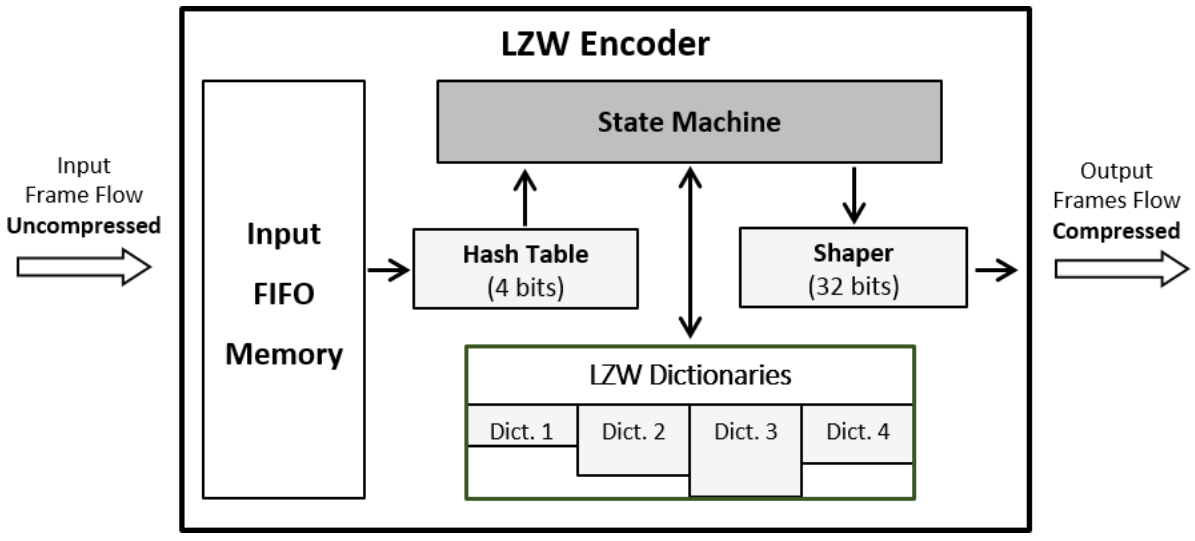


Figure 70 – Représentation par blocs de l'architecture de l'encodeur LZW.

La Figure 70 présente l'architecture matérielle de l'encodeur LZW inspirée des travaux de Ce et al. (81). Le flux de trames entrant entre dans l'encodeur LZW via un bus de communication de largeur 32 bits. Ce flux est stocké dans une mémoire tampon FIFO qui assure la régulation du flux. Ensuite, la table de hachage extrait les symboles sources de quatre bits. Le dictionnaire LZW stocke les séquences de symboles sources et génère les mots encodés sur n bits. Enfin, le bloc de mise en forme de sortie concatène les mots encodés de n bits pour envoyer un mot de 32 bits à la mémoire de réception via un bus de communication.

Les mots encodés sont d'une taille de n bits. Cette taille dépend du nombre de séquences de symboles sources que les quatre dictionnaires peuvent stocker. D'une manière générale, les dictionnaires plus grands permettent une meilleure compression des trames mais au prix d'un coût en ressource mémoire et d'un coût en temps de décompression plus élevés.

La taille des dictionnaires nécessite un compromis entre le gain de compression et l'utilisation des ressources mémoires pour être efficace. En outre, la question de la répartition de la mémoire entre les quatre dictionnaires doit être analysée car le nombre de séquences enregistrées dans chaque dictionnaire a un impact important sur le gain de compression final.

5.3.2 Plateforme matérielle de test

L'évaluation du gain de compression des trames AFDX avec notre compresseur LZW matériel requiert une plateforme matérielle complète que nous présentons dans cette partie.

Pour réaliser une mesure de gain de compression, nous avons besoin de plusieurs ensembles de trames composées de plusieurs millions de symboles sources. Ces ensembles sont créés avec un générateur de trames codé en langage C, présenté dans la première section de ce chapitre. Ainsi, le caractère configurable de ce générateur permet de choisir le degré de redondance que nous souhaitons introduire parmi les symboles sources.

Pour un paramétrage donné du générateur de trames, un ensemble de trames dit *de référence* est généré, puis des ensembles *secondaires* sont créés par la modification de l'ordre des trames de l'ensemble de référence. En effet, l'ordre dans lequel les trames sont reçues est susceptible de modifier les séquences enregistrées dans les dictionnaires, et donc de faire varier le gain mémoire potentiel. L'ensemble de référence et les ensembles secondaires forment un *jeu* de trames.

Après la génération d'un jeu de trames, les trames sont enregistrées dans des fichiers distincts à l'intérieur d'une carte **SD (Secure Digital)** placée sur la carte de test FPGA.

La Figure 71 représente la plateforme matérielle de test implémentée sur FPGA. La plateforme est construite autour d'un processeur softcore NiOS II qui gère l'écriture et la lecture des trames dans le bloc LZW. Il envoie des trames depuis la carte SD au bloc LZW via un bloc d'entrées/sorties parallèles (en anglais : **PIO (Parallel Input Output)**). Les trames sont compressées, puis elles sont stockées dans la mémoire de réception.

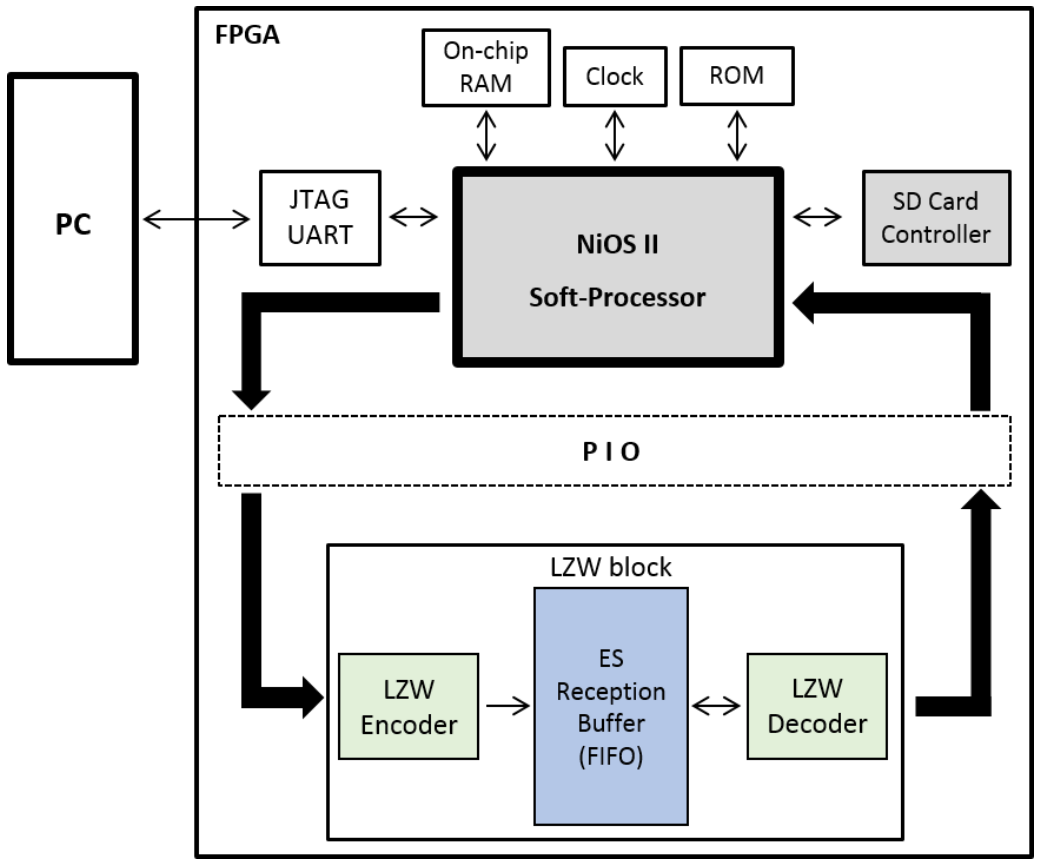


Figure 71 – Plateforme matérielle de test pour évaluer les gain de compression du code LZW.

En lecture, le NiOS II propose deux modes de fonctionnement. Dans le premier mode, il lit les trames compressées directement depuis la mémoire pour évaluer le gain de compression en comparant avec la taille des trames compressées avec la taille des trames de l'ensemble initial. Dans le second mode, il lit les trames compressées via le décodeur LZW. Le décodeur LZW décompresse les trames et leur redonne leur format initial. Le NiOS vérifie alors la validité de la décompression LZW et sa durée. Les résultats sont assemblés dans un rapport envoyé au PC via le JTAG UART.

5.4 Gain de compression apporté par le code LZW

Cette dernière section présente l'ensemble des résultats obtenus lors de nos différentes campagnes de tests. Nous mesurons des gains de compression pour des ensembles de trames dont le degré de redondance varie. Aussi, nous discutons de la manière optimale de partager l'espace mémoire entre les quatre dictionnaires pour maximiser le gain de compression.

5.4.1 Gain de compression pour un jeu de trames standard

Dans cette première phase de mesure, nous analysons le gain de compression d'un jeu de trames standard pour différentes configurations de dictionnaires, à partir d'un jeu de trames standard.

5.4.1.1 Génération du jeu de trames standard

Nous devons générer un jeu de trames dont le niveau de redondance ne soit ni trop grand, ni trop petit mais moyen. Nous appelons ce premier jeu, un jeu de trames *standard*. Vis-à-vis du générateur de trames, nous devons choisir les quatre coefficients de pondération tel que le niveau de redondance résultant soit moyen. Nous rappelons que le générateur de trames est composé de quatre listes de séquences de symboles sources de même taille dont les séquences sont tirées de façon uniforme. La génération d'une trame consiste à venir piocher aléatoirement des séquences de ces listes suivant la valeur des coefficients de pondération. Le jeu standard de trames est produit à partir de coefficients {15 ; 25 ; 20 ; 40}

Comme expliqué précédemment, des ensembles secondaires sont créés à partir de l'ensemble de référence par modification de l'ordre des trames. Ainsi, la redondance globale de l'ensemble n'est pas affectée mais l'ordre de réception des trames agit sur la façon dont les dictionnaires sont remplis. Le gain de compression s'en retrouve donc affecté. Pour chaque ensemble de référence, nous formons **vingt** ensembles secondaires.

5.4.1.2 Mesures de gains

À partir des nombreux tests réalisés, il est apparu que la taille du dictionnaire 4 influence fortement la valeur du gain de compression. C'est pourquoi, la Figure 72 présente quatre graphiques où les gains minimums, moyens et maximaux sont tracés en fonction de la taille du dictionnaire 4. Pour chaque configuration de dictionnaires, nous calculons le gain de compression pour le jeu standard et chaque ensemble de trames permet le calcul d'un gain. Pour une configuration de dictionnaires donnée, nous obtenons un ensemble de valeurs de gains dont le nombre est égal au nombre d'ensembles de trames.

Sur la Figure 72, les graphiques (a), (b), (c) et (d) correspondent respectivement à des tailles de 9, 10, 11 et 12 bits pour le mot de code à la sortie de l'encodeur. Pour moins de 9 bits, le gain de compression est systématiquement négatif. À partir de 9 bits, il apparaît que le dictionnaire 2 doit être de taille maximale (256 séquences avec l'alphabet hexadécimal) pour avoir un gain substantiel. Par conséquent, la plage d'adresse restante est divisée entre les dictionnaires 3 et 4 suivant l'équation :

$$S_3 + S_4 = 2^n - 256 - 16$$

Où S_3 et S_4 sont respectivement la taille des dictionnaires 3 et 4.

D'après les quatre graphiques de la Figure 72, nous obtenons des formes concaves similaires pour le gain de compression. Il apparaît que le gain diminue lorsque le dictionnaire 4 est trop petit. Ensuite, le gain atteint un maximum et diminue à nouveau lorsque la taille du dictionnaire 4 devient trop grande, jusqu'à devenir négatif pour tout nombre de bits de codage. Le meilleur gain est obtenu pour une configuration de codage 10 bits (Figure 72 (b)) avec 22,35%, mais la taille du dictionnaire 4 doit être définie avec précision. Cependant, pour 11 et 12 bits (Figure 72 (c), (d)), le gain est plus faible, mais la plage de valeurs où le gain est quasi-constant est supérieure pour 10 bits. Pour 9 bits (Figure 72 (a)), le gain maximal est inférieur à celui pour 10 bits et il a une forme en cloche avec un maximum à 8,59%.

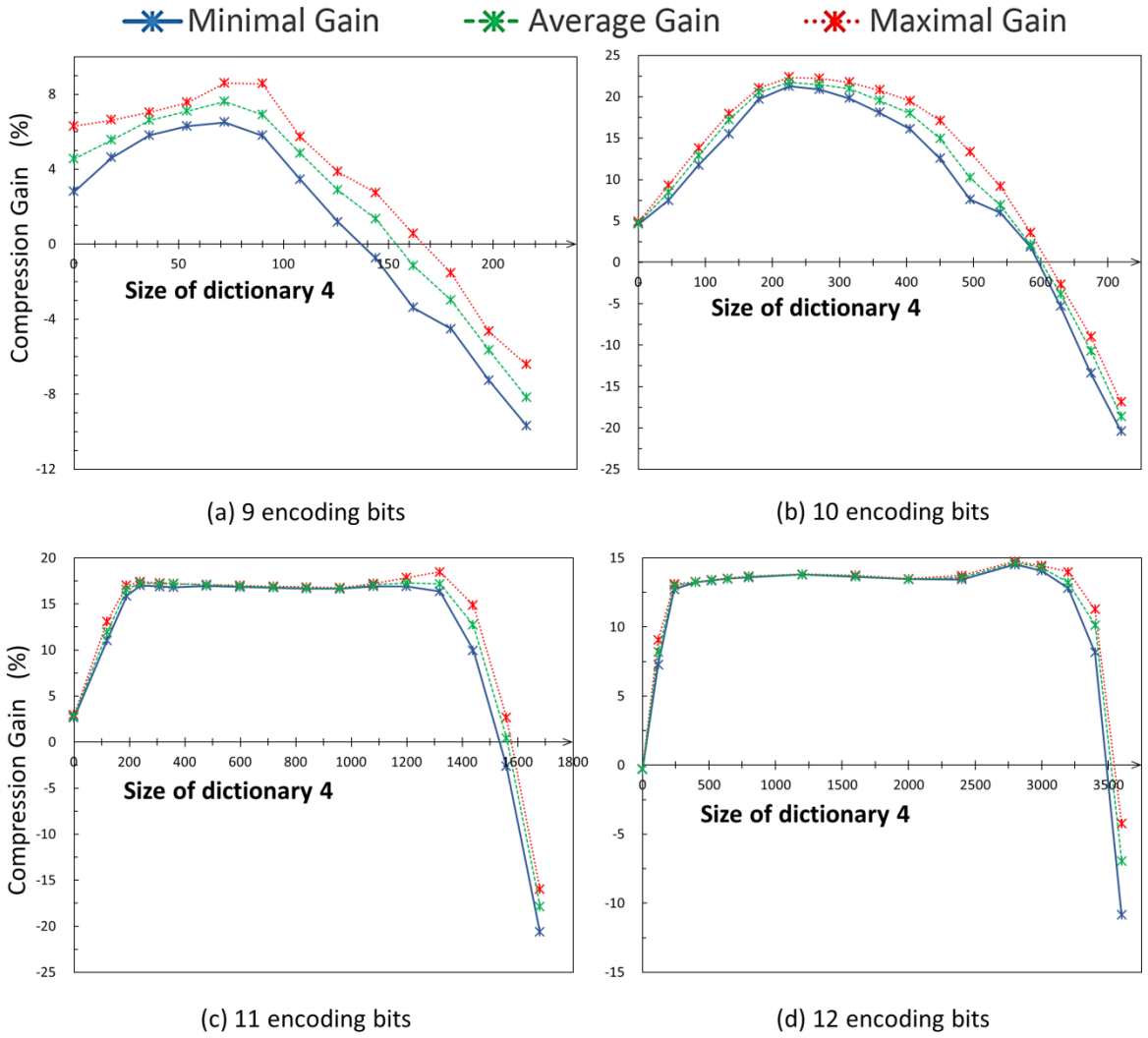


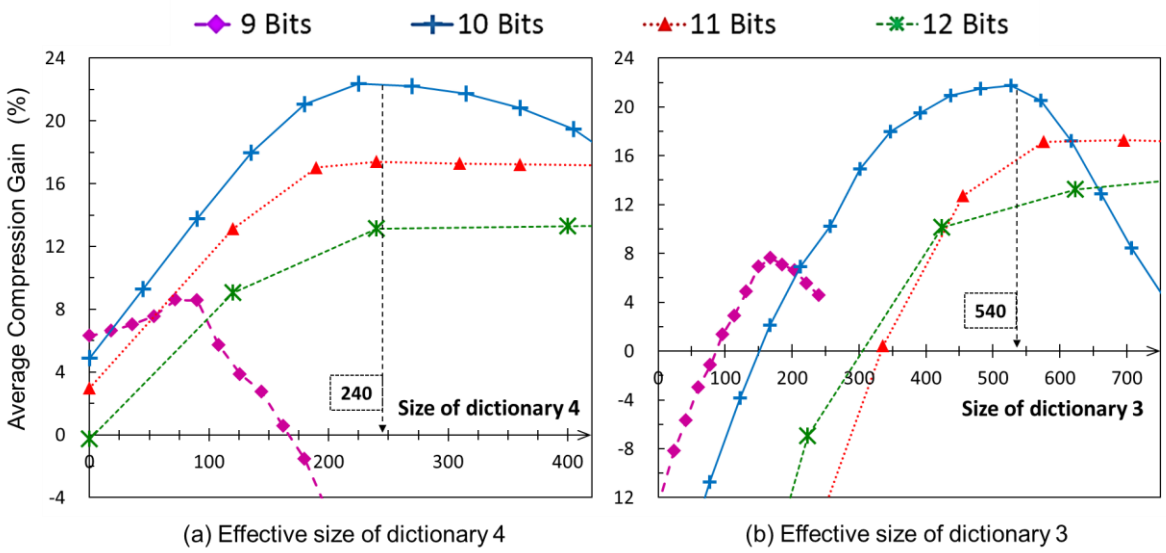
Figure 72 – Gain de compression en fonction de la taille du dictionnaire 4 pour le jeu standard et pour quatre valeurs de bits d'encodage.

Pour résumer, lorsque la taille du dictionnaire 4 est petite, le gain est également petit. Puis, le gain reste constant sur un champ de valeurs. Plus le nombre de bits de d'encodage est élevé, plus ce champ de valeurs est grand. Enfin, le gain diminue à partir d'une certaine taille du dictionnaire 4 jusqu'à être négatif. La hausse du gain est concomitante à la croissance du dictionnaire 4 qui permet la compression la plus élevée sur une séquence donnée de symboles sources. À l'inverse, lorsque le dictionnaire 4 devient trop grand, le dictionnaire 3 n'est plus de taille suffisante pour permettre un enregistrement fréquent de séquences de quatre symboles sources. La partie constante de la courbe de gain correspond à une zone où la taille des dictionnaires 3 et 4 est adéquate.

À partir de ces observations, nous formulons l'hypothèse suivante :

Hypothèse 5.1 : *la configuration optimale du dictionnaire LZW est obtenue lorsque les dictionnaires 3 et 4 sont assez grands et pour la plus petite taille de bits d'encodage permettant cette configuration.*

Pour vérifier cette hypothèse, nous traçons le gain de compression moyen en fonction de la taille du dictionnaire 4 (Figure 73 (a)) et du dictionnaire 3 (Figure 73 (b)) pour les quatre tailles d'encodage mises en évidence précédemment. Nous nous concentrons sur les zones critiques de la Figure 72 qui correspondent à la partie croissante et à la partie décroissante.



(a) Effective size of dictionary 4 (b) Effective size of dictionary 3

Figure 73 – Gain de compression en fonction de la taille du dictionnaire 4 (a) et de la taille du dictionnaire 3 (b) pour le jeu standard et pour quatre valeurs de bits d'encodage.

La Figure 73 fournit les valeurs attendues par rapport celles de la Figure 72. Nous observons le même comportement. Le gain commence par des valeurs négatives et augmente proportionnellement avec la taille des dictionnaires 3 et 4. Il devient rapidement constant pour 11 et 12 bits de codage. La zone constante pour 11 et 12 bits d'encodage correspond à l'espace où les dictionnaires 3 et 4 sont assez grands mais ils ne conduisent pas à un gain optimal en raison du nombre de bits d'encodage requis trop élevé.

Cependant, pour 10 bits d'encodage, la zone constante est étroite et correspond au gain maximal pour l'ensemble des mesures. Pour 9 bits d'encodage, le gain n'a pas de zone constante. Le comportement du gain pour 9 et 10 bits d'encodage est causé par la taille limitée des dictionnaires 3 et 4.

Ces résultats montrent que la compression nécessite une taille minimale pour le dictionnaire 3 avant de considérer l'espace mémoire du dictionnaire 4. Les séquences des symboles sources stockées dans le dictionnaire 4 doivent « dériver » d'une séquence parente du dictionnaire 3. De la même manière, le dictionnaire 3 a des liens héréditaires avec le dictionnaire 2. L'hérédité entre les dictionnaires impose au dictionnaire 2 la taille maximale de 256 séquences de deux symboles qui couvrent toutes les combinaisons possibles.

Types de Ressources Matérielles	Encodeur	Décodeur
Eléments logiques	19 625	236 364
Registres	4 742	12 266

Tableau 13 – Ressources matérielles pour 10 bits d'encodage (Meilleur gain mesuré : 22,35%).

À partir de la Figure 73, nous notons que le gain de compression pour 10, 11 et 12 bits d'encodage commence à être constant pour une taille de 240 pour le dictionnaire 4 et 540 pour le dictionnaire 3. Pour 10 bits d'encodage, le gain offre une courte zone constante parce que la taille minimale requise pour les dictionnaires 3 et 4 est presque unique. Le gain atteint sa valeur maximale de 22,35% pour un coût en ressources matérielles précisé dans le Tableau 13 et un temps de simulation n'excédant par quelques secondes par configuration de dictionnaires. Pour 9 bits d'encodage, la configuration n'est pas pertinente car les tailles des dictionnaires 3 et 4 ne sont pas assez grandes et les gains qui en résultent sont faibles.

5.4.2 Gain de compression en fonction de l'espace mémoire occupé par le dictionnaire 4

Sur les Figure 74, Figure 75 et Figure 76, nous représentons le gain de compression en fonction du pourcentage d'espace mémoire occupé par le dictionnaire 4 par rapport à l'espace mémoire alloué aux dictionnaires 3 et 4. Chaque figure correspond à un jeu de trames généré pour différents coefficients de pondération. Une échelle en pourcentage permet de représenter plusieurs tailles d'encodage sur le même graphique.

L'avantage que présentent ces graphiques est qu'ils sont plus synthétiques que les graphiques précédents car le gain est mesuré pour toutes les valeurs des dictionnaires 3 et 4. En effet, bien que seul l'espace mémoire alloué au dictionnaire 4 soit représenté, la formule $S_3 + S_4 = 2^n - 256 - 16$ nous permet de déduire la taille du dictionnaire 3 à partir du dictionnaire 4. Les dictionnaires 1 et 2 sont saturés pour l'ensemble des mesures.

L'objectif des trois figures est de montrer un lien entre le degré qualitatif de redondance des séquences de symboles sources dans les jeux de trames et les gains de compression obtenus.

5.4.2.1 Jeu de trames standard

Ainsi, la Figure 74 est obtenue à partir du jeu standard de trames généré dans la partie précédente. Il s'agit de trames présentant un degré moyen de redondance tel que les coefficients de pondération valent {15 ; 25 ; 20 ; 40}.

La Figure 74 reprend les résultats de la Figure 73 et les étend. Les commentaires sont les mêmes que pour la Figure 73.

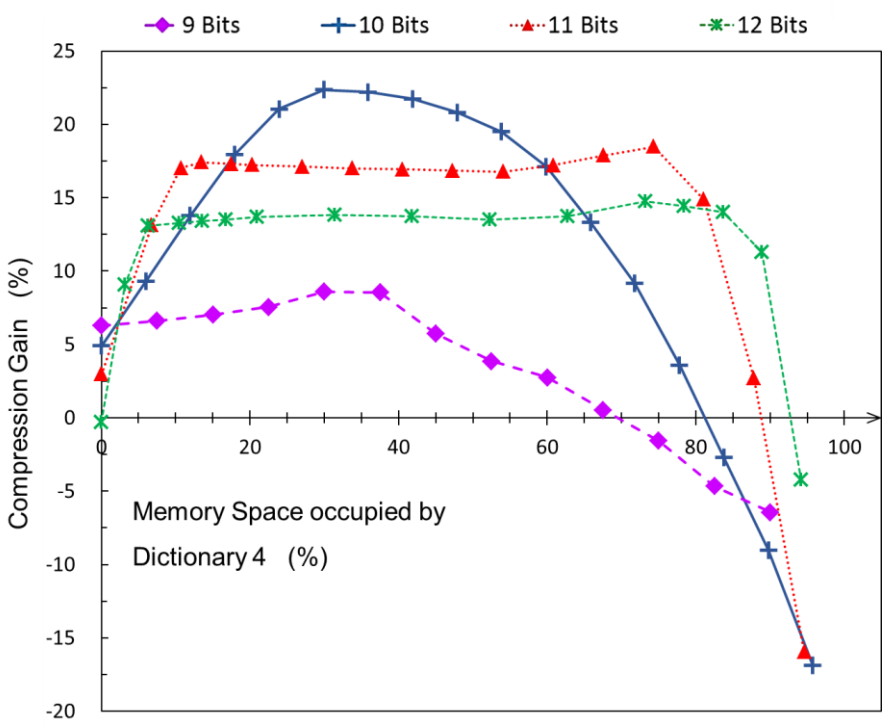


Figure 74 – Gain de compression pour un jeu standard de trames (12 millions de symboles sources).

5.4.2.2 Jeu de trames orienté séquences courtes

La Figure 75 est construite à partir d'un jeu de trames relativement moins riche en séquences de quatre symboles sources et présentant plutôt une redondance forte sur les séquences de deux ou trois symboles. Nous parlons de jeu de trames orienté séquences *courtes*. Les coefficients de pondération choisi valent {15 ; 35 ; 15 ; 35}.

Sur la Figure 75, le gain maximum de 12,0% est obtenu pour 11 bits d'encodage et le gain moyen est meilleur que le gain obtenu pour 10 bits d'encodage. Ces résultats sont dus à la dualité entre deux phénomènes. D'une part, le dictionnaire 4 doit avoir la même taille que pour le jeu standard de trames. D'autre part, le dictionnaire 3 doit être plus grand car la génération de trames est plus orientée vers les séquences courtes. Par conséquent, le résultat de l'addition des dictionnaires 3 et 4 doit être plus grand que pour le jeu standard, et donc 10 bits d'encodage n'est pas la meilleure solution.

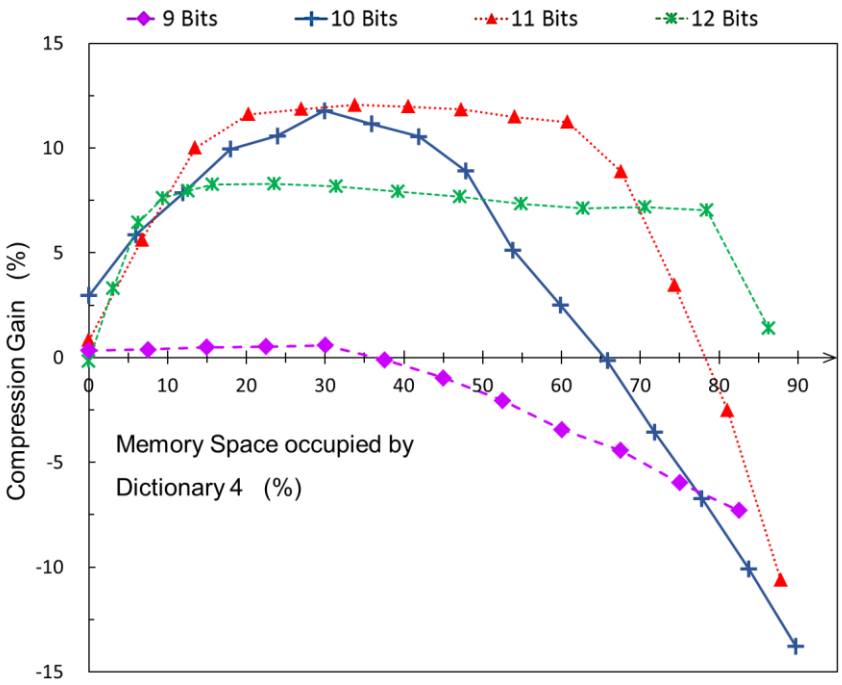


Figure 75 – Gain de compression pour un jeu de trames orienté séquences *courtes*.

5.4.2.3 Jeu de trames orienté séquences longues

Enfin pour la Figure 76, les listes de tirages pour la génération des trames sont agrandies pour les séquences de trois et quatre symboles sources et les coefficients de tirage favorisent le tirage de séquences de quatre symboles : {5 ; 20 ; 15 ; 60}. Ce dernier jeu de trames est donc plutôt orienté séquences *longues*.

Sur la Figure 76, la courbe correspondant à 9 bits d'encodage n'est pas tracée car le gain reste négatif dans cette configuration quelle que soit la taille du dictionnaire 4. Le gain de compression pour 10 bits d'encodage est faible et 12 bits d'encodage offrent un meilleur gain de compression que dans les autres exemples. La meilleure configuration d'encodage est de nouveau pour 11 bits (avec un gain maximum de 13,5%) bien que la zone constante soit plus petite.

Étant donné que les listes de génération sont plus longues, les dictionnaires doivent être plus grands pour garantir que la plupart des séquences se retrouvent bien dans les dictionnaires 3 et 4. Donc, la taille requise des dictionnaires favorise des valeurs élevées pour les bits de codage.

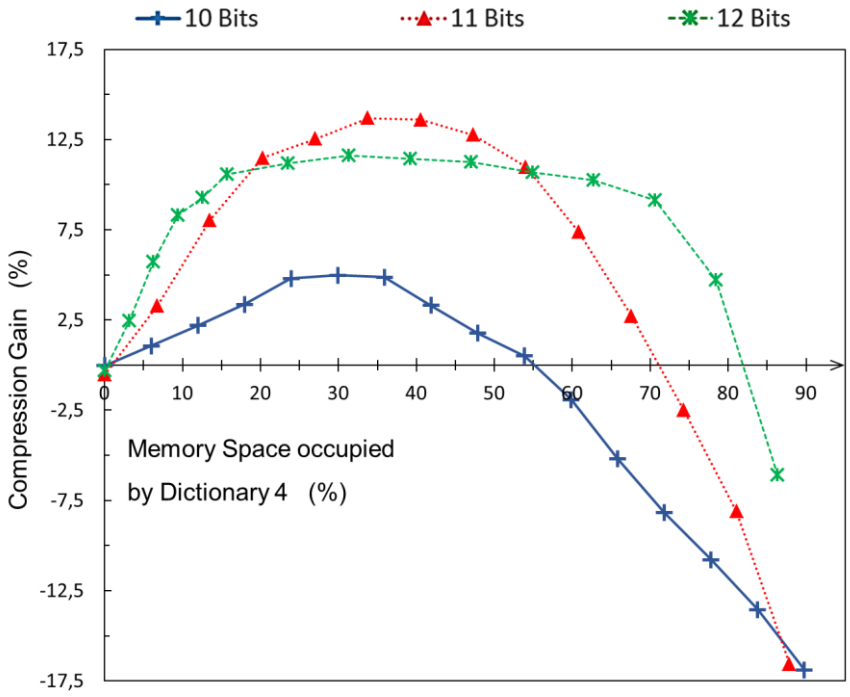


Figure 76 – Gain de compression pour un jeu de trames orienté séquences *longues*.

5.5 Conclusion

Dans ce chapitre, nous sommes allés plus loin que la modélisation du flux de trames entrant pour réduire la taille de la mémoire de réception. En effet, nous avons proposé d'implémenter une méthode de compression sans perte au sein de l'ES de réception en amont de la mémoire et de mesurer les gains obtenus. L'étude des algorithmes de compression nous a amenés à deux constats.

Le premier est que le gain de compression est d'autant plus fort que le degré de redondance sur l'ensemble des trames est fort, et ce indépendamment de la méthode de compression choisie. En l'absence de jeux de trames issus de configurations réseaux industrielles, le premier défi a été de générer des trames AFDX réalistes, dont le degré de redondance globale est ajustable. Une distribution uniforme pour les symboles sources constituant les trames est la solution la plus générale mais elle présente un degré de redondance incertain et non maîtrisé. C'est pourquoi, nous avons proposé une distribution non-uniforme à listes de tirage et à coefficients de pondération. Les coefficients de pondération permettent de choisir qualitativement le degré de redondance des séquences de symboles sources.

Le second constat est que la compression sans perte à dictionnaire LZW se prête bien au contexte des trames AFDX, dont l'absence de statistiques sur la redondance du flux d'entrée contraint fortement le choix de la méthode de compression. Un ensemble encodeur/décodeur LZW a été implémenté en matériel sur un FPGA ainsi que la plateforme nécessaire aux mesures de gain. Le gain de compression a été évalué pour des tailles de dictionnaires variables et trois jeux de trames, chacun

composé de plusieurs millions de symboles sources. Les gains obtenus oscillent entre 12,0% et 22,3% et permettent donc une réduction de la mémoire de réception d'autant.

Enfin, nous pouvons affirmer qu'un gain optimal est obtenu si le flux d'entrée est analysé hors ligne puisque des paramètres généraux sur la taille des dictionnaires ne peuvent pas générer de gains élevés. À ce sujet, de futurs développements pourraient conduire à un algorithme LZW adaptatif dont les tailles de dictionnaires seraient initialement ajustées selon des statistiques calculées sur le flux d'entrée. Après un certain nombre d'itérations, le gain de compression obtenu pourrait être plus élevé qu'avec des dictionnaires statiques.

Conclusion générale

Au terme des cinq chapitres de cette thèse, il est temps d'en présenter la conclusion générale en rappelant les principaux points du contexte de la réalisation de ce travail, les contributions majeures ainsi que les résultats les plus significatifs.

Ce travail s'insère dans le contexte des architectures multicœurs dont l'utilisation se répand dans le domaine aéronautique. Les contraintes de fonctionnement et la nécessité de certifier de telles architectures ont conduit les développeurs d'architecture à adopter de nouvelles méthodes de conception basées sur le partitionnement robuste. Cela consiste à assurer par construction une isolation matérielle et fonctionnelle entre les applications logicielles hébergées sur l'architecture multicœurs. De par leur nombre limité, les IOs ont un accès partagé avec plusieurs cœurs et ils constituent ainsi un point potentiel de rupture de partitionnement. Dans notre étude, nous nous sommes focalisés sur une IO complexe : l'IO AFDX, nommée End-System (ES).

L'étude de l'ES a mis en évidence la problématique critique du dimensionnement de la mémoire de réception de l'ES dans lequel sont stockées les trames reçues. Notre ES étant développé sur FPGA, la mémoire sur puce est limitée. De plus, la mémoire de réception est placée à un goulot d'étranglement de l'ES de réception entre les couches matérielles et logicielles. En bref, un mauvais dimensionnement de cette mémoire peut être à l'origine, ou bien d'un gaspillage de ressources mémoires s'elle est trop grande, ou bien d'une perte/corruption de trames si elle est trop petite. La perte ou la corruption d'une trame peut avoir un impact important sur l'exécution des partitions logicielles et mettre en danger les passagers et le personnel de bord. Nous avons alors travaillé à l'élaboration d'heuristiques et d'hypothèses pour caractériser le flux entrant dans la mémoire et nous avons observé que la réception de SBFs conduisait à un backlog au sein de la mémoire. Nous avons fait l'hypothèse que la réception du LSBF pour une CTRES donnée conduisait au WFB à partir duquel nous pouvions déduire la taille optimale de la mémoire.

La construction du LSBF pour une CTRES donnée est le but des chapitres 2 et 3. Deux approches sont proposées et permettent la construction du LSBF pour de nombreuses CTRESs, ainsi que le calcul de la taille optimale de la mémoire de réception pour chaque CTRES. Chaque approche consiste en une méthode de construction du LSBF basée d'abord sur un modèle pessimiste, puis sur un modèle utilisant les intervalles de réception des trames. Une étude comparative a montré que le modèle par intervalles, dont la granularité est plus fine par rapport au modèle pessimiste, apporte une réduction de la taille du LSBF (en nombre de trames) qui peut s'élever à 65% pour 100 VLs et donc à une réduction d'autant de la taille de la mémoire de réception.

Fort de ces premiers résultats, nous avons orienté nos travaux dans une autre direction, celle d'une analyse probabiliste de l'occurrence de SBFs présentée dans le chapitre 4. L'idée sous-jacente était de réfléchir au bien-fondé de la prise en compte des LSBFs pour le dimensionnement de la mémoire. En effet, si le LSBF se produit extrêmement rarement, nous pourrions considérer un SBF plus petit et accepter une perte très exceptionnelle de trames. Pour cela, nous avons proposé une méthode probabiliste de recherche de SBFs dans le flux de trames entrant, basée sur le modèle par intervalles auquel nous avons retiré les heuristiques de placement pour les remplacer par des tirages basés sur des distributions de probabilités.

La caractérisation des phénomènes temporels liés au transit des trames sur le réseau a permis d'arrêter le choix des distributions de probabilité sur les lois uniformes et de Laplace-Gauss afin de générer un flux de trames entrant réaliste. Après traitement des plusieurs centaines de millions de trames, les résultats ont montré que les LSBFs obtenus avec la méthode de construction du LSBF pessimiste étaient de taille bien supérieure aux LSBFs obtenus avec la méthode probabiliste (de l'ordre de 50% pour plusieurs CTRESs). De plus, les probabilités d'occurrence des LSBFs obtenus avec la méthode probabiliste étaient inférieures à 10^{-6}. Ainsi, nous avons discuté de la prise en compte du LSBF construit pour le dimensionnement de la mémoire. Un seuil pourrait être fixé et, sous condition que la probabilité d'occurrence du LSBF serait inférieure à ce seuil, nous déciderions de ne pas prendre en compte ce LSBF pour le dimensionnement de la mémoire. Une fiabilité à 10^{-9} ou à 10^{-10} est généralement suffisante pour les systèmes critiques.

Les méthodes de construction du LSBF et la méthode probabiliste de recherche de SBFs sont des caractérisations du flux de trames en réception. Par ce biais, nous pouvons dimensionner la mémoire de réception au plus juste des exigences de la CTRES bien que nous n'agissions pas directement sur le flux de trames entrant pour réduire la taille de la mémoire. C'est pourquoi, nous prolongeons l'étude du dimensionnement au travers du chapitre 5 en proposant l'implémentation matérielle d'une méthode de compression de trames au sein de l'ES de réception. L'idée est d'estimer le degré de redondance dans les trames reçues et de proposer un code de compression sans perte adéquat. Ainsi, nous avons implémenté en matériel une adaptation d'un code LZW sous la forme d'un couple encodeur/décodeur placé respectivement en amont et en aval de la mémoire de réception. En l'absence d'exemples de trames AFDX issues de configurations réelles, nous avons également proposé un générateur de trames AFDX permettant d'agir sur le degré de redondance des symboles sources constituant les trames. Pour des millions de symboles sources générés, nous avons relevé des gains de compression variant entre 12,0% et 22,3%, soit une possible réduction de la taille de la mémoire d'autant, tout en garantissant la non-perte d'information contenue dans les trames compressées.

En résumé, nous avons proposé trois approches différentes qui contribuent chacune à la réduction de la taille de la mémoire de réception. Chaque approche met en œuvre la même démarche scientifique depuis l'étude de l'état de l'art à la synthèse des résultats obtenus en passant par la mise au point d'un modèle théorique et des simulateurs codés en C ou en VHDL. Enfin, les méthodes présentées se plient une certaine progression logique. Mis bout en bout, les gains globaux de réduction de la taille de la mémoire varient entre 55 et 70%, avec la considération d'un cas moins-pire pour le LSBF calculé avec la méthode probabiliste et par rapport aux tailles de la mémoire mesurées avec la méthode de construction du LSBF pessimiste.

Perspectives

Dans cette ultime partie, nous présentons trois pistes pour la poursuite de ces travaux : des essais du un banc AFDX pour l'ajustement des méthodes et des modèles proposés, une technique de « lissage » du flux de trames entrant dans la mémoire par la mise en place d'une petite IP matérielle en amont de la mémoire, et une extension de la méthode de compression aux transmissions AFDX.

Essais sur un banc AFDX

Les méthodes de construction de LSBF et la méthode probabiliste de recherche de SBFs ont une pertinence théorique certaine et leur implémentation en langage C dans des simulateurs a permis l'obtention de résultats théoriques permettant d'estimer la taille de la mémoire de réception.

Cela dit, ces méthodes mériteraient d'être confrontées à des mesures effectuées sur un banc AFDX. Ces mesures n'ont pas pu être faites dans le cadre de cette thèse, faute d'avoir un tel banc à disposition. D'une part, ces mesures permettraient de mesurer le LSBF pour une CTRES donnée après une certaine durée de fonctionnement du réseau, et donc d'évaluer la qualité des méthodes de construction. En outre, la possibilité de tester la pile AFDX complète en réception avec un banc permettrait d'affiner le modèle en lecture du flux de trames sortant de la mémoire.

D'autre part, les occurrences de SBFs pourraient être comptées et comparées à celles obtenues avec la méthode probabiliste. L'analyse du flux de trames entrant et la mesure du backlog dans la mémoire sur des essais longs pourraient nous fournir des statistiques précieuses pour la définition d'un seuil de fiabilité. Nous pourrions ainsi poursuivre la discussion au sujet de la pertinence de la prise en compte du pire-cas pour le dimensionnement de la mémoire et proposer des tailles mémoire plus réduites.

Enfin, un banc AFDX pourrait nous fournir une banque de trames AFDX dont l'analyse pourrait révéler de façon précise les redondances dans les symboles sources des trames. Cela aurait une utilité pour calculer des gains de compression de trames plus finement et certainement que les gains obtenus seraient supérieurs à ceux que nous avons établis dans le chapitre 5. En effet, nous sommes restés prudents avec notre générateur de trames et nous avons limité la redondance des symboles sources pour pressentir l'intérêt d'un algorithme de compression sans perte LZW.

Lissage du flux de trames entrant

Dans cette partie, nous présentons la technique du lissage du flux de trames entrant dans la mémoire. Il s'agit d'une technique simple inspirée de la méthode des pipelines pour l'exécution sur CPU, dont l'objectif est de limiter les effets des SBFs sur la mémoire de réception. Le lissage du flux de trames entrant consiste simplement à placer une petite mémoire tampon en amont de la mémoire de réception dont le rôle est de réguler le flux de trames entrant. Le lissage agit en complément de la compression LZW. Alors que la compression LZW réduit la taille individuelle des trames, le lissage agit sur la réception des SBFs en « cassant » les accolements par l'ajout de courts délais entre les trames accolées.

Le but de cette technique est de délester la mémoire de réception et de diminuer son WFB, tout en garantissant que l'addition des ressources mémoires de la mémoire tampon et de la mémoire de réception donne un résultat inférieur à la quantité de mémoire requise pour la mémoire de réception seule. Pour cela, nous jouons sur le fait que la taille de la mémoire de réception est multiple de 2. Ainsi, sous des conditions que nous allons préciser dans la suite, il est possible de réduire l'utilisation des ressources mémoires.

La Figure 77 représente l'IP matérielle de cassage des SBFs (IP SBF Breaker) qui comprend un contrôleur de délai en amont de la mémoire de réception. L'IP SBF Breaker se compose de deux blocs : un contrôleur de délai et une mémoire de type FIFO (que nous nommerons simplement *FIFO* pour ne pas la confondre avec la mémoire de réception).

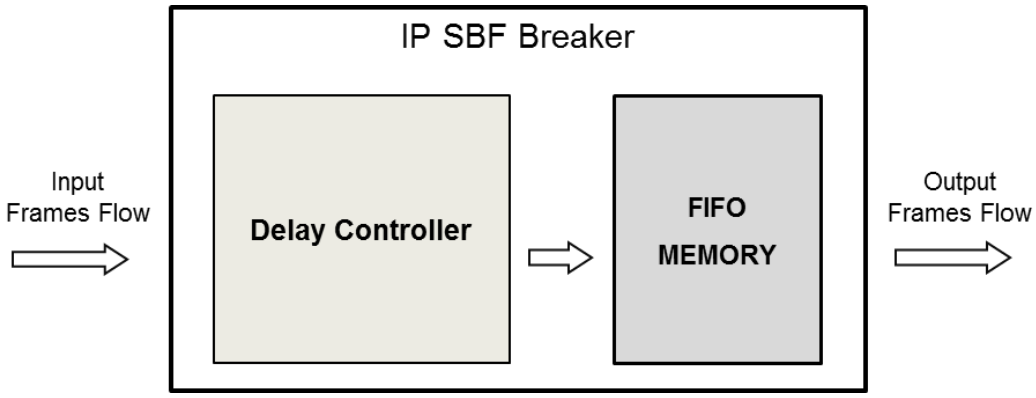


Figure 77 – IP matérielle de réduction de SBFs (IP SBF Breaker).

Le contrôleur de délai détecte les trames accolées par une mesure systématique du délai entre la réception de deux trames consécutives sur le lien physique. Si deux trames consécutives sont détectées accolées, la seconde trame est stockée un court instant dans la FIFO, avant d'être réexpédiée. Le délai d'attente avant la réexpédition de la seconde trame peut correspondre à différentes politiques comme illustrées sur la Figure 78.

Pour la première politique d'insertion de délais, le délai d'attente est choisi constant et multiple de ε (le délai minimum entre deux trames consécutives), et il est placé entre deux trames consécutives

appartenant à un SBF de façon systématique (a). Une seconde politique plus adaptative consiste à insérer des délais plus longs après les trames dont la taille est supérieure à 300 octets et à ne pas placer de délai supplémentaire après les trames dont la taille est inférieure à 300 octets (b).

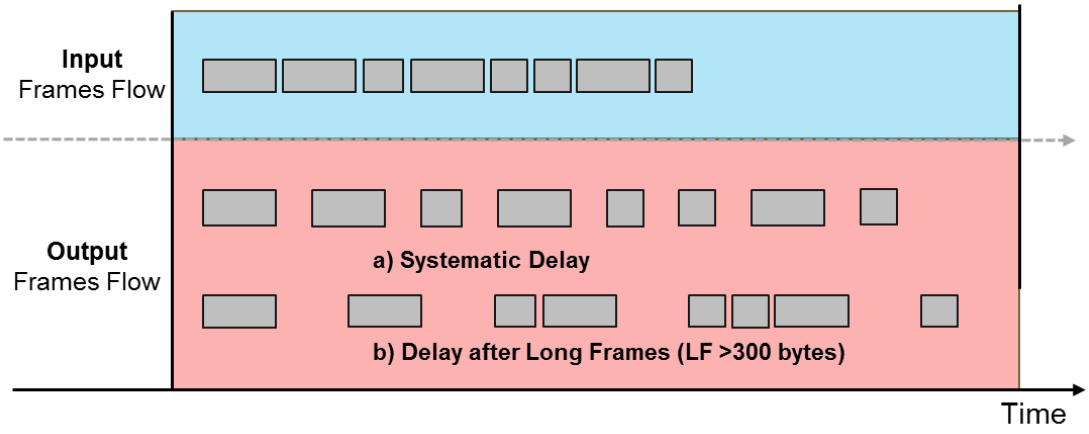


Figure 78 – Exemples de politique d'insertion de délais mises dans œuvre dans le contrôleur de délais.

Ensuite, nous proposons une implémentation du contrôleur de délai en matériel l'IP SBF Breaker, et nous réalisons quelques tests préliminaires avec la politique d'insertion de délais systématiques (Figure 78 (a)). Nous reprenons le LSBF mesuré au chapitre 2 pour la CTRES industrielle et nous mesurons les backlogs dans la mémoire de réception (Figure 79) et dans la FIFO de l'IP SBF Breaker (Figure 80) pour différentes valeurs de délai multiple de ε.

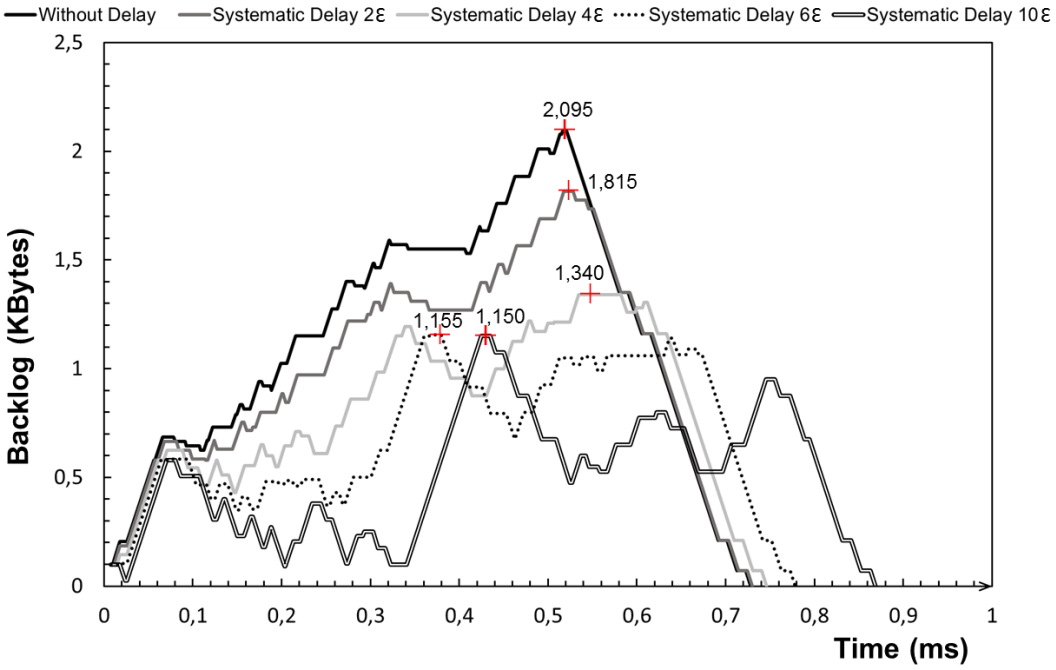


Figure 79 – Mesure du backlog dans la mémoire de réception pour différentes valeurs de délai avec la politique d'insertion de délai systématique.

Nous constatons que la mise en place de l'IP SBFs Breaker réduit la taille du WFB de la mémoire de réception quelle que soit la valeur du délai inséré. Une augmentation de la valeur du délai implique une diminution du WFB dans la mémoire et une augmentation concomitante du WFB de la mémoire FIFO. Ceci est assez logique puisque si les trames sont davantage retardées au sein de l'IP SBF Breaker, les trames vont s'accumuler dans la mémoire FIFO. Si le délai choisi est trop grand, le WFB de la mémoire FIFO dépasse celui de la mémoire de réception comme nous le constatons pour un délai de 10ε.

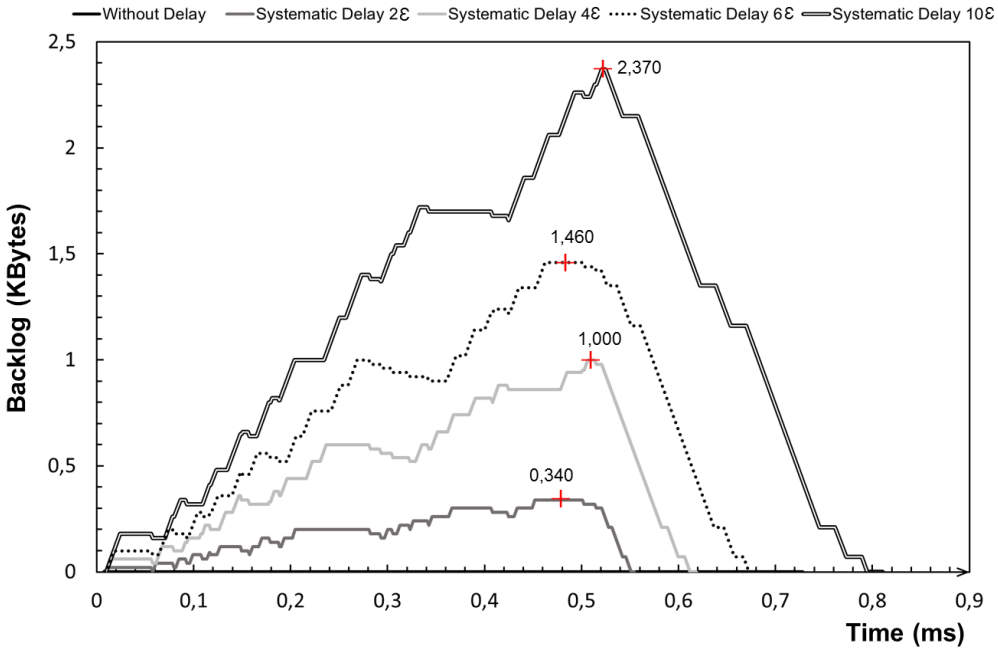


Figure 80 – Mesure du backlog dans la mémoire FIFO de l'IP SBF Breaker pour différentes valeurs de délai avec la politique d'insertion de délai systématique.

Par conséquent, nous pourrions avoir effectivement un gain mémoire à condition que la somme des tailles de la mémoire de réception et de la mémoire FIFO n'excède pas la taille initiale de la mémoire de réception seule. Une réduction de la mémoire globale peut être observée pour une insertion de délai systématique petit. Pour confirmer cela, nous avons mesuré plus finement l'évolution des WFBs sous un accroissement du délai systématique et nous avons rassemblé les résultats sur la Figure 81.

Les résultats obtenus montrent que le WFB diminue dans la mémoire de réception lorsque le délai systématique augmente entre 1ε et 11ε. À partir de 5ε, le WFB semble se stabiliser à 1 150 octets, ce qui correspond à la taille de la plus grande trame de la CTRES industrielle. Ainsi, nous pouvons extrapoler que si la taille de la plus grande trame de la CTRES était de valeur inférieure, nous observerions une stabilisation du WFB à une valeur inférieure. Par ailleurs, le WFB de la FIFO augmente logiquement sous l'accroissement du délai systématique en partant de 0 (en l'absence d'insertion de délais) et jusqu'à une valeur de 2 500 octets pour un délai de onze fois le délai minimum ε. Le WFB ainsi mesuré dépasse donc le WFB initial de la mémoire de réception.

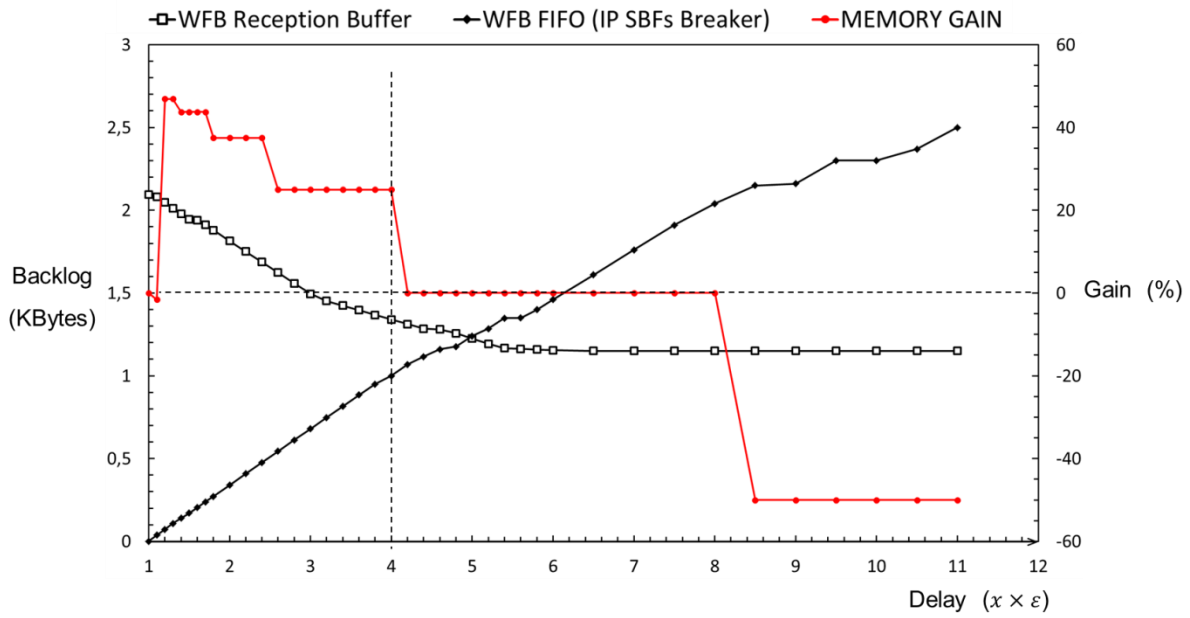


Figure 81 – WFBs de la mémoire de réception et de la FIFO, et gain mémoire global en fonction du délai systématique inséré.

Enfin, le gain mémoire (en rouge) a une forme en paliers puisque les tailles des mémoires sont forcément exprimées en multiple de 2. Conclusion : un gain positif est obtenu pour un délai compris entre $1{,}2\varepsilon$ et 4ε et il s'élève jusqu'à 46,88% dans nos conditions de test. Cependant, à partir d'un délai systématique de $4{,}2\varepsilon$, le gain devient nul voire négatif à partir de $8{,}5\varepsilon$ puisque le WFB de la FIFO atteint des valeurs de plus en plus importantes.

Ainsi, les résultats préliminaires de la technique de lissage du flux de trames entrant sont prometteurs et mériteraient d'être approfondis davantage. D'autres mesures réalisées sur d'autres CTRESs pourraient renforcer les conclusions de cette étude mais également l'exploration de d'autres politiques d'insertion de délai.

Compression de trames de bout en bout sur le réseau AFDX

L'implémentation du code de compression sans perte LZW a donné des résultats prometteurs sur la réduction de la taille des trames stockées dans la mémoire de réception. Suivant le degré de redondance des symboles sources constituant les trames, nous avons pu obtenir un gain maximal variant entre 12,0% et 22,3%.

Une perspective intéressante serait d'appliquer la compression de façon plus étendue, lors de la transmission des trames sur le réseau AFDX. Un encodeur LZW serait placé dans chaque ES source et compresserait une partie des trames lors de leur transmission sur le réseau. De l'autre côté de la chaîne de transmission, un décodeur LZW placé dans l'ES de réception décompresserait les trames pour un coût en ressources et en temps de calcul réduit.

En effet, des mécanismes de compression sont déjà appliqués dans le cadre de la transmission sur réseau Ethernet, ceci dû à l'évolution des besoins grand public. Le transfert de vidéos de très haute résolution 4K (82) ou bien pour le transfert de fichiers volumineux pour une sauvegarde durable des données (83), (84) requiert l'utilisation de codes de compression sans quoi la bande passante requise serait très importante. Si dans un proche avenir, le réseau AFDX est utilisé pour la transmission de contenu multimédia, un code de compression serait très utile. Nous pourrions alors envisager l'emploi de codes de compression avec perte qui offrent des gains de compression plus élevés que les codes de compression sans perte.

Dans le cas du réseau AFDX, les trames seraient compressées partiellement lors de leur envoi sur le réseau, et décompressées lors de leur réception au niveau de la couche matérielle de l'ES de réception. Une compression partielle serait portée uniquement sur le payload des trames en laissant les en-têtes lisibles pour gagner du temps lors de la traversée d'un commutateur. Une autre idée serait de compresser la trame IP (donc headers IP, UDP et payload) puisque le commutateur ne fait une lecture que de la couche MAC pour le transfert de la trame son VL attitré.

Le premier avantage d'une telle méthode de compression serait un gain en bande passante sur le réseau. Une trame serait donc plus rapidement transmise sur les liens physiques, occuperait moins de place dans les ports de sortie des commutateurs et donc cela réduirait la probabilité d'occurrence d'une congestion dans lesdits ports de sortie. Par conséquent, les délais de bout en bout maximum seraient réduits et le caractère déterministe du réseau serait renforcé.

Un autre avantage serait que le réseau pourrait faire face à une évolution des besoins en terme de bande passante sans que cela nécessite une évolution technologique majeure vers un autre système de communication embarquée (comme le TTEthernet® (55), (85)) ou un besoin d'augmentation de la vitesse de transmission du réseau, ce qui compliquerait dans le même temps la démonstration du déterminisme et de la fiabilité des données transmises.

Publications

Publications Internationales avec comités de lecture :

Y. Baga, F. Ghaffari, E. Zante, M. Nahmiyace and D. Declercq, “Worst Frame Backlog Estimation in an Avionics Full-Duplex switched Ethernet End-System,” in 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), pp. 1–10, Sept 2016.

Y. Baga, F. Ghaffari, D. Declercq, E. Zante and M. Nahmiyace, “Probabilistic Model of AFDX Frames Reception for End-System Backlog Assessment,” in 2017 IEEE 12th International Symposium on Industrial Embedded Systems (SIES), June 2017.

Y. Baga, F. Ghaffari, D. Declercq, E. Zante and M. Nahmiyace, “Reduction of Frames Storage Size in AFDX Reception End-System using a Lossless Compression Algorithm,” in 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), Sept 2017.

En cours de soumission à un journal :

Y. Baga, F. Ghaffari, D. Declercq, E. Zante and M. Nahmiyace, “Dimensioning of an End-System Reception Buffer based on a Realistic Model of AFDX Frames Flow” in IEEE ACCESS 2017.

Bibliographie

1. **COMMITTEE, AIRLINES ELECTRONIC ENGINEERING.** *ARINC SPECIFICATION 664P7.* Annapolis, Maryland : AERONAUTICAL RADIO, INC., 2005.

2. *ARINC429: Mark 33 Digital Information Transfer System (DITS), Part 1: Functional Description, Electrical Interfaces, Labels Assignments and Words Formats.* **A. E. E, Commitee.** s.l. : Aeronautical Radio Inc., 1995.

3. **A. E. E, Commitee.** *ARINC specification 664p7: Aircraft Data Network, part 7: Avionics Full-Duplex Switched Ethernet (AFDX) network.* s.l. : Aeronautical Radio Inc., June 2005.

4. **Wilson, Alex.** Sûreté de fonctionnement et systèmes multicoeurs pour l'avionique doivent s'apprivoiser. *L'EMBARQUÉ.* Novembre 2013, pp. 28 - 30.

5. **Jehl, Régis.** lesnumeriques. *site Web lesnumeriques.* [Online] Février 26, 2016. [Cited: Septembre 06, 2017.] http://www.lesnumeriques.com/cpu-processeur/point-sur-processeurs-actuels-a-venir-a2629.html.

6. **Cobham Gaisler, AB.** *LEON4 Product Sheet.* Plainview, New-York : s.n., 2010.

7. **Xilinx Inc.** *MicroBlaze Processor Reference Guide.* San Jose, California : s.n., 2010.

8. **Altera Corp.** *Nios II Gen2 Processor Reference Guide.* San Jose, California : s.n., April 2015.

9. *Symmetric multiprocessing on programmable chips made easy.* **Hung, A. and Bishop, W. and Kennings, A.** March 2005, Design, Automation and Test in Europe, 2005. Proceedings, pp. 240-245 Vol. 1.

10. *An Efficient and Low-Cost Design Methodology to Improve SRAM-Based FPGA Robustness in Space and Avionics Applications.* **Lanuzza, M., et al., et al.** 2009, Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications, pp. 74-84.

11. *Managing cache partitioning in multicore processors for certifiable, safety-critical avionics software applications.* **King, T.** October 2014, Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd, pp. 8C3-1-8C3-7.

12. *Deterministic Execution Model on COTS Hardware.* **Boniol, F., et al., et al.** February 2012, International Conference on Architecture of Computing Systems (ARCS).

13. *Integrated Modular Avionics (IMA) Partition Scheduling with Conflict-Free I/O for Multicore Avionics Systems.* **Kim, J., et al., et al.** July 2014, Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual, pp. 321-331.

14. *Mastering the behavior of multi-core systems to match avionics requirements.* **Agrou, H., et al., et al.** October 2012, Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st, pp. 6E5-1-6E5-12.

15. *Transitioning from federated avionics architectures to Integrated Modular Avionics.* **C. B, Watkins and Walter, R.** October 2007, Digital Avionics Systems Conference, 2007. DASC '07. IEEE/AIAA 26th, pp. 2.A.1-1-2.A.1-10.

16. *Integrated modular avionics.* **Prisaznuk, P. J.** May 1992, Electronics and Aerospace Conference NAECON 1992., Proceedings of the IEEE 1992, pp. 1:39–45.

17. *Use of multicore processors in avionics systems and its potential impact on implementation and certification.* **Kinnan, L.M.** October 2009, Digital Avionics Systems Conference, 2009. DASC '09., pp. 1.E.4–1-1.E.4–6.

18. *Incremental functional certification for avionic functions reuse amp; evolution.* **Gatti, S., et al., et al.** October 2012, Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st, pp. 7A5–1-7A5–16.

19. **RTCA.** *DO-297: Integrated Modular Avionics (IMA Development, Guidance and Certification Considerations.* s.l. : RTCA, 2005.

20. **A. E. E, Commitee.** *ARINC653: Avionics Application Software Standard Interface, part 1 - Required Services.* s.l. : Aeronautical Radio Inc., 2005.

21. *Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance.* **Rushby, J.** March 2000, NASA Langley Technical Report, DOT/FAA/AR-99/58.

22. *Invariant performance: a statement of task isolation useful for embedded application integration.* **Wilding, M. M., Hardin, D. S. and Greve, D. A.** January 1999, Dependable Computing for Critical Applications 7, pp. 7A5-1-7A5-16.

23. *Commercial Off-The-Shelf Real-Time Operating System and Architectural Considerations.* **Krodel, J.** February 2004, Federal Aviation Administration final report DOT/FAA/AR-03/77.

24. *Predictability Considerations in the Design of Multi-Core Embedded Systems.* **Cullmann, C., et al., et al.** 2010, Proceedings of Embedded Real Time Software and Systems ERTS2.

25. *Challenges in Future Avionic Systems on Multi-Core Platforms.* **Lofwenmark, A. and Nadjm-Tehrani, S.** November2014, Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on, pp. 115-119.

26. *ARINC629: Avionic Data Bus Standard.* **A. E. E, Commitee.** s.l. : Aeronautical Radio Inc., 1995.

27. *Application of Network Calculus to Guaranteed Service Networks.* **Le Boudec, J. Y.** 1998, IEEE Transactions on Information Theory, Vol. 113, pp. 1087-1096.

28. *Worst-case end-to-end delay analysis of an avionics AFDX network.* **Bauer, H., Scharbarg, J. L. and Fraboul, C.** 2010, Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1220-1224.

29. *Improving AFDX end-to-end delays analysis.* **Kemayo, G., et al., et al.** September 2015, 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1-8.

30. *A calculus for network delay.* **Cruz, R. L.** January 1991, IEEE Transactions on Information Theory, Vol. 37, pp. 114-131.

31. *Optimistic problems in the trajectory approach in FIFO context.* **Kemayo, G., et al., et al.** 2013, 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1-8.

32. *The Trajectory approach for AFDX FIFO networks revisited and corrected.* **Li, X., Cros, O and George, L.** 2014, 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 1-10.

33. **Grieu, J.** *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques.* s.l. : Institut National Polytechnique de Toulouse, 2004.

34. *Probabilistic upper bounds for heterogeneous flows using a static priority queueing on an AFDX network.* **Ridouard, F., Scharbarg, J. L. and Fraboul, C.** September 2008, IEEE International Conference on Emerging Technologies and Factory Automation.

35. *An improved timed automata approach for computing exact worst-case delays of AFDX sporadic flows.* **Adnan, M., et al., et al.** 2012, Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies and Factory Automation (ETFA).

36. *Modelling and Simulation of an Avionics Full Duplex Switched Ethernet.* **Charara, H. and Fraboul, C.** July 2005, Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings, pp. 207-212.

37. *Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network.* **Bauer, H., Scharbarg, J. L. and Fraboul, C.** 2009, 2009 IEEE Conference on Emerging Technologies & Factory Automation, pp. 1-8.

38. *Worst-Case Backlog Evaluation of Avionics Switched Ethernet Networks with the Trajectory Approach.* **Bauer, H., Scharbarg, J. L. and Fraboul, C.** July 2012, 2012 24th Euromicro Conference on Real-Time Systems.

39. *Optimism due to serialization in the trajectory approach for switched Ethernet networks.* **Kemayo, G., et al., et al.** 2013, Proc. of Int. Conf. on Junior Researcher Workshop on Real-Time Computing (JRWRTC), pp. 13-16.

40. *Impact of End System scheduling policies on AFDX performance in avionic on-board data network.* **Suthaputchakun, C., Lee, K. and Sun, Z.** August 2015, Advanced Informatics: Concepts, Theory and Applications (ICAICTA), 2015 2nd International Conference on, pp. 1-6.

41. *Optimal scheduling policy for jitter control in AFDX End-System.* **Gurjar, S. and Lakshmi, B.** September 2014, Recent Advances and Innovations in Engineering (ICRAIE), pp. 1-4.

42. *Priority assignment on an avionics switched Ethernet Network (QoS AFDX).* **Hamza, T., Scharbarg, J. L. and Fraboul, C.** May 2014, 10th IEEE Workshop on Factory Communication Systems (WFCS), pp. 1-8.

43. *The Research of AFDX System Simulation Model.* **Ding, L., et al., et al.** October 2010, 2010 International Conference on Multimedia Technology, pp. 1-4.

44. *End to end jitter control on AFDX network.* **Ren, Y., Hu, F. and Li, J.** December 2011, Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), pp. 515-518.

45. *Using traffic phase shifting to improve AFDX link utilization.* **Mancuso, R., Louis, A. V. and Caccamo, M.** October 2015, 2015 International Conference on Embedded Software (EMSOFT), pp. 256-265.

46. *Network topology optimization for distributed integrated modular avionics.* **Annighoefer, B., Reif, C. and Thieleck, F.** October 2014, 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), pp. 4A1-1-4A1-12.

47. **(IEEE), Institute of Electrical and Electronics Engineers.** *IEEE 802.3 standard.* s.l. : Institute of Electrical and Electronics Engineers (IEEE), 1983.

48. **Group, Network Working.** *RFC791: Internet protocol darpa internet program protocol specification.* 1981.

49. —. *RFC 1071: Computing the Internet Checksum.* 1988.

50. *The buffer size assignment of AFDX based on network calculus.* **Zhitao, W., et al., et al.** 2011, Reliability, Maintainability and Safety (ICRMS).

51. *Calculation of Worst Case Backlog for AFDX Buffers with Two Priority Levels using Trajectory Approach.* **Garikiparthi, N. R., Fohler, G. and Coelho, R.** July 2013, 12th Workshop on Real-time Networks (RTN'13) in conjuction with 25th Euromicro International Conference on Real-time Systems (ECRTS'13).

52. *Dimensioning buffers for AFDX networks with multiple priorities virtual links.* **Coelho, R., Fohler, G. and Scharbarg, J. L.** 2015, Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th.

53. *Experimental assessment of timing verification techniques for AFDX.* **Boyer, M., Navet, N. and Fumey, M.** Toulouse : s.n., February 2012, Proceedings of the 6th European Congress on Embedded Real Time Software and Systems.

54. *Integrating end-system frame scheduling for more accurate AFDX timing analysis.* **Boyer, M., et al., et al.** 2014, Proceedings on Embedded Real-Time Software and Systems.

55. *TTTech TTE End-System A664 T datasheet.* **TTTech.** s.l. : TTTech, 2015.

56. *"PEGASE – a robust and efficient tool for worst-case network traversal time evaluation on AFDX.* **Boyer, M., Migge, J. and Fumet, M.** Toulouse : s.n., 2011, SAE Aerotech 2011.

57. **Poisson, Simeon Denis.** *Recherche sur la probabilité de jugements en matière criminelle et en matière civile.* Paris : Bachelier, Imprimeur-Libraire, 1837.

58. *Performance analysis of Poisson and Exponential distribution queuing model in Local Area Network.* **Sadeghi, M. and Barati, M.** July 2012, 2012 International Conference on Computer and Communication Engineering (ICCCE), pp. 499-503.

59. *Traffic behavior of Local Area Network based on M/M/1 queuing model using poisson and exponential distribution.* **Atefi, K., et al., et al.** May 2016, 2016 IEEE Region 10 Symposium (TENSYMP), pp. 19-23.

60. *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC).* **Zhang, Z., et al., et al.** July 2017, Research and Analysis about the Length of Vertex-Degree Sequence of Complex Networks with Poisson Distribution, Vol. 1, pp. 28-31.

61. **De Laplace, S.** *Essai philosophique sur les Probabilités.* Paris : Bachelier, Imprimeur-Libraire, 1840.

62. *A Note on the Generation of Random Normal Deviates.* **Box, G. E. P. and Muller, M. E.** s.l. : Ann. Math. Statist. 29, 1958.

63. *LZ4m: A fast compression algorithm for in-memory data.* **Kwon, Se-Jun, et al., et al.** January 2017, 2017 IEEE International Conference on Consumer Electronics (ICCE), pp. 420-423.

64. *Run-Length encodings.* **Golomb, Solomon Wolf.** 1966, IEEE Transactions on Information Theory, Vol. 12, pp. 399-401.

65. *Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data.* **Robert F. Rice, James R. Plaunt.** 6, 1971, IEEE Transactions on Communications, Vol. 19, pp. 889-897.

66. *Selecting the Golomb Parameter in Rice Coding.* **Kiely, A.** 42-159, 2004, The interplanetary Network Progress.

67. *A method for the construction of minimum-redundancy codes.* **Huffman, D.A.** 1952, Proceedings of the Institute of Radio Engineers, pp. 1098-1102.

68. *RFC 1951.* **Deutsch, L. Peter.** 1996.

69. *An Introduction to Arithmetic Coding.* **Langdon, G.** 2, 1984, IBM Journal of Research and Development , Vol. 28, pp. 135 - 149.

70. *Compression of Black-White Images with Arithmetic Coding.* **Langdon, G.** 6, 1981, IEEE Transactions on Communications , Vol. 29, pp. 858 - 867.

71. *Universal Codeword Sets and Representations of the Integers.* **Elias, Peter.** 2, 1975, IEEE Transactions on Information Theory, Vols. IT-21, pp. 194 - 203.

72. *Reduced-Time Fascimile Transmission by Digital Coding.* **H. Wyle, T. Erb, R. banow.** 3, 1961, IRE Transactions on Communications Systems, Vol. 9, pp. 215 - 222.

73. *Results of a prototype Television Bandwidth Compression Scheme.* **A.H. Robinson, C. Cherry.** 3, 1967, Proceedings of the IEEE, Vol. 55, pp. 356 - 364.

74. *A Block-sorting Lossless Data Compression Algorithm.* **M. Burrows, D. J. Wheeler.** 1994, Technical Report 124, Digital Equipment Corporation.

75. *Ultrafast and memory-efficient alignment of short DNA sequences.* **Ben Langmead, Cole Trapnell, Mihai Pop and Steven L Salzberg.** 3, May 2009, Genome Biology, Vol. 10.

76. *IEEE Transactions on Information Theory.* **Ziv, J. and Lempel, A.** 1977, A universal algorithm for sequential data compression, pp. 337-343.

77. *A Technique for High-Performance Data Compression.* **Welch, T. A.** 1984, Computer, pp. 8-19.

78. *A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture.* **Lin, M. b., Lee, J. f. and Jan, G. E.** September 2006, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 925-936.

79. *BD-LZW picture compression algorithm for WSN system.* **Zhan, J., et al., et al.** October 2008, 2008 Third International Conference on Pervasive Computing and Applications, pp. 146-150.

80. *A Parallel VLSI Architecture For The LZW Data Compression Algorithm.* **Lin, Ming-Bo.** June 1997, Proceedings of Technical Papers. International Symposium on VLSI Technology, Systems, and Applications, pp. 98-101.

81. *Design and implementation of lossless compression encoding for high-speed data acquisition and storage.* **Ce, Z. and Hui, X.** July 2015, 2015 12th IEEE International Conference on Electronic Measurement \& Instruments (ICEMI), pp. 502-506.

82. *4K Video over SMPTE 2022-5/6 Workflows.* **Levy, M., Richardson, L. and Rouvroy, G.** October 2015, SMPTE 2015 Annual Technical Conference and Exhibition, pp. 1-12.

83. *2016 Conference on Design of Circuits and Integrated Systems (DCIS).* **Osorio, R. R.** November 2016, Transaction level and RTL modeling of an architecture for network data compression within ethernet switches in large file transfer scenarios, pp. 1-6.

84. *Performance evaluation of mobile front-haul employing ethernet- based TDM-PON with IQ data compression [Invited].* **Shibata, N., et al., et al.** 11, November 2015, IEEE/OSA Journal of Optical Communications and Networking, Vol. 7, pp. B16-B22.

85. *TTEthernet Dataflow Concept.* **Steiner, W., et al., et al.** 2009, 2009 Eighth IEEE International Symposium on Network Computing and Applications.

86. **(SAE), Society of Automotive Engineers.** *ARP 4754: Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* 1996.

87. *ARINC 653 and multi-core microprocessors, Considerations and potential impacts.* **Huyck, P.** October 2012, Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st, pp. 6B4-1-6B4-7.

88. *Ensuring robust partitioning in multicore platforms for IMA systems.* **Jean, X., et al., et al.** October 2012, Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st, pp. 7A4-1-7A4-9.

89. *How to address certification for multi-core based IMA platforms: Current status and potential solutions.* **Fuchsen, R.** October 2010, Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th, pp. 5.E.3-1-5.E.3-11.

90. **RTCA.** *DO-254: DESIGN ASSURANCE GUIDANCE FOR AIRBORNE ELECTRONIC HARDWARE.* s.l. : RTCA, 2000.

91. *A software approach for managing shared resources in multicore IMA systems.* **Jean, X., et al., et al.** October 2013, Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd, pp. 7D1-1-7D1-15.

92. *Performance Impact of the Interactions between time-triggered and rate-constrained Transmissions in TTEthernet.* **Boyer, M., et al., et al.** 2016, Proceedings of the 8th European Congress Embedded Real Time Software and Systems (ERTSS).

93. *The time-triggered architecture.* **Kopetz, H. and Bauer, G.** 2003, Proceedings of the IEEE 124, pp. 112-126.

94. **A. E. E, Commitee.** *ARINC600 : Avionics Packaging Standard for the Integrated ModularAvionics (IMA).* s.l. : Aeronautical Radio Inc., 2003.

95. *A Probabilistic Analysis of End-To-End Delays on an AFDX Avionic Networ.* **Scharbarg, J. L., Ridouard, F. and Fraboul, C.** 2009, IEEE Transactions on Industrial Informatics, pp. 38-49.

96. *A simple and efficient class of functions to model arrival curve of packetised flows.* **Boyer, M., Migge, J. and Navet, N.** Vienna : s.n., February 2011, First International Workshop on Worst-case Traversal Time (WCTT).

ABSTRACT

Analyse de Flux de Trames AFDX en Réception et Méthodes d'Optimisation Mémoire

L'essor des réseaux AFDX comme infrastructure de communication entre les équipements de bord des aéronefs civils motive de nombreux travaux de recherche pour réduire les délais de communication tout en garantissant un haut niveau de déterminisme et de qualité de service. Cette thèse traite de l'effet des accolements de trames sur l'End-System de réception, notamment sur la mémoire interne afin de garantir une non perte de trames et un dimensionnement mémoire optimal. Deux approches proposent une estimation du pire scénario de réception pour le flux de trames entrant dans la mémoire, chacune basée sur une modélisation de la réception des trames. Une méthode probabiliste met en œuvre des distributions gaussiennes pour évaluer les probabilités d'occurrences de ces pires scénarios et apporte un éclairage qui ouvre une discussion sur la pertinence de ne considérer que le pire scénario pour dimensionner la mémoire de réception. Un gain mémoire supplémentaire peut être obtenu par l'implémentation d'un code de compression sans perte LZW.

– Mots-clés : Réseaux AFDX, flux de trames AFDX en réception, dimensionnement mémoire, estimation du pire scénario en réception, méthode probabiliste, codes de compression de données sans perte.

AFDX Frame Flow Analysis in Reception and Memory Optimization Methods

The rise of AFDX networks as a communication infrastructure between civil aircraft onboard equipment is driving many research projects to reduce communication delays while ensuring a high level of determinism and quality of service. This thesis deals with the effect of frame attachments on the reception End-System, in particular on the internal memory, in order to guarantee a non-loss of frames and optimal memory sizing. Two approaches propose an assessment of the worst reception scenario for the flow of frames entering the memory, each based on a modeling of the reception of the frames. A probabilistic method implements Gaussian distributions to evaluate the probabilities of occurrences of these worst-case scenarios and provides a light that opens a discussion on the relevance of considering only the worst-case scenario for sizing the reception memory. An additional memory gain can be obtained by implementing a LZW lossless compression code.

– Keywords : AFDX networks, AFDX frame flow in reception, memory sizing, worst case analysis of frames, probabilistic modeling of the back-to-back frames, lossless compression codes.