Titre: Title:	Intégration itérative des systèmes avioniques communicants en mode synchrone et asynchrone
Auteur: Author:	Sofiene Beji
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Beji, S. (2018). Intégration itérative des systèmes avioniques communicants en mode synchrone et asynchrone [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3728/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/3728/
Directeurs de recherche: Advisors:	John Mullins, & Abdelouahed Gherbi
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

INTÉGRATION ITÉRATIVE DES SYSTÈMES AVIONIQUES COMMUNICANTS EN MODE SYNCHRONE ET ASYNCHRONE

SOFIENE BEJI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

INTÉGRATION ITÉRATIVE DES SYSTÈMES AVIONIQUES COMMUNICANTS EN MODE SYNCHRONE ET ASYNCHRONE

présentée par : BEJI Sofiene
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. KHOMH Foutse, Ph. D., président
M. MULLINS John, Ph. D., membre et directeur de recherche
M. GHERBI Abdelouahed, Ph. D., membre et codirecteur de recherche
Mme NICOLESCU Gabriela, Doctorat., membre
M. BONIOL Frédéric, Doctorat., membre externe

DÉDICACE

Je dédie ce travail à :

Ma mère Cherifa,

Que nulle dédicace ne peut exprimer ce que je lui dois, pour sa bienveillance des ma naissance, son affection et son soutien inconditionnel. En témoignage de mon profond amour et ma gratitude pour les sacrifices qu'elle avait consentis.

Mon père Mohamed Faouzi,

Qui n'a jamais cessé de m'encourager, de me conseiller et de me soutenir dès mon enfance. À qui je voudrais adresser ce travail en guise de mon estime et ma reconnaissance.

Mes sœurs Sondes et Selma,

Pour tous les encouragements qu'ils m'ont fournis et pour la complicité qui nous unissent.

Pour mon oncle Adel, mes proches et mes amis,

En témoignage de mes sincères reconnaissances et remerciements pour les conseils et les encouragements qui m'ont procuré.

À tous ceux que j'aime . . . Et à tous ceux qui auraient voulu partager ma joie . . .

REMERCIEMENTS

Je tiens à remercier tous ceux qui de près ou de loin, ont contribué à la réalisation de ce travail. À cet effet, le minimum de justice impose que l'apport de chacun des acteurs soit reconnu :

J'adresse mes remerciements les plus sincères à M. John Mullins, Professeur à Polytechnique Montréal et directeur de ma thèse et M. Abdelouahed Gherbi, Professeur à l'École de technologie supérieure et codirecteur de ma thèse pour leurs encadrements, leurs disponibilités et leurs conseils fructueux tout au long de ma thèse.

J'adresse aussi mes remerciements à M. Sardaouna Hamadou, associé de recherche à Polytechnique Montréal, pour son encadrement, ses conseils et pour sa collaboration dans ce projet.

Je ne saurais terminer sans adresser un mot de reconnaissance à toute ma famille, mes proches et mes amis pour leurs soutiens.

RÉSUMÉ

Les systèmes avioniques modernes sont des systèmes distribués complexes et évolutifs. Ces systèmes sont conçus d'une manière itérative en intégrant à chaque itération une ou plusieurs fonctionnalités. L'ajout de nouvelles fonctionnalités impose des coûts supplémentaires de reconfiguration de telle sorte que l'ensemble du système soit conforme aux exigences temps-réel. Ces systèmes reposent également sur l'adoption d'un protocole de communication déterministe tel que le protocole AFDX. Ce dernier est utilisé dans les avions modernes tels que l'A380 de Airbus et le B787 de Boeing. Il repose sur une communication asynchrone avec limitation de la bande passante. Ce mécanisme permet d'assurer des délais finis de communication. La recherche de plus de déterminisme a poussé la communauté scientifique à chercher d'autres alternatives à AFDX. Le standard Time-triggered Ethernet constitue une bonne alternative. En plus de la communication asynchrone à bande passante limitée, il définit également une communication synchrone.

Suivant le type de communication, les approches de vérification des exigences temps-réel diffèrent. Pour analyser les flux asynchrones, on utilise principalement des approches analytiques. Elles assurent un bon compromis entre performance et pessimisme. Pour les flux synchrones, on s'appuie plutôt sur le formalisme de contraintes pour synthétiser un ordonnancement faisable. La combinaison des deux flux constitue un défi en termes de vérification. De plus, les approches de vérification définies ne modélisent ni l'aspect évolutif ni la notion coût.

Nous définissons dans le cadre de cette thèse une méthodologie à base de formalisme de contraintes pour l'intégration itérative des fonctionnalités avioniques. Cette méthodologie a pour objectif de synthétiser une nouvelle configuration optimale des fonctionnalités synchrones et également d'analyser les flux asynchrones. Pour ce dernier, nous supprimons le pessimisme introduit par les approches analytiques. Afin d'atteindre ces objectifs, nous proposons un modèle formel de l'architecture avionique communicant à travers un réseau TTEthernet ainsi qu'un modèle de configuration associé. Pour assurer les différentes exigences temps-réels, nous formalisons dans ce modèle un ensemble de contraintes. Nous nous intéressons par la suite en une première partie à intégrer les fonctionnalités avioniques synchrones puis à intégrer les différents flux synchrones et asynchrones. Afin d'éviter les risques d'erreurs suite à un codage manuel du programme par contraintes, nous proposons dans une troisième partie une approche d'ingénierie dirigée par les modèles pour la génération automatique des contraintes d'intégration.

ABSTRACT

Modern avionics systems are complex and evolving distributed ones. They are designed iteratively by integrating at each iteration one or more functionalities. Adding new functionality may impose additional reconfiguration costs so that the whole system complies with the real-time requirements. These systems also rely on the adoption of a deterministic communication protocol such as AFDX. The latter is used in modern aircrafts such as the Airbus A380 and the Boeing B787. It relies on asynchronous communication with bandwidth limitations. This mechanism ensures finite communication delays. The search for more determinism encourage the scientific community to look for other alternatives to AFDX. The Time-triggered Ethernet standard is a good alternative. In addition to asynchronous communication with limited bandwidth, it also defines synchronous ones.

Depending on the type of communication, verification approaches of real-time requirements differ. To analyze asynchronous flows, we mainly use analytical approaches. They ensure a good compromise between performance and pessimism. For synchronous flows, we rely instead on constraint formalism to synthesize a feasible scheduling. The combination of the two flows is a challenge in terms of verification. In addition, defined verification approaches do not model neither the evolving aspect nor the cost concept.

We define, in this thesis, a methodology based on constraint formalism for the iterative integration of avionics functionalities. This methodology aims to synthesize a new optimal configuration of synchronous functionalities and also to analyze asynchronous flows. For this latter, we suppress the pessimism introduced by analytical approaches. To this aim, we propose a formal model of architecture communicating avionics through a TTEthernet network as well as an associated configuration model. To ensure the different real-time requirements, we formalize in this model a set of constraints. We are interested in a first part to integrate the synchronous avionics functionalities then to integrate the different synchronous and asynchronous flows. To avoid the risk of errors following manual coding of the constraint program, we propose in a third part a model-driven engineering approach for the automatic generation of integration constraints.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AFDX	–	Avionics Full DupleX switched ethernet
BAG	–	Bandwidth Allocation Gap
BE	–	Best-Effort
BIP	–	Programmation en nombre entier binaire
COP	–	Constraint Optimization Problem
CP	–	Constraint Programming
CSP	–	Problème de Satisfaction de Contraintes
ES	–	End-System
ET	–	Event-Triggered
FIFO	–	First In First Out
FMS	–	Flight Management System
ILP	–	Programmation linéaire en nombre entier
IMA	–	Integrated Modular Avionics
MAF	–	MAjor Frame
MILP	–	Programmation Linéaire Mixte
MLG	–	Main Landing Gear
NLG	–	Noise Landing Gear
PCF	–	Protocol Control Frame
PPCM	–	Plus Petit Commun Multiple
RC	–	Rate-Constrained
RDC	–	Remote Data Concentrator
RO	–	Recherche Opérationelle
SMT	–	Satisfiability Modulo Theories
TT	–	Time-Triggered
TTA	–	Time-Triggered Architecture
TTEthernet	–	Time-Triggered Ethernet
TTP	–	Time-Triggered Protocol
WCD	–	Worst-Case Delay
WCL	–	Worst-Case Latency
WPMS	–	Weighted Partial Max-SMT

LISTE DES NOTATIONS

$func$	Une fonctionnalité avionique
sys	Un système avionique
$\mathcal{DP}$	L'ensemble des chemins de données du réseau
$\mathcal{ES}$	L'ensemble des systèmes terminaux
$\mathcal{F}$	L'ensemble des trames du réseau
$\mathcal{L}$	L'ensemble des liens du réseau
$\mathcal{M}$	L'ensemble des modules IMA
$\mathcal{NS}$	L'ensemble des commutateurs réseaux
$\mathcal{P}$	L'ensemble (distribué) des partitions communicantes
$\mathcal{VL}$	L'ensemble des liens virtuels
$\mathcal{F}^{sys}$	L'ensemble des trames du système sys
HP	L'hyperpériode des trames du réseau
HP_M	L'hyperpériode des partitions du module M
$\mathcal{L}_{sys}(f)$	L'ensemble des liens du système sur lesquels la trame f est envoyée
$\mathcal{F}_{TT}$	Les instances des trames TT
$\mathcal{F}_{RC}$	Les instances des trames RC
Γ	Les structures d'arbre générées par tout les liens virtuels du réseau
Γ_{TT}	Les structures d'arbre générées par les liens virtuels associés aux trames TT
Γ_{RC}	Les structures d'arbre générées par les liens virtuels associés aux trames RC
$\mathcal{F}^{sys}_{inst}$	Les instances des trames TT d'un système sys
$\mathcal{P}^{sys}_{inst}$	Les instances des partitions d'un système sys
$dest$	La fonction qui retourne les partitions destinations d'une trame
$first$	Fonction qui retourne le premier lien d'un lien virtuel
$last$	Fonction qui retourne l'ensemble des derniers liens d'un lien virtuel
$leaves$	Fonction qui retourne les modules feuilles d'un lien virtuel
$links$	Fonction qui retourne l'ensemble des liens d'un lien virtuel
$root$	Fonction qui retourne le module racine d'un lien virtuel
$source$	Fonction qui retourne la partition source d'une trame

$Instances(f)$	Les différentes instances de la trame f
$Instances(P)$	Les différentes instances de la partition P
$avail$	Fonction qui définit la date de disponibilité de l'information transmise par une trame RC
$regul$	Fonction partielle qui indique la date où le régulateur de trafic débloque la transmission
$f.BAG$	Le BAG d'une trame f de type RC
$f.deadline$	La date limite pour la trame f
$f.Rate$	Le quota réel d'une trame f de type RC
$f.C$	La durée de transmission de la trame f sur un lien de données
$P.T$	La période de la partition P
$P.C$	La durée de transmission de la partition P sur un lien de données
$f.T$	La période de la trame f
α	La fonction qui associe les partitions aux systèmes terminaux
Δ	La fonction qui associe chaque trame f à l'ensemble non vide des partitions destinations qui consomment les messages transportés par f
Π	La fonction qui associe chaque trame f à l'ensemble non vide des partitions destinations qui consomment les messages transportés par f
Ω	La fonction qui associe chaque partition P à l'ensemble des trames produites par P
ϕ	La configuration des trames TT
ψ	La configuration des modules IMA
$P^M_{i,k}$	La k$^{\text{ème}}$ instance de la i$^{\text{ème}}$ partition du module M
$f_{i,k}$	La k$^{\text{ème}}$ transmission de la trame f_i
$f^l_{i,k}$	La k$^{\text{ème}}$ instance de la i$^{\text{ème}}$ trame sur le lien l
$f_{i,k}.min\_avail$	La première date de disponibilité de la k$^{\text{ème}}$ transmission de f_i
$f_{i,k}.max\_avail$	La dernière date de disponibilité de la k$^{\text{ème}}$ transmission de f_i
$children(f^l_{i,k})$	L'ensemble des fils de $f^l_{i,k}$
$pred(f^l_{i,k})$	Le prédecesseur (père) de $f^l_{i,k}$
$I(sys,func)$	Les configurations possibles aprés l'intégration de la fonctionnalité $func$
$\kappa(f^l_{i,k})$	Le coût de re-ordonnancer $f^l_{i,k}$
$\omega(P^M_{i,k})$	Le coût de re-ordonnancer $P^M_{i,k}$
$\delta^M_{i,k}$	La fonction indicateur retournant 1 si l'instance $P^M_{i,k}$ est reconfigurée

$\sigma_{i,k}^{l}$	La fonction indicateur retournant 1 si l'instance $f_{i,k}^{l}$ est reconfigurée
$\partial_{i,k}^{l}$	Fonction qui détermine pour l'ancien ordonnancement Time-Triggered (TT) et l'ordonnancement actuel si l'instance $f_{i,k}^{l}$ est reconfigurée
tt_switch_delay	La durée de traitement d'une trame TT par un commutateur réseau
rc_switch_delay	La durée de traitement d'une trame RC par un commutateur réseau

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les années 70 ont marqué l'histoire de l'avionique par un avènement technologique. Ce progrès est survenu suite à l'intégration des systèmes numériques embarqués qui ont commencé déjà à remplacer les vieux systèmes analogiques (Gangl, 2006). De nos jours, les avions reposent de plus en plus sur leurs systèmes embarqués. Ces systèmes intègrent plusieurs fonctionnalités allant de la plus critique tel que le contrôle de fuel, le contrôle du vol au moins critique tel que le divertissement des voyageurs. L'importance de ces systèmes embarqués dans les avions modernes se confirme aussi de point de vue du coût budgétaire. La part du coût de développement de ses systèmes du coût total de développement des avions est estimée à $35 - 40\%$ pour les avions civils et à 50% pour les avions militaires (Bieber et al., 2012).

Mis à part de ces coûts de développement importants, les systèmes avioniques sont en évolution continue et le nombre de fonctionnalités à intégrer ne cesse d'augmenter au fil du temps. À titre indicatif, le nombre de lignes de code des fonctionnalités embarquées dans l'avion a augmenté d'une centaine dans les années 80 à une échelle de 10^5 dans les années 2010 avec l'apparition du $A380$ (Chuyanov et al., 2014). L'intégration de nouvelles fonctionnalités peut exiger de reconfigurer le système avionique pour que l'ensemble des fonctionnalités répondent aux exigences temps-réels critiques. Cette reconfiguration induit des coûts supplémentaires dus à la recertification et/ou l'allocation des ressources supplémentaires. La maitrise des coûts est ainsi devenue une préoccupation majeure des avionneurs pour avoir des avions plus compétitifs sur le marché.

L'essor de l'avionique embarqué a bien commencé avec le développement de l'architecture fédérée (Watkins and Walter, 2007). Cette architecture se caractérise par la présence de fonctions inter-connectées et qui disposent de leurs propres ressources. Néanmoins, elle a connu ses limites avec l'augmentation du nombre de fonctionnalités, le coût de maintenance qui est devenu si élevé et le poids des différents équipements qui devient assez contraignant même de point de vue conceptuel.

Afin de surpasser ces limites, les avionneurs se sont tournés vers un nouveau modèle d'architecture qui favorise plutôt le partage des ressources. C'est dans ce contexte que l'architecture avionique modulaire intégrée notée IMA (ARINC, 2006) est devenu un nouveau standard. Elle est basée essentiellement sur le concept de ressources modulaires qui sont partagées par quelques fonctions en évitant toute interférence entre elles. Les fonctions ainsi exécutées peuvent avoir différents niveaux de criticité et pour des raisons de sûreté elles doivent être isolées les unes des autres. On les appelle ainsi partition et l'isolation est faite par des

mécanismes d'isolement spatial et temporel. L'isolement spatial est assuré par l'assignation statique de toutes les ressources du module à la partition et l'isolement temporel est plutôt implémenté par l'allocation d'une fenêtre de temps périodique pour l'exécution de chaque partition (ARINC, 2006).

L'adoption de l'architecture modulaire qui favorise le partage a son impact sur le réseau. Les liens de communication point-à-point doivent aussi évoluer vers des réseaux partagés. Ethernet constitue un standard célèbre pour les réseaux partagés. Ce dernier est largement utilisé et a prouvé sa fiabilité et son applicabilité dans un large spectre de domaines. Malheureusement, ce protocole ne garantit pas les exigences temps-réel strictes pour les applications critiques. Les messages transmis à travers des réseaux Ethernet n'ont pas la garantie d'atteindre leurs destinations dans les délais fixés.

La nécessité de trafic prédictible a ainsi poussé vers l'exploration de possibles extensions tel que l'Avionics Full DupleX switched ethernet (AFDX) (ARINC, 2009). Ce protocole repose sur le principe de limitation de la bande passante destinée pour la communication de chaque nœud. Ainsi, les scénarios de congestion et la perte des messages sont évités et la garantie de l'arrivée de ces derniers à destination dans des délais finis est assurée. AFDX est utilisé dans les avions modernes tels que le $A380$ d'Airbus et le $B787$ de Boeing.

La quête de plus de déterminisme a poussé la communauté scientifique à chercher d'autres alternatives à AFDX. C'est dans ce contexte que Time-Triggered Ethernet (TTEthernet) (SAE, 2011) a vu le jour et adopté comme nouveau standard. En plus de la communication asynchrone avec limitation de bande passante notée Rate-Constrained (RC), TTEthernet définit un autre type de trafic déclenché par le temps noté Time-Triggered (TT). La communication peut être activée à des instants fixes dans le temps. Un des avantages de TTEthernet est que la combinaison du trafic RC et TT est possible. Les applications ainsi définies sur les réseaux AFDX sont en parfaite compatibilité avec des réseaux TTEthernet.

Plusieurs approches ont été proposées pour valider les exigences temps-réel de ce type de réseaux. Pour le trafic RC, le calcul réseau (Cruz, 1991a,b; Zhao et al., 2017) et l'approche par trajectoire (Martin and Minet, 2006) sont les principales approches utilisées pour borner les délais de transmission au pire cas. Ces bornes sont utilisées pour prouver que les trafics RC ne violent pas leurs exigences de dates limites.

Malheureusement, ces approches ne conviennent pas pour synthétiser un ordonnancement pour les flux TT déterministes. Les techniques de programmation par contraintes sont maintenant largement utilisées pour synthétiser de tels ordonnancements. Avec le formalisme de contraintes, les exigences sont modélisées par un ensemble de contraintes et une solution pour le système de contraintes fournit un ordonnancement faisable. La diversification des

approches utilisées pour la vérification des flux TT et RC rend ainsi la recherche d'un ordonnancement faisable qui vérifie les exigences temps réels pour ces deux types de trafic une tâche difficile.

D'autre part, les approches définies pour vérifier les flux RC et mentionnées ci-dessus sont analytiques. Ces approches reposent sur des hypothèses pessimistes pour analyser les pires cas de transmission. Ainsi, la suppression de ce pessimisme en adoptant une approche de vérification exacte constitue aussi un défi en tant que tel.

D'une autre perspective, c'est non seulement les systèmes avioniques qui sont en évolution continue, mais leur méthode de développement est aussi incrémentale. On affirme dans (Nam et al., 2014) que pour minimiser le besoin de reconcevoir et de recertifier, les nouvelles fonctionnalités sont ajoutées d'une manière incrémentale. D'où la naissance du concept de certification incrémentale avec l'A380 (Andrillon, 2008) qui stipule que l'ajout de nouvelles applications logicielles n'implique pas de recertifier tout le système.

À notre connaissance, les approches de vérification présentées dans ce cadre ne suivent pas cette nouvelle tendance de conception. En effet, elles ne modélisent ni l'aspect incrémental ni le coût de reconfiguration. Ces dernières s'intéressent à la vérification des exigences temps-réel d'un système en entier et optimisent d'autres critères que le coût de reconfiguration.

Objectifs de recherche

Nous fixons dans le cadre de cette thèse comme objectif de définir une approche qui permet de synthétiser un ordonnancement faisable tout en vérifiant les exigences temps-réel pour les fonctionnalités communicantes avec des trafics TT et RC. Cette approche cherchera aussi à supprimer le pessimisme introduit par les approches analytiques. Elle doit également modéliser l'aspect incrémental et les coûts de reconfiguration de ces systèmes. Nous nommons cette approche par : *approche d'intégration itérative des systèmes avioniques communicants en mode synchrone et asynchrone*.

L'approche d'intégration itérative illustrée par la figure 1.1 résout le problème défini comme suit. Étant données les entrées suivantes :

— Un ensemble de fonctionnalités déjà intégrées qui peuvent communiquer soit en mode synchrone ou asynchrone.

— L'ordonnancement des partitions IMA et du trafic TT des fonctionnalités déja intégrées et établi en mode déconnecté.

— Un ensemble de fonctionnalités à ajouter qui peuvent aussi communiquer en mode

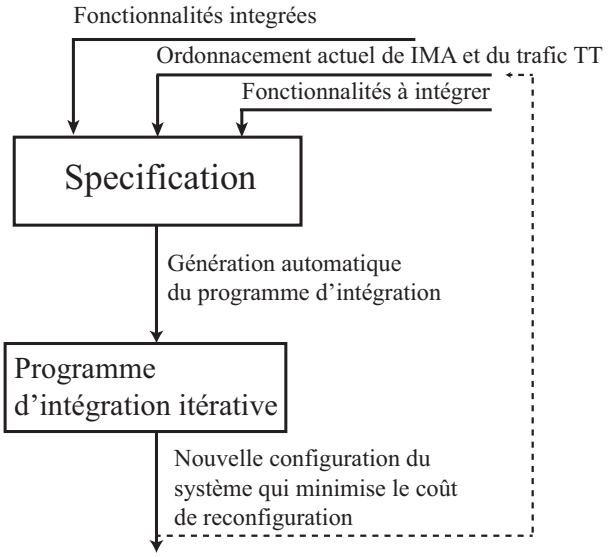


Figure 1.1 Approche d'intégration itérative

synchrone et asynchrone

Peut-on intégrer les nouvelles fonctionnalités ? Si oui, quel pourrait être une nouvelle configuration optimale du système qui minimise les coûts de reconfiguration ? Les coûts peuvent exprimer ici les ressources dépensées sur la reconfiguration et/ou la certification dues aux modifications introduites durant le processus d'intégration. Une nouvelle configuration du système se caractérise par le nouvel ordonnancement IMA et du trafic TT.

Méthodologie

Comme le montre la figure 1.1, nous découpons notre problème en deux sous-problèmes. Le premier sous problème consiste à trouver une formalisation à base de contraintes pour le problème d'intégration itérative. Afin de maitriser la complexité de ce sous-problème, nous étudions dans une première partie l'intégration des fonctionnalités avioniques communicantes en mode TT et nous abordons dans une deuxième partie l'intégration itérative des flux TT et RC. Quant au deuxième sous-problème, il fait l'objet de notre troisième partie de la thèse. Il consiste a générer automatiquement à partir d'une spécification du problème d'intégration le programme d'intégration adéquat pour générer une nouvelle configuration optimale.

Contributions de la thèse

Nous détaillons dans cette section les différentes contributions pour chaque partie de la thèse.

Contribution 1 : Intégration itérative des fonctionnalités avioniques communicantes en mode TT. Nous présentons dans cette partie un modèle formel pour les systèmes avioniques IMA déployés sur une architecture IMA. Nous présentons également un modèle de configuration pour ce type de système qui capte son aspect évolutif en formalisant le processus d'intégration. En nous basant sur ce modèle formel, nous définissons deux ensembles de contraintes. Un premier ensemble, dit ensemble de contraintes dures, vérifie la compatibilité avec les standards IMA et TTEthernet ainsi que les exigences de latence et un deuxième ensemble de contraintes, dites souples, servent à minimiser les coûts de reconfiguration. Considérant l'ensemble des contraintes dures et souples, nous caractérisons formellement notre approche à base de formalisme Satisfiability Modulo Theories (SMT) pour l'intégration des fonctionnalités avioniques communicantes en mode TT. Cette approche satisfait les contraintes dures et minimise les coûts associés aux contraintes souples.

Contribution 2 : Intégration itérative des flux déclenchés par le temps et à quotas limités. Nous présentons, dans cette partie, une spécification formelle d'un ensemble complet de contraintes représentant les exigences temporelles et d'intégration des communications TT et RC. Nous définissons également une nouvelle approche incrémentale à deux phases et basée sur le formalisme de contraintes pour l'optimisation de l'intégration des flux TT et RC. À la différence des approches analytiques présentés pour l'analyse de l'ordonnançabilité des flux RC, l'approche de programmation par contrainte que nous présentons enlèvera le pessimisme. Nous présentons également une heuristique qui améliore la résolution du problème d'optimisation et nous discutons l'utilité de notre heuristique à travers un cas d'étude.

Contribution 3 : Approche d'ingénierie dirigée par les modèles pour la génération automatique des contraintes d'integration itérative. Nous présentons dans cette partie une approche d'ingénierie dirigée par les modèles qui fournit les abstractions nécessaires (méta-modèles) et les processus de transformation pour permettre la génération automatique des contraintes d'intégration.

Liste des publications

- **Partie 1**

  Article Publié

  — S. Beji, S. Hamadou, A. Gherbi, et J. Mullins, "Smt-based cost optimization approach for the integration of avionic functions in ima and ttethernet architectures",

dans Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications. IEEE Computer Society, 2014, pp. 165–174.

Article Soumis

— S. Beji, S. Hamadou, A. Gherbi, et J. Mullins, "On the optimization of the integration of avionic functions", Real-Time Systems (RTS) 2018

- **Partie 2**

  Article Accepté

  — S. Beji, S. Hamadou, A. Gherbi, et J. Mullins, "Iterative integration of ttethernet network flows", International Journal of Critical Computer-Based Systems, vol. 8, no. 1, p. 1, 2018.

- **Partie 3**

  Article Publié

  — S. Beji, A. Gherbi, J. Mullins, et P.-E. Hladik, "Model-driven approach to the optimal configuration of time-triggered flows in a ttethernet network", dans System Analysis and Modeling. Technology-Specific Aspects of Models, J. Grabowski et S. Herbold, éds. Cham : Springer International Publishing, 2016, pp. 164–179

Plan de la thèse

Cette thèse est organisée en cinq chapitres. Nous présentons dans le chapitre 2 les notions préliminaires nécessaires pour la compréhension de la thèse. Dans le chapitre 3, nous examinons différents travaux de recherches liés à notre problématique. Nous présentons dans le chapitre 4 notre approche d'intégration itérative des fonctionnalités communicantes en mode TT. Nous développons par la suite dans le chapitre 5 notre approche d'intégration des différents flux TT et RC. Nous exposons dans le chapitre 6 notre approche à base d'ingénierie dirigée par les modèles pour la génération automatique des programmes d'intégration et nous concluons dans le chapitre 7 la thèse.

CHAPITRE 2 NOTIONS PRÉLIMINAIRES

Nous présentons dans ce chapitre les notions préliminaires nécessaires pour la compréhension de cette thèse. Nous présenterons dans la section 2.1 deux approches pour la conception des systèmes temps-réels distribués. Ceci nous permettra dans la section 2.2 de présenter l'architecture déclenchée par le temps. Dans la section 2.3, nous nous intéresserons particulièrement aux concepts de l'architecture avionique modulaire intégrée. Ceci nous mènera dans la section 2.4 à étudier les protocoles de communication AFDX et TTEthernet qui sont bien adaptés avec une telle architecture. Le protocole TTEthernet définit des flux qui ne suivent pas le même mode de transmission. Ainsi, nous introduirons dans la section 2.5 différents techniques d'intégration des flux. Nous présenterons dans la section 2.6, les concepts de base de l'ingénierie dirigée par les modèles. Par la suite, nous nous intéresserons aux formalismes de résolution par contraintes. Pour ce faire, nous présenterons dans la section 2.7 la satisfiabilité modulo théories et dans la section 2.8 les concepts de base de la programmation par contraintes. Vers la fin, nous présenterons dans la section 2.9 différentes techniques d'analyse d'ordonnannçabilité des flux RC.

2.1 Approches de conception des systèmes temps réels distribués

Suivant les mécanismes de déclenchement d'exécution des tâches et de la communication, les systèmes temps-réels distribuées peuvent être conçues suivant deux approches : systèmes *déclenchés par le temps* notés TT et systèmes *déclenchés par des évènements* notés Event-Triggered (ET) (Kopetz, 1991).

Dans un système ET, les activités du système sont déclenchés suite à l'occurrence d'évènements significatifs. Un évènement significatif se définit dans (Kopetz, 1991) comme étant un changement d'état d'un objet ou d'une entité temps réel et que ce changement doit être traité par le système. D'une manière générale, dans un système ET, les évènements significatifs sont implémentés par le mécanisme d'interruptions qui notifient l'unité de contrôle de l'occurrence de ces évènements. Contrairement aux systèmes ET, dans les systèmes TT, les activités sont initiées à des points prédéfinis dans le temps. L'ordonnancement du système se fait en mode déconnecté et ceci avant même le déploiement.

Une comparaison entre les deux approches a été établie dans (Kopetz, 1991) et (Wajs and P., 2003). Dans les systèmes TT, chaque tâche est observée d'une manière périodique. Bien évidemment, ceci suppose la définition d'un monde périodique. Les systèmes ET sont plus

flexibles vu l'aspect dynamique d'exécution et de communication. Par contre, les systèmes TT sont plutôt déterministes et prédictibles. En termes d'utilisation des ressources, généralement les systèmes ET garantissent une meilleure utilisation vu que les fenêtres de temps dans le modèle déclenché par le temps doivent être assez larges pour prévoir une charge maximale. La définition du modèle déclenché par le temps nous conduit à définir dans la section suivante le modèle d'architecture déclenchée par le temps notée Time-Triggered Architecture (TTA).

2.2 Architecture déclenchée par le temps

La TTA a été introduite par le groupe de Herman Koepetz à TU de Vienne et commercialisée par la suite par la société TTTech. L'architecture TTA (Kopetz and Bauer, 2003) se définit comme étant une infrastructure informatique pour la conception et l'implémentation des systèmes embarqués distribués essentiellement à sureté critiques et avec environnements dépendants. Une architecture TTA se caractérise par la distribution des applications sur des ensembles de nœuds et de grappes presque autonomes. Elle repose aussi sur la définition d'un protocole de communication déclenché par le temps noté Time-Triggered Protocol (TTP).

Le TTP offre un service de synchronisation des horloges tolérant aux fautes offrant ainsi un service de communication synchrone et sans congestion. Le temps global dans le cadre d'une architecture TTA sert à définir une interface entre les différentes applications distribuées dont le but est de simplifier leurs communications, garantir leurs rapidités et de détecter rapidement les erreurs. Par la suite, plusieurs protocoles déclenchés par le temps ont vu le jour et pour des besoins divers. On cite FlexRay (Consortium et al., 2005) pour l'industrie automobile et TTEthernet (SAE, 2011) pour le domaine aérospatial. Nous nous intéressons particulièrement dans le cadre de cette thèse au protocole TTEthernet. Avant de détailler les concepts de base de TTEthernet, nous introduisons dans la section suivante les concepts de base de l'architecture avionique modulaire intégrée notée IMA.

2.3 L'architecture avionique modulaire intégrée : IMA

IMA a été adoptée dans les années 90 comme modèle d'architecture pour les systèmes avioniques. Ce type d'architecture repose sur la définition d'un certain nombre de modules de calculs capables de supporter plusieurs applications. Les applications hébergées sur des modules distants peuvent communiquer les unes avec les autres à travers un réseau partagé. Un système IMA interagit avec son environnement à travers un ensemble de capteurs et d'actionneurs. Ces capteurs et ces actionneurs ayant leurs propres formats de données sont connectés sur un système avionique à travers un type de module particulier appelé *Remote*

Data Concentrator (RDC). Les RDC jouent le rôle de convertisseurs de données dans le format adéquat.

L'architecture avionique est centrée sur deux modèles essentiels : Le modèle d'exécution des modules distribués et le modèle de communication (Lauer, 2012). Le modèle d'exécution décrit comment les fonctions sont exécutées dans les modules distribués. Ce modèle est défini par la spécification IMA, standardisée sous la norme *ARINC653* (ARINC, 2006). Quant au modèle de communication, il spécifie plutôt comment les données sont transmises entre les divers nœuds distribués. Nous réservons la section 2.4 à présenter les détails de communication d'un système avionique et nous nous intéressons dans le reste de cette section à présenter d'une manière globale le fonctionnement d'un système IMA. Dans le reste de cette section, nous détaillons dans une première partie le modèle d'exécution IMA. Dans une deuxième partie, nous illustrons à travers l'exemple d'un sous système du système de gestion de vol noté *FMS* le fonctionnement global d'un système avionique ainsi que l'exigence de latence dont nous chercherons à vérifier par notre approche d'intégration itérative.

2.3.1 Modèle d'exécution IMA

L'architecture IMA repose sur la définition des modules distribués. Chaque module est un processeur sur lequel plusieurs partitions peuvent être exécutées. L'ordonnancement des partitions au sein du même module est répété périodiquement. Une telle période est nommée le *MAF* (MAjor Frame) et la longueur de la *MAF* est l'hyper-période d'ordonnancement notée par *HP*. Elle est définie comme étant le plus petit multiple commun des périodes des partitions au sein du module. La figure 2.1 illustre l'ordonnancement de deux partitions notées P_{11} et P_{12} du module M. Chaque partition comme le montre la figure est caractérisée par trois paramètres : la période, la longueur d'exécution et l'offset (décalage par rapport au début de la période)

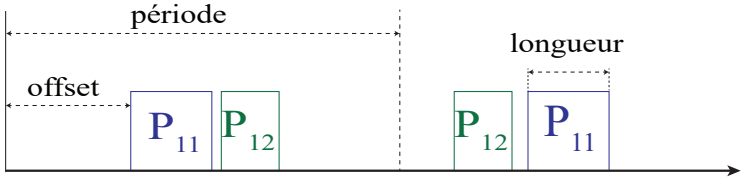


Figure 2.1 Un exemple d'ordonnancement IMA

Les partitions déployées dans des modules distants collaborent les unes avec les autres pour assurer une fonctionnalité spécifique. La communication entre diverses partitions est assurée par le moyen de deux types de réseaux que nous détaillerons dans la prochaine section. Nous introduisons aussi dans la prochaine section le fonctionnement global d'un système avionique à travers l'exemple d'un sous-système du FMS et nous illustrons par la même occasion l'exigence de latence du système.

2.3.2 Système IMA : Illustration et Exigence de latence

Nous présentons dans cette section un exemple de système IMA qui représente un sous-système du système de gestion de vol FMS. Cet exemple est pris de (Lauer, 2012). Il sert à contrôler l'affichage des informations statiques de la navigation dans les écrans des pilotes. L'architecture générale de ce système est illustrée par la figure 2.2

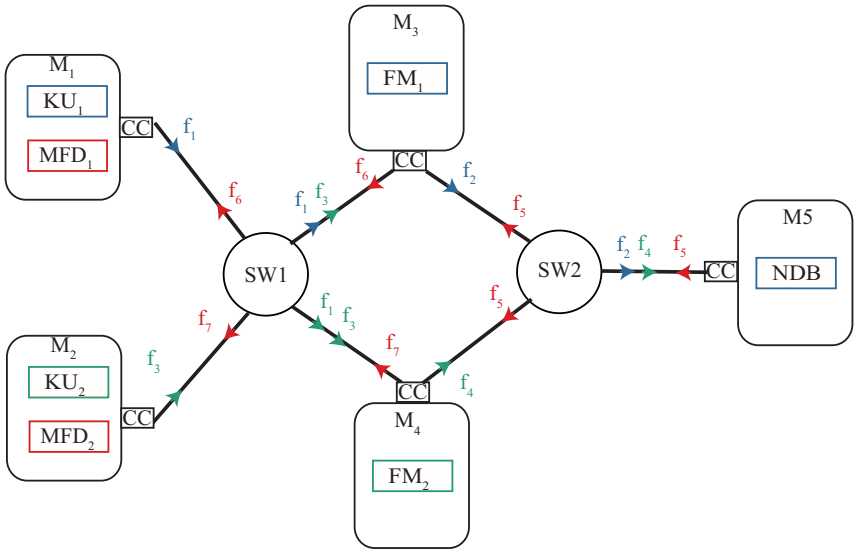


Figure 2.2 Système IMA : Exemple du FMS

Deux partitions KU_i $(i = 1; 2)$, *Keyboard and cursor Unit*, envoient des requêtes de demande d'information du waypoint provenant du pilote et du copilote. Chaque requête est transmise à FM_1 et FM_2, les deux partitions *Flight Manager*. Chaque *FM* envoie séparément des requêtes pour le *NDB*, *Navigation DataBase*, à travers une communication mono diffusion pour récupérer les informations du waypoint. Chaque FM_i $(i = 1; 2)$ transmet le résultat de la requête à son MFD_i (*Multi Functional Display*) associé, qui à son tour affiche l'information dans l'écran correspondant. Les trames sont notées par f_i $(i = 1; \cdots; 7)$. L'ensemble des partitions du FMS et le flux de données entre eux permettent d'assurer la fonctionnalité de

gestion de vol. Nous dénotons ainsi cet ensemble par *chaine fonctionnelle* et nous l'illustrons par la figure 2.3.

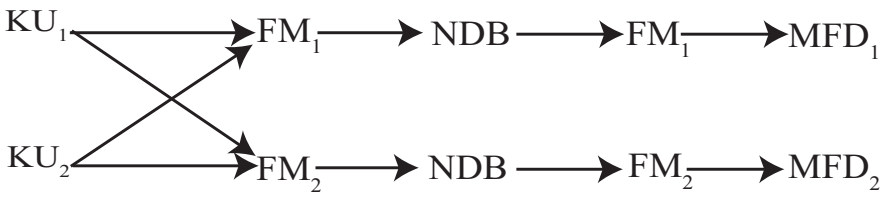


Figure 2.3 Chaine fonctionnelle du FMS

Dans (Lauer, 2012), on a identifié l'exigence de latence sur un système IMA. Deux types de latences peuvent être définies sur le système.

Le premier type ou *latence de bout en bout* permet de contraindre la durée maximale écoulée depuis le début de l'exécution de la partition source jusqu'à la fin d'exécution de la partition destination. Un exemple dans ce cas serait de contraindre l'affichage sur l'écran du moment de la détermination du résultat par FM_i à l'affichage du résultat sur l'écran par la partition MFD_i.

Le deuxième type ou *latence d'une chaine fonctionnelle* contraint le temps global à travers toute une chaine fonctionnelle et sert à limiter le temps de réponse d'une fonctionnalité au complet. Considérons la chaine fonctionnelle donnée par la figure 2.3, nous pouvons citer l'exemple suivant : Les partitions MFD_i doivent afficher les informations de navigation dans les écrans des pilotes après un délai maximal de 700 ms commençant de la requête d'un des deux pilotes.

2.4 Réseaux avioniques

Les systèmes avioniques actuels reposent en grande partie sur une communication conforme au protocole *AFDX* (ARINC, 2009) standardisée sous le *ARINC 664 part 7*. Il a été utilisé pour la première fois par Airbus pour spécifier le réseau du A380 et puis par Boeing dans les avions de type B777. Après l'adoption de TTEthernet (SAE, 2011) comme nouveau standard réseau dans le domaine aérospatial sous la référence *AS 6802*, les efforts se tournent vers la définition des modèles de communication conformes à ce standard. Nous présentons en premier lieu les concepts de base d'une communication AFDX et nous présentons par la suite les principes d'une communication TTEthernet.

2.4.1 Réseaux AFDX

AFDX est un standard réseau basé sur le protocole TTEthernet. Il est défini à la base par Airbus sous le Brevet (Moreaux, 2005) et devenu par la suite une référence pour une communication déterministe des systèmes avioniques distribués. Ce dernier offre une communication ET pour les applications à sureté critique et définit deux types de nœuds : (1) les producteurs/ consommateurs qui sont les modules IMA distribués et (2) les commutateurs réseau.

Les réseaux AFDX sont conçus pour assurer les exigences temps réels suivantes :

— fiabilité de la transmission de données (les pertes de trames et la congestion ne sont pas tolérées) ;

— un mécanisme d'isolation des fautes entre les différents flux ;

— une haute disponibilité ;

— service de transfert déterministe (Une latence et une gigue bornée).

La transmission des données se fait à travers des liens virtuels. Un lien virtuel est un objet conceptuel de communication dont le but est de définir un flux unidirectionnel d'une source à plusieurs destinations. Il virtualise ainsi un bus avionique pour chaque flux dans lequel il est le seul à émettre. Chaque lien virtuel dispose d'un quota qui limite l'utilisation de la bande passante du réseau. Ce quota reste fixe indépendamment des taux d'utilisation réels des autres liens virtuels partageant le même réseau. La limitation de la bande passante est un mécanisme de contrôle de flux qui permet de garantir d'une part une ségrégation (c-à-d. isolation logique) entre les différents flux de communication et d'une autre part d'assurer les propriétés déterministes du réseau par la garantie d'une distribution finie des temps d'arrivées. Un lien virtuel est défini par (Lauer, 2012) :

— Un identifiant unique ;

— Une ou plusieurs adresses destinations ;

— Un chemin utilisé pour atteindre chaque destination ;

— La taille minimale et maximale du paquet ;

— Le temps minimal entre l'émission de deux paquets consécutifs est appelé Bandwidth Allocation Gap (BAG)

Un émetteur peut envoyer seulement dans un lien virtuel au plus un paquet avec une taille maximale chaque intervalle de temps de durée BAG. Si on note par s_{max} la taille maximale d'un paquet, le débit maximal d'un lien virtuel est donné par :

$$\rho = \frac{s_{max}}{BAG} \tag{2.1}$$

Le transfert de données suivant le protocole AFDX est caractérisé dans chaque flux par les deux paramètres suivants :

— La BAG : L'intervalle minimal séparant l'envoi des deux premiers bits de deux paquets consécutifs.

— La gigue : Un délai introduit par l'ordonnanceur afin d'acheminer des trames venant de plusieurs liens virtuels dans le même lien de données.

Physiquement, un régulateur assure la régulation de trafic avec le débit maximal d'une seule trame chaque BAG. La figure 2.4 illustre la fonction de régulation du régulateur de trafic pour la trame f_i. La partie gauche visualise le trafic en entrée au régulateur (trafic aléatoire) quant à la partie droite elle visualise plutôt la sortie du régulateur de trafic.

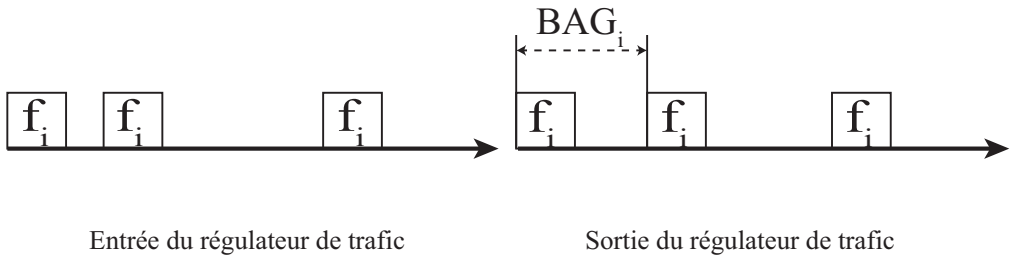


Figure 2.4 Régulation de trafic AFDX

Un lien donné de notre réseau AFDX peut participer à la formation de plusieurs liens virtuels. Il est utile dans ce cadre d'ordonnancer les trames provenant de différents liens virtuels sur le même lien physique. La figure 2.5 illustre l'ordonnancement des trames venant de trois différents liens virtuels. L'ordonnanceur prend en entrée le trafic régulé des trois liens virtuels. Afin de combiner ces trafics, l'ordonnanceur peut introduire un délai appelé gigue dans certains flux de trafic. Cette technique permet de combiner les trois différents flux de trafic dans le même médium de communication.

La gigue introduite par l'ordonnanceur est bornée supérieurement par le *max_gigue* défini

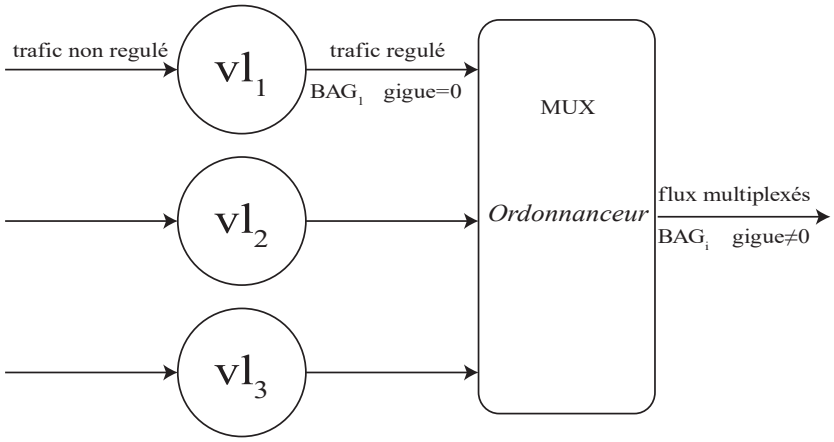


Figure 2.5 Ordonnacement des flux AFDX

par les contraintes 2.2

$$\begin{aligned} max\_gigue &\leq 40\ \mu s + \frac{\sum_{vl_i \in \mathcal{VL}} (180 + L_i^{max})}{Nbw} \\ max\_gigue &\leq 150\ \mu s \end{aligned} \tag{2.2}$$

où , Nbw est le débit du canal en $bits/s$ et L^{max} est la taille maximale d'une trame en bits. La première contrainte définit la max_gigue suivant les tailles maximales des trames provenant de chaque lien virtuel partageant le même canal. La max_gigue doit être aussi en dessous d'un seuil fixe de $150\mu s$ défini par la deuxième contrainte. Nous présentons dans la section 2.4.2 les concepts de base d'une communication TTEthernet.

2.4.2 Réseaux TTEthernet

TTEthernet est un protocole réseau de couche 2. Il définit une stratégie de synchronisation d'horloges dans les systèmes distribués et il supporte deux types de trafics (Steinhammer et al., 2006) (Kopetz, 2008) : Trafic déclenché par des événements ET et trafic déclenché par le temps TT.

Le trafic TT est adéquat pour les applications critiques. Une fenêtre de temps est réservée pour chaque trame sur chaque lien et un ordonnancement global du réseau est établi en mode déconnecté. Chaque fenêtre de temps est répétée périodiquement et chaque trame transmise au cours de sa fenêtre dédiée a la garantie d'arriver à destination avant un délai maximal.

Un réseau TTEthernet (Kopetz et al., 2005) peut être conçu selon deux configurations dis-

tinctes : configuration standard ou configuration à sureté critique. La figure 2.6 illustre ces deux configurations. La configuration standard est illustrée par la partie gauche. Celle de droite illustre plutôt la configuration pour les applications à sureté critique.

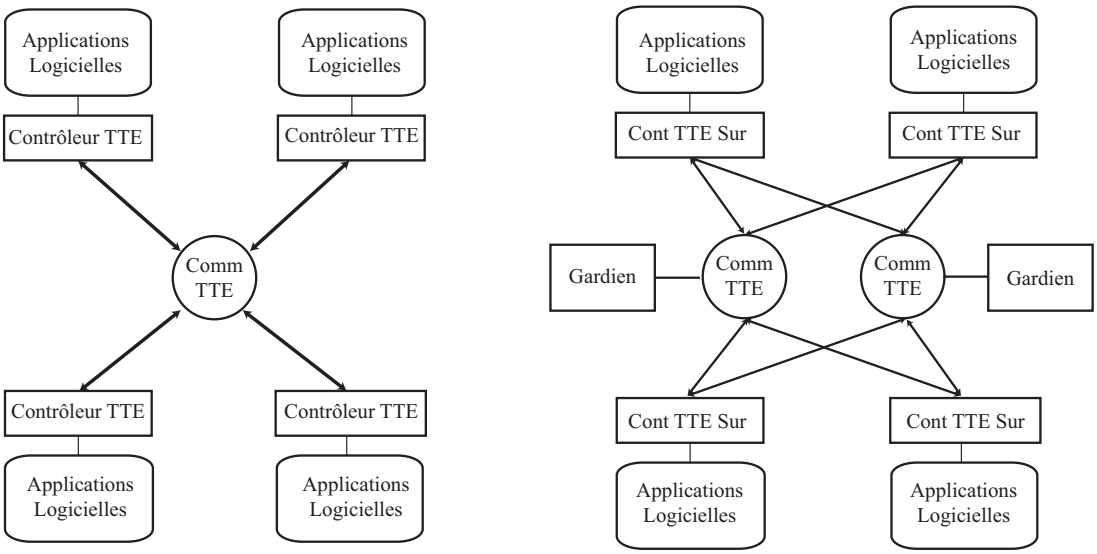


Figure 2.6 Configurations du réseau TTEthernet

Dans la configuration standard, les nœuds terminaux sont composés par les applications logicielles qui communiquent les unes avec les autres. Le contrôleur TTEthernet standard a deux ports. Le premier est relié au nœud terminal et le second au routeur TTEthernet. Il permet d'enclencher l'envoi des messages TT durant les fenêtres de temps dédiés. Le routeur TTEthernet relaie les messages TT suivant le protocole TTEthernet.

La configuration à sûreté critique est plus adaptée au contexte avionique. Dans cette configuration, les nœuds terminaux et les routeurs sont dupliqués pour des raisons de tolérance aux fautes. Le contrôleur TTEthernet à sureté critique envoie le message à deux routeurs différents afin d'éviter la propagation de fautes quand un routeur échoue. Pour la même raison, les nœuds terminaux sont aussi dupliqués. Le suivi des fautes est assuré par les gardiens. Ces derniers, connectés aux routeurs, analysent le trafic et gèrent par conséquent les composants défectueux.

Dans le type de trafic déclenché par les événements, deux sous-classes sont définies : la sous-classe à quota limité noté RC et la sous-classe Best-Effort (BE). Pour la sous-classe RC la garantie de service est assurée en limitant la bande passante allouée pour chaque flux et ainsi éviter les scénarios de congestion et de perte de messages. Pour la sous-classe BE, aucune

garantie de transmission n'est assurée.

Le travail de (Steiner et al., 2009) constitue une référence pour la spécification et l'analyse comparative entre les différentes classes de trafic TTEtherenet. Les caractéristiques de chaque type de trafic peuvent être résumées ainsi :

Trafic TT

Ce type de trafic exige une synchronisation entre les différents nœuds participant afin de produire le temps global du système. La synchronisation est réalisée par l'échange périodique d'un type particulier de trames nommées Protocol Control Frame (PCF). À part le rôle d'établir une synchronisation des nœuds TT, l'échange des messages PCF permet aussi de maintenir cette synchronisation. Chaque nœud TT maintient une table dans laquelle il sauvegarde l'instant d'envoi d'une trame TT dans le réseau. Le trafic TT a les avantages et les inconvénients suivants :

- *Avantages :*
  - Le temps de latence et la gigue sont petits
  - Le conflit d'utilisation d'un canal partagé est évité en réservant une fenêtre de temps dédié pour chaque trame.
  - Une stratégie itérative d'ordonnancement est possible.
- *Inconvénients :*
  - L'ordonnancement dépend du service de synchronisation.
  - Une méthode complexe pour établir et maintenir la synchronisation des horloges locales.

Trafic RC

Le trafic RC est transmis selon le paradigme de communication à quota limité. Chaque expéditeur qui émit selon la politique à quota limité assure qu'il ne dépasse pas un quota fixe d'utilisation de bande passante. Le trafic à quota limité a les caractéristiques suivantes :

- *Avantages :*
  - La synchronisation n'est pas nécessaire pour la transmission.
- *Inconvénients :*
  - Les scénarios des pires cas doivent être considérés afin de calculer les tailles requises des files d'attente et pour assurer que la latence est inférieure à une borne fixée.

— Les nœuds terminaux doivent assurer la fonction de régulation de trafic et les routeurs vérifient les quotas de transmission.

— Des services supplémentaires tels que l'intégrité et la redondance doivent être considérés pour assurer le déterminisme et pour pallier aux faibles qualités temporelles de ce mode de transfert.

— L'ajout d'un nouveau composant au système nécessite le recalcul des paramètres de performance tel que la taille des files et le temps de latence.

Trafic Best-Effort

Avec le type de trafic BE, nous n'avons pas la garantie des qualités temps réel de la transmission. En effet, la latence est non bornée et les trames peuvent être supprimées.

Ces classes de trafic ont différentes priorités. La priorité la plus haute est accordée pour les flux de synchronisation. Les flux TT viennent après. Le flux RC sont moins prioritaires que ceux du TT et les flux BE ont la priorité la plus faible. Dans le domaine avionique, la sous-classe à quota limité est assurée essentiellement par le protocole AFDX. Ce type de trafic assure également la compatibilité d'une fonctionnalité communicante conformément au protocole AFDX sur des réseaux TTEthernet. Nous étudions dans la section 2.5 l'intégration des différents flux de TTEthernet

2.5 Intégration du flux TTEthernet : Techniques et Porosité

Comme les flux TT et ET ne suivent pas le même principe de transfert et partagent la même architecture physique, il est nécessaire de définir une technique d'intégration qui détermine la manière de cohabitation des deux flux. Nous procédons dans la section 2.5.1 à la présentation des différents techniques d'intégration et nous étudions dans la section 2.5.2 un phénomène lié à cette cohabitation appelé *porosité*

2.5.1 Techniques d'intégration du flux TTEthernet

Le partage des canaux de communication par les différents flux de TTEthernet induit à des situations conflictuelles. Trois types de conflits peuvent se présenter. Dans le premier type, les trames conflictuelles ont la même priorité. Elles sont ainsi servies suivant la politique First In First Out (FIFO). Le deuxième type de conflit survient lorsque une trame de priorité haute H est servie et une autre de priorité moindre L devient prête. Dans ce cas, la trame avec la priorité faible est gardée dans une file. Le troisième cas survient lorsqu'une trame de priorité

faible est en cours de transmission et une autre plus prioritaire devient prête. Dans ce cas, trois mécanismes sont définis dans (Steiner et al., 2009) et qui peuvent être utilisée pour intégrer la trame la plus prioritaire avec la moins prioritaire. Nous illustrons ces différents techniques par la figure 2.7.

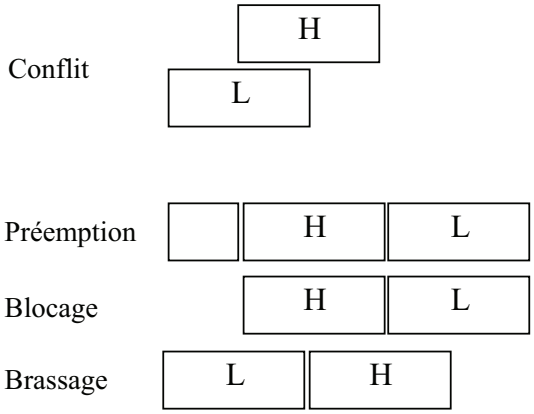


Figure 2.7 Technique d'intégration des trames TTEthernet

Préemption

Dans ce cas, la trame L est relayée par un routeur et une trame plus prioritaire H arrive. Le relai de la trame L est interrompu. Le routeur établit un temps de silence pour transmettre la trame H. Cette technique a les avantages et les inconvénients suivants :

- *Avantages :*
  - Une latence connue pour la trame H.
- *Inconvénients :*
  - La trame L est tronquée et ainsi nous avons la génération des faux messages.
  - Indétermination pour la transmission de la trame L.
  - Le besoin de retransmettre les faux messages.

Blocage

La trame prioritaire H est une trame TT. L'ordonnancement de cette trame est connu d'avance (il est complètement défini temporellement). Nous pouvons prévenir un éventuel

conflit en bloquant la transmission d'un message L si, nous ne sommes pas sûr que la transmission se termine avant le début de transmission du message H . Cette méthode a les avantages et les inconvénients suivants :

- *Avantages :*
  - L'intégration impose un délai fixe pour transmettre L.
  - Pas d'interruption de la transmission de la trame L.
  - L'ordonnancement de la trame H n'est pas retardé .
- *Inconvénients*
  - Avec le blocage, on bloque la transmission pour la durée d'envoi d'une trame L de taille maximale. Ceci implique une perte en termes de bande passante.

Cette technique garantie que la transmission de la trame L se termine avant l'envoi de la trame H.

Brassage

Dans ce cadre, si la trame L est relayée par une trame quand la trame H arrive alors, l'envoi de la trame H est retardé jusqu'à ce que le relai de la trame L se termine. Par conséquent, le brassage a les avantages et les inconvénients suivants :

- *Avantages :*
  - En terme d'utilisation de bande passante, le brassage est optimal tant que nous ne préemptons pas la trame L et nous ne bloquons pas le trafic pour une durée bien déterminée.
- *Inconvénients :*
  - Mauvaise qualité temps réel puisque la trame prioritaire H peut être retardée bien qu'elle est destinée pour un trafic critique.
  - Le brassage nécessite des contraintes supplémentaires pour l'ordonnanceur.
  - L'augmentation de la gigue et de la latence doivent être maitrisées dans ce cas.

Dans le cadre avionique, la méthode d'intégration utilisée est le blocage. Elle ne tronque pas les trames moins prioritaires comme la méthode de préemption et ne favorise pas une trame moins prioritaire sur une plus prioritaire comme la méthode de brassage.

Afin de véhiculer les deux types de trafics TT et RC sur le même support de communication, il faut s'assurer qu'il existe suffisamment de bande passante pour ces deux types de trafics. Malheureusement, cette étape est insuffisante pour assurer une bonne intégration. En effet,

si les trames TT sont ordonnancées collées les unes sur les autres, le trafic TT étant plus prioritaire, va bloquer le trafic RC. Pour une bonne intégration de ces deux types de flux, l'ordonnancement TT doit être assez éparpillé pour laisser des fenêtres de temps pour le flux RC. Cette propriété est connue sous le non de *porosité* (Zurawski, 2005).

2.5.2 Porosité

La figure 2.8 illustre la différence entre deux ordonnancements poreux et non poreux. Celui du haut est non poreux. Les trames TT sont collées les uns aux autres. Celui du bas est plutôt poreux.

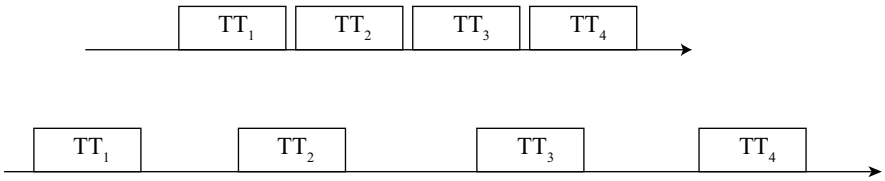


Figure 2.8 Porosité d'un ordonnancement TT

Trois types de techniques sont définies dans (Steiner, 2011) pour assurer la porosité. Avant ordonnancement, après ordonnancement et avec interprétation de l'ordonnancement. La technique avant ordonnancement prend en entrée (pour la recherche d'ordonnancement) comme exigences des intervalles d'espacement entre les trames consécutives. La technique après ordonnancement cherche plutôt à imposer l'espacement entre les trames après l'établissement de l'ordonnancement par un post-traitement. La dernière technique appelée interprétation de l'ordonnancement réserve des fenêtres de communication et assure que celles-ci sont assez larges pour englober à la fois le trafic TT et le trafic RC.

2.6 Ingénierie dirigée par les modèles

L'ingénierie dirigée par les modèles est une branche du génie logiciel ou les modèles sont considérés l'entité de base pour toutes les phases du processus du développement (Rutle et al., 2015). Les modèles sont utilisés ainsi pour spécifier les logiciels sous développement et pour effectuer diverses transformations automatiques.

À la différence des pratiques traditionnelles du génie logiciel où la modélisation est utilisée principalement pour la documentation et pour des fins de communication, l'ingénierie dirigée par les modèles se base sur le principe que tout est modèle (Wagner et al., 2011). Cette façon de voir permet ainsi de surpasser les limites de la technologie objet à lever le niveau d'abstraction et à traiter des problèmes complexes pour des systèmes évolutifs.

L'ingénierie dirigée par les modèles repose sur les concepts de base système, modèle et méta-modèle et la relation entre eux (Wagner et al., 2011) ainsi, un modèle représente un système et doit être conforme à un méta-modèle. Elle repose aussi sur le concept de transformations de modèles. Une transformation de modèle se définit comme étant la conversion d'un ou plusieurs modèles sources à un modèle destination ou tous les modèles doivent être conformes à un méta-modèle y compris la transformation considérée (Gaševic et al., 2009). Les transformations peuvent être classées en trois catégories : (1) modèle à modèle (2) modèle au système et (3) modèle du système au modèle.

Les applications de l'ingénierie dirigée par les modèles sont diverses. Dans (Czarnecki and Helsen, 2006), on a identifié les applications suivantes.

— La génération des modèles de bas niveau à partir du niveau plus haut.
— La génération d'artefacts de développement (fichiers de configuration, code source).
— Association et synchronisation des modèles.
— Créations des vues à partir des requêtes sur le système
— Réfactorisation des modèles.
— Rétro-ingénierie.
— Vérification.

2.7 Satisfiabilité Modulo Théories

La satisfiabilité modulo théories (Cimatti, 2008) est un formalisme bien établi pour résoudre les problèmes à satisfaction de contraintes. Plusieurs solveurs ont été définis dans ce cadre tel que Yices (Dutertre and De Moura, 2006) et Z3 (de Moura and Bjørner, 2008). Le principe de la satisfiabilité modulo théories consiste à évaluer des formules de premier ordre avec le respect de certaines théories de fond (arithmétique linéaire, vecteur à bits, etc.). Plus précisément, le concept consiste à avoir des solveurs à deux niveaux. Dans le niveau haut, on trouve un solveur SAT qui identifie la théorie de base de chaque clause et fait appel au solveur adéquat du deuxième niveau. Ce dernier a la capacité de résoudre une telle clause. L'un des éléments de succès de la satisfiabilité modulo théories est son fondement sur le célèbre algorithme de backtraking DPLL, qui par la suite adapté en DPLL(T) pour résoudre les formules moyennant une théorie de base. Ce formalisme a été adopté dans un large spectre de domaines tels que la vérification de modèles (Trindade and Cordeiro, 2016), l'ordonnancement (Steiner, 2010; Craciunas and Oliver, 2016), et la synthèse de programmes (Metzner et al., 2005). Les solveurs SMT sont devenus de plus en plus populaires et ont progressé d'une manière spectaculaire de point de vue performance dans les dernières années (Barrett et al., 2013).

Considérons la satisfiabilité modulo théories, nous notons que dans ce cadre nous sommes limité aux problèmes de satisfaction. La décision du solveur est par rapport à des propriétés qui peuvent être soit vraies ou fausses avec le retour d'une solution possible si ça existe.

Max-SMT généralise les problèmes SMT dans le cas où l'objectif n'est pas de satisfaire toutes les contraintes, mais d'en satisfaire le maximum (Bofill et al., 2008). Max-SMT est par la suite étendue pour la version pondérée : Max-SMT à pondération partielle (WPMS). Dans ce cadre, nous avons deux types de contraintes : *contraintes dures* qu'une solution valide doit satisfaire et *contraintes douces* qui peuvent être satisfaites ou non. Un poids est associé à chaque contrainte douce et compte comme coût de violation de cette contrainte. Le solveur retourne par la suite une solution qui minimise la somme des coûts des contraintes violés. Les problèmes Max-SMT et WPMS sont ainsi les versions d'optimisation de SMT.

2.8 Programmation par contraintes

Constraint Programming (CP) ou programmation par contraintes est un paradigme de programmation qui a fait son apparition vers la fin des années 70 (Mackworth, 1977) pour résoudre les problèmes combinatoires de grande taille. La CP (Rossi et al., 2006) de nos jours est devenu un formalisme très bien établi pour les problèmes d'optimisation de contraintes (Constraint Optimization Problem (COP)) et plus particulièrement les problèmes d'ordonnancement. Il repose sur la séparation entre la partie modélisation du problème et sa résolution. Ainsi, l'effort peut être focalisé séparément soit dans la modélisation ou dans la résolution et des algorithmes de résolution dédiés peuvent être définis séparément pour améliorer les performances.

Comme mentionné dans (Rossi et al., 2006), un de facteurs de succès de CP pour les problèmes d'ordonnancement est leurs relations étroites au domaine de Recherche Opérationelle (RO). RO offre des algorithmes efficaces qui contribuent aux performances élevées des programmes CP. Ces programmes CP peuvent contenir des heuristiques de résolution qui peuvent guider le solveur au moment de l'exécution à faire les bons choix de résolution.

Plus encore, et pour améliorer davantage ces performances, la CP permet l'exploitation de certaines contraintes dites globales qui résolvent d'une manière optimisée un problème récurrent. À titre d'exemple, généralement, les problèmes d'ordonnancement à ressources limitées peuvent bénéficier d'une formulation sous forme de problèmes de *bin packing* (Berkey and Wang, 1987) pour le résoudre d'une manière efficiente.

Un Problème de Satisfaction de Contraintes (CSP) est défini dans (Rossi et al., 2006) comme un triplet $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ où :

— $\mathcal{X} = \{x_1, ..., x_n\}$ est un ensemble fini de variables.

— $\mathcal{D} = D(x_1) \times ... \times D(x_n)$ est le domaine de $\mathcal{X}$. $D(x_i) \subset \mathbb{Z}$ est l'ensemble des valeurs que la variable x_i peut prendre.

— $\mathcal{C} = \{c_1, ..., c_e\}$ est un ensemble de contraintes. L'ensemble des variables considérées pour la contrainte c_j est noté $\mathcal{X}(c_j)$ avec $\mathcal{X}(c_j) \subset \mathcal{X}$. Chaque contrainte est une relation sur un ensemble de variables.

Une solution au problème CSP est une assignation des valeurs $d_i \in D(x_i)$ pour chaque x_i de telle sorte que l'ensemble des contraintes est satisfait. Si nous voulons optimiser un objectif $f : \mathcal{D} \rightarrow \mathcal{Z}$, nous parlons dans ce cadre plutôt d'un problème d'optimisation de contraintes COP modélisé par $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C}, f)$.

Un problème CSP ou COP noté P se résout en combinant ces deux opérations :

1. *Propagation de contraintes :* Un solveur de programmation par contraintes définit un algorithme de filtrage pour enlever quelques valeurs inconsistantes des variables. Le filtrage se fait jusqu'à ce qu'un point fixe est atteint (c.-à-d. pas de valeurs additionnelles qui peuvent être supprimés)
2. *Recherche de solution :* Les valeurs obtenues suite à la phase de propagation des contraintes sont consistantes. Néanmoins, considérant l'ensemble des contraintes et une assignation partielle des valeurs aux variables, certaines valeurs des autres variables peuvent conduire à une inconsistance. Ainsi, il est essentiel de vérifier la consistance des valeurs restantes. Pour se faire, les différents choix sont présentés par un arbre où chaque niveau représente une variable et les nœuds de ce niveau constituent les différentes valeurs possibles que nous pouvons assigner à cette variable. L'opération de recherche consiste ainsi à explorer cet espace pour déterminer une solution.

Pour améliorer les performances de recherche essentiellement quand il s'agit d'un problème d'optimisation, nous nous appuyions sur des *heuristiques de branchement* (Kiziltan et al., 2007) pour adopter une stratégie intelligente pour le choix des variables et des valeurs. Plus précisément, nous citons la stratégie de sélection des variables $first\_fail$ (Rossi et al., 2006) qui consiste à sélectionner la variable x_i avec la taille du domaine $|D(x_i)|$ minimal et *min_domain* (Rossi et al., 2006) comme stratégie de sélection des valeurs qui sélectionne la valeur minimale de ce domaine en premier lieu.

Les formalismes de SMT et de programmation par contraintes sont bien adaptés pour trouver des ordonnancements statiques. Pour analyser l'ordonnançabilité des flux dynamiques et plus précisément ceux de RC, d'autres méthodes analytiques sont adoptées que nous présenterons dans la section 2.9.

2.9 Analyse d'ordonnançabilité des flux RC : calcul réseau et approche par trajectoire

Nous débutons cette section par la présentation des principes de base de la méthode de calcul réseau.

2.9.1 Calcul réseau

Le calcul réseau (Cruz, 1991a,b; Le Boudec and Thiran, 2000) est un ensemble d'outils mathématiques utilisés pour analyser le trafic réseau. Parmi les propriétés qu'on peut analyser avec cette méthode, une borne supérieure pour la taille des files et la latence dans le réseau. Le calcul réseau est basé essentiellement sur le formalisme de l'algèbre min-plus (Cuninghame-Green, 1979) où, l'addition arithmétique est remplacée par la minimisation représentée par $\oplus$ et la multiplication arithmétique par l'addition représentée par $\otimes$.

Un flux dans le calcul réseau est représenté par une fonction cumulative R où, $R(t)$ indique le nombre total de bits envoyés jusqu'à l'instant t. Le calcul réseau utilise la notion de la courbe d'arrivée et la courbe de service pour analyser le trafic.

Courbe d'arrivée La courbe d'arrivée caractérise le flux en entrée à un nœud. Plus spécifiquement, elle constitue une enveloppe pour le trafic reçu par un nœud. On dit qu'une courbe d'arrivée α est une enveloppe pour le trafic τ_i, si pour chaque intervalle de temps $[t_1, t_2]$ le trafic donné par le flux τ_i est inférieur à $\alpha(t_2 - t_1)$. Formellement, si l'on note par $R(t)$ le trafic cumulatif pour le flux τ_i, ce dernier est contraint à une courbe d'arrivée α si et seulement si :

$$R(t + \delta) - R(t) \leq \alpha(\delta) \tag{2.3}$$

La figure 2.9 illustre un exemple de courbe d'arrivée. La courbe en haut représentée par la couleur bleue est une courbe d'arrivée pour le trafic représentée par la courbe en bas. Ceci peut être justifié par le fait que le taux de croissance de la fonction cumulative est toujours inférieur à α.

Pour illustrer la notion de courbe d'arrivée, on prend comme exemple le trafic AFDX. On rappelle ici que le flux de trafic AFDX est un flux avec limitation du quota d'utilisation de la bande passante. Un maximum d'une trame de longueur L_i^{max} peut être reçue par un nœud h dans le réseau chaque intervalle de temps de longueur bag_i. Le flux τ_i peut être ainsi contraint à la courbe d'arrivée dont son équation est donnée par 2.4 :

$$\alpha_h(\delta) = \frac{L_i^{max}}{bag_i} \times \delta + L_i^{max} \tag{2.4}$$

La courbe d'arrivée dans ce cadre est une fonction affine ayant une intersection avec l'axe des ordonnées au point d'ordonnée L_i^{max} (la quantité de données maximale reçue par le nœud h à $t = 0$). La pente de cette courbe est donnée par $\frac{L_i^{max}}{bag_i}$ (le quota maximal qui caractérise le flux τ_i)

Courbe de service Afin d'avoir une idée globale sur le flux au niveau de chaque nœud, en plus de la courbe d'arrivée, on doit avoir une idée sur le service minimal offert par ce nœud. Cette information est donnée par le deuxième type de courbe ou courbe de service. Formellement, si l'on note par R et R' le flux en entrée et le flux en sortie à un nœud h, on dit que h a une courbe de service β si et seulement si :

$$\forall t \geq 0,\ \exists t_0 \leq t \quad R'(t) - R(t_0) \geq \beta(t - t_0) \tag{2.5}$$

Ceci est équivalent à dire :

$$R' \geq \inf_{0 \leq s \leq t} \beta(t - s) + R(s) = R \otimes \beta \tag{2.6}$$

où R est le flux en entrée et R' est le flux en sortie de ce nœud.

Comme exemple, on prend aussi celui du flux de trafic AFDX utilisé pour illustrer la courbe de service. Le flux en entrée est contraint à un délai fixe de traitement par le routeur noté l. Après ce délai, le flux en sortie est fourni avec le débit maximal R. La courbe de service du nœud h est donnée ainsi par la fonction 2.7 illustrée par la figure 2.10 :

$$\beta_h(\delta) = R\lceil \delta - l \rceil^+ \tag{2.7}$$

où $\lceil \delta - l \rceil^+ = max\{0, \delta - l\}$

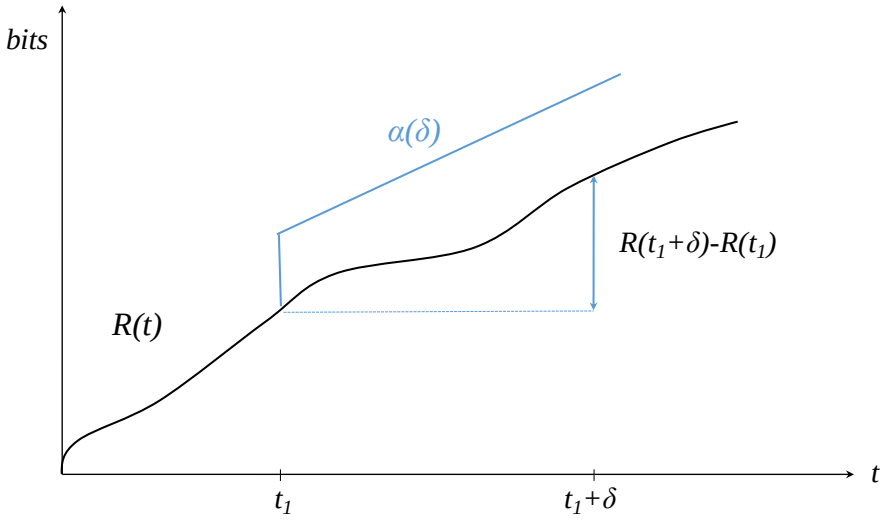


Figure 2.9 Exemple de courbe d'arrivée

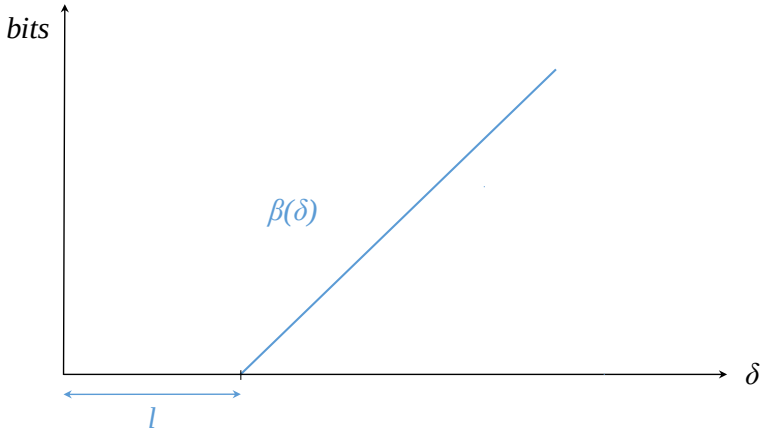


Figure 2.10 Exemple de courbe de service

Une fois qu'on a défini les deux courbes du calcul réseau (courbe d'arrivée et courbe de service), on peut analyser le flux de trafic.

Évaluation du trafic par calcul réseau La courbe d'arrivée et la courbe de service nous permettent d'évaluer trois bornes (Lauer, 2012) :

— L'occupation maximale

— Le délai maximal de traversée d'un nœud.

— La courbe d'arrivée en sortie du nœud.

La figure 2.11 illustre les deux premières bornes. Le pire cas de délai de traversée de nœud est donné par la déviation horizontale maximale d_{max} entre la courbe d'arrivée et la courbe de service. L'occupation maximale d'un nœud (backlog) est donnée plutôt par la déviation verticale maximale w_{max} entre les deux courbes.

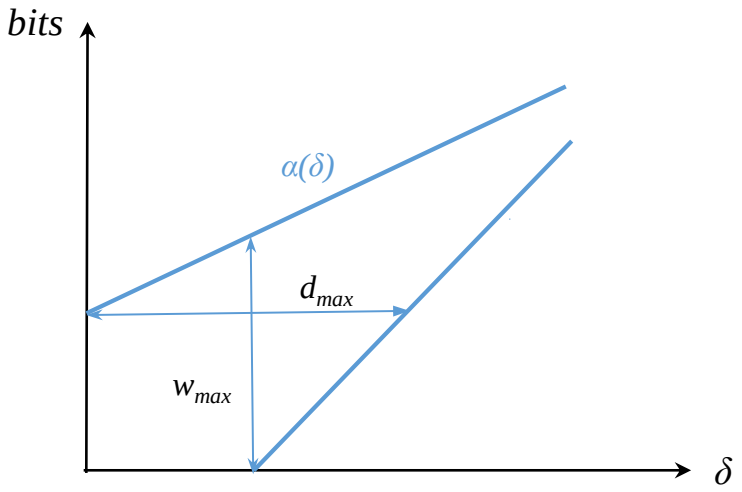


Figure 2.11 Occupation maximale et délai maximal au niveau de chaque nœud

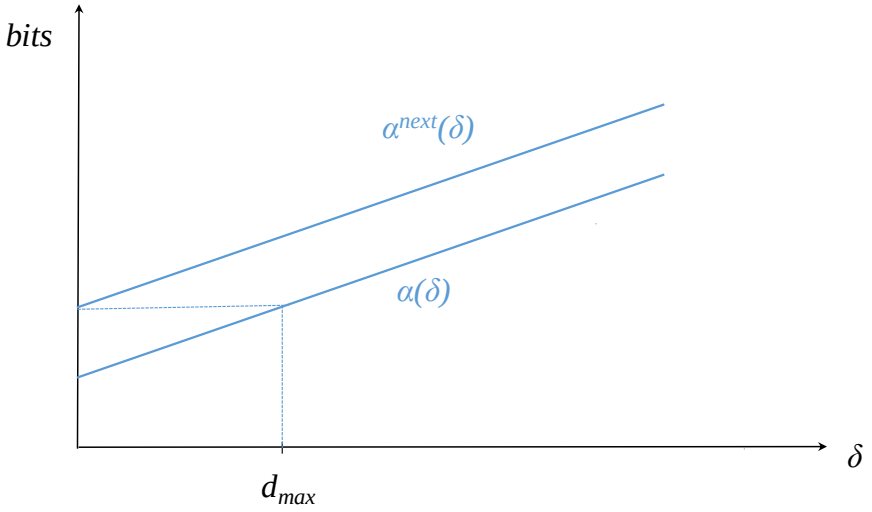


Figure 2.12 La courbe d'arrivée en sortie d'un nœud

La troisième borne qu'on peut déterminer par le calcul réseau est la courbe d'arrivée en sortie d'un nœud. Cette courbe sert à déterminer la courbe d'arrivée en entrée pour le prochain nœud du trafic. Cette dernière est déterminée en imposant un délai de d_{max} pour la courbe d'arrivée en entrée. La figure 2.12 illustre la courbe d'arrivée en sortie d'un nœud notée α^{next}.

L'approche par trajectoire suit une logique différente pour aborder le problème de vérification. Le principe général de cette approche est présenté dans la section suivante.

2.9.2 Approche par trajectoire

L'approche par trajectoire (Martin, 2004; Martin and Minet, 2006) a été développée à la base pour avoir des bornes supérieures pour les temps de réponse de bout en bout dans les systèmes distribués. Contrairement à la méthode de calcul réseau, l'approche par trajectoire analyse le pire cas d'un flux non au niveau de chaque nœud, mais plutôt dans son chemin complet.

Étant donné un flux τ_i et une trame f dans ce flux, l'approche par trajectoire consiste à identifier en premier lieu les trames qui peuvent inférer avec f. Une fois qu'on a identifié les trames, l'idée de cette approche et de chercher la dernière date d'envoi de la trame f dans le dernier nœud du flux τ_i. Illustrons cette approche à travers l'exemple de la figure 2.13.

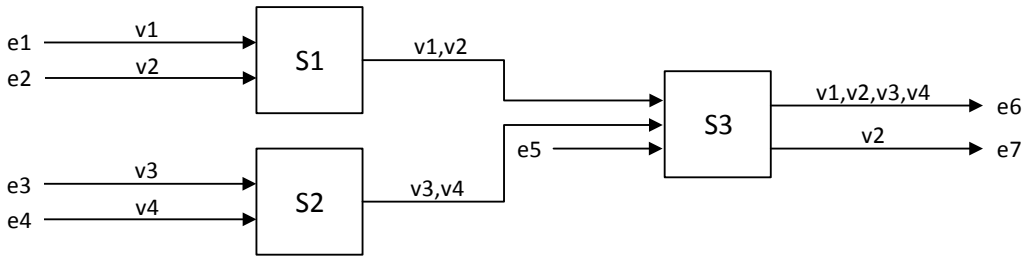


Figure 2.13 Exemple de l'approche par trajectoire

Le système présenté est composé de 7 nœuds terminaux (e1 à e7) et 3 routeurs (S1 à S3). Les nœuds (e1 à e5) envoient les trames sur les liens virtuels (vl1 à vl5). Les différents chemins d'un lien virtuel sont identifiés par les arcs étiquetés par le nom du lien virtuel. On note par τ_i le flux du lien virtuel vl_i.

On s'intéresse dans cet exemple particulier au délai de bout en bout du flux τ_3. On note par a_f^h le temps d'arrivée de la trame f au nœud h. La figure 2.14 illustre un exemple d'ordonnancement de la trame 3 (générée par e3) dans la trajectoire du flux τ_3.

Après que la trame 3 soit traitée par le nœud $e3$, elle arrive au nœud $S2$. La trame 4 arrive au niveau du nœud $S2$ avant la trame 4. Elle est ainsi traitée immédiatement. La trame 3 doit attendre jusqu'à ce que le port correspondant soit libéré. La trame 4 arrive ensuite au niveau du routeur $S3$ après les trames 1 et 5. La trame 3, ayant été traité après la $4^{\text{ème}}$, arrive la dernière au niveau du nœud $S3$.

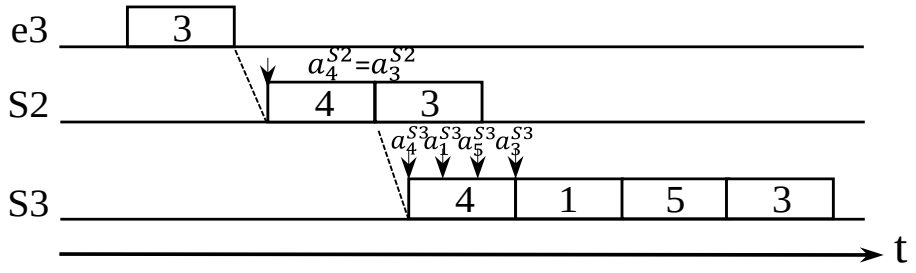


Figure 2.14 Un exemple d'ordonnancement dans la trajectoire du flux τ_3

Moyennant cette figure, on détermine R_3 le temps nécessaire pour la trame f pour traverser le flux τ_3. Il est défini comme suit

$$R_3 = W_3^{S3}(t) - t + C_3^{S3} \tag{2.8}$$

où t dénote le temps d'arrivée de f dans le réseau, $W_3^{S3}(t)$ le temps d'arrivée de f au niveau du nœud $S3$ et C_3^{S3} le temps nécessaire pour le nœud $S3$ pour transmettre la trame 3.

Pour le calcul de $W_3^{S3}(t)$ on procède comme suit :

— Au niveau du nœud $e3$, on compte C_3^{e3}.

— Au niveau du nœud $S2$ on compte plutôt, $C_4^{S2} - (a_3^{S2} - a_4^{S2})$.

— Quant au niveau $S3$, on compte $(C_1^{S3} + C_5^{S3} + C_4^{S3}) - (a_4^{S3} - a_1^{S3})$.

Ainsi on a :

$$W_3^{S3}(t) = C_3^{e3} + L + C_4^{S2} - (a_3^{S2} - a_4^{S2}) + L + (C_1^{S3} + C_5^{S3} + C_4^{S3}) - (a_4^{S3} - a_1^{S3})$$

Afin de maximiser le dernier temps de départ de la trame 3, l'approche par trajectoire considère que chaque trame venant d'un nœud précédent devrait être reportée à l'heure d'arrivée de la trame 3. Dans le cas de notre exemple ceci correspond à l'arrivée de la trame 3 en même temps que les trames 1 et 5 au niveau du nœud $S3$.

2.10 Conclusion

Nous avons présenté dans ce chapitre les notions préliminaires pour la compréhension de cette thèse. Nous avons introduit, dans la section 2.1, deux approches pour la conception des systèmes temps-réels distribués. Ce qui nous a permis dans la section 2.2 de présenter l'architecture déclenchée par le temps et plus spécifiquement l'architecture globale des systèmes avionique distribuées dans la section 2.3. Par la suite nous nous sommes intéressées dans la section 2.4 à l'étude des différents modes de communication dans le cadre de cette architecture et aux différents techniques assurant la cohabitation de ces flux dans la section 2.5. Après, nous avons introduit les différentes notions nécessaires pour la compréhension de nos approches. Nous avons présentés, dans la section 2.6, les concepts de base de l'ingénierie dirigée par les modèles. Plus loin, nous avons introduit les formalismes de résolution par contraintes soit la satisfiabilité modulo théories, dans la section 2.7, et les concepts de base de la programmation par contraintes dans la section 2.8. Vers la fin, nous avons présenté différentes techniques pour l'analyse de l'ordonnançabilité des flux RC.

CHAPITRE 3 REVUE DE LITTÉRATURE

Nous présentons dans ce chapitre un aperçu des différents travaux de recherches reliées à notre problématique de recherche. Le chapitre est organisé en quatre parties. Nous présentons en premier lieu les travaux relatifs aux systèmes IMA. Dans la deuxième partie, nous étudions les différents travaux relatifs à la vérification des flux asynchrones. Dans la troisième partie, nous nous intéressons plutôt aux techniques et travaux relatifs à la synthèse d'un ordonnancement synchrone. Nous réservons une dernière partie aux différents travaux visant l'étude et la vérification simultanée des flux synchrones et asynchrones.

3.1 Systèmes IMA

Nous étudions dans cette section la revue des différents travaux relatifs aux systèmes IMA que nous organisons en quatre parties. Nous réservons une première partie à l'ingénierie des systèmes IMA. Dans une deuxième partie, nous étudions différents travaux reliés à l'ordonnançabilité de ces systèmes. Nous étudions par la suite dans une troisième partie la sureté et la multicriticité de ces systèmes et nous réservons une dernière partie aux approches étudiant l'intégration incrémentale.

3.1.1 Ingénierie des systèmes IMA

La transition vers l'architecture IMA a fait l'objet de plusieurs recherches. Nous citons dans ce cadre (Watkins and Walter, 2007) qui a identifié les considérations à envisager pour passer de l'architecture fédérée à l'architecture IMA. Ce travail fait également le point sur les différences entre ces deux architectures et établit les avantages d'une telle transition. Ces avantages ont été aussi recensés dans (Andrillon, 2008) dont les plus importants sont la possibilité d'envisager plusieurs fonctions partagées par plusieurs systèmes et la possibilité de reconfiguration et la certification incrémentale.

Dans (Bieber et al., 2012), une étude comparative a été également faite entre l'architecture IMA et federée. Les auteurs discutent aussi les deux nouvelles catégories de défis liés à l'adoption de l'architecture IMA. La première est la capacité de reconfiguration tandis que la deuxième est reliée aux exigences sur le plan des performances en s'orientant vers le choix de processeurs multicœurs et les défis d'analyse d'ordonnançabilité qui sont liés avec. Les auteurs font aussi le point sur trois types de contraintes dont les systèmes avioniques doivent vérifier. Le premier type de contraintes est celui de sûreté. En effet, les systèmes avioniques

doivent fonctionner correctement dans tous les cas. Le deuxième type de contraintes est celui de dépendance. Ces dernières assurent que le système marche correctement même en mode dégradé survenu à la suite d'évènements d'échecs. La dernière catégorie de contraintes ou contraintes temps réels contrôle les délais de l'entrée des données jusqu'aux sorties du résultat. Vu que nous cherchons dans notre contexte à vérifier l'ordonnançabilité des systèmes lors de l'intégration, nous visons plutôt ce type de contraintes.

Les travaux relatifs aux architectures IMA touchent essentiellement trois grands volets : la modélisation, la simulation et l'analyse de l'ordonnançabilité de ces systèmes. Ces travaux dont nous ferons l'étude dans cette partie sont d'autant complémentaires les unes pour les autres. Alors que les travaux de modélisation servent à spécifier correctement un système avionique complexe, ils servent néanmoins comme base sur laquelle certaines propriétés peuvent être vérifiées d'une manière formelle. Parmi les propriétés importantes que nous cherchons à vérifier dans notre cadre l'ordonnançabilité de ces systèmes. Les travaux de simulation peuvent être complémentaires dans ce cadre dans le sens où ils permettent de donner une idée sur le comportement réel de ces systèmes.

Vu la complexité des systèmes avioniques et les exigences auxquelles ils doivent répondre, une bonne modélisation constitue un enjeu essentiel. Dans (Farcas et al., 2010), les auteurs présentent une vision pour une approche de conception basée sur les modèles pour les systèmes automobiles et avioniques. Les auteurs fournissent également des recommandations pour l'adoption d'une telle approche. Parmi les recommandations qu'ils fournissent, la considération des propriétés de sûreté et de fiabilité vue qu'elles sont mandatées par des lois. Ils recommandent également la définition des méthodes qui établissent le lien entre la spécification et les exigences et ceci aux différents niveaux d'abstraction. Les auteurs font également le point sur différentes études ciblant l'ingénierie à base de modèles des systèmes avioniques et constatent que ces travaux focalisent sur quatre axes : (1) le modèle d'exigences (2) les architectures (3) la technologie et l'implémentation et (4) les plateformes et les outils de support qui permettent entre autres de vérifier les contraintes temporelles.

De point de vue des exigences, (Paulitsch et al., 2008) fait la revue des différentes exigences non fonctionnelles des systèmes avioniques à savoir : la sécurité, la sûreté, la maintenance, disponibilité, intégrité et les aspects de performances temporelles. Cette liste exhaustive d'exigences permet de dresser une image claire des critères que pourrait avoir un outil pour analyser les besoins fonctionnels et non fonctionnels des systèmes avioniques. Dans le cadre de ce travail, nous nous intéressons aux aspects de performances temporelles vu que nous vérifions que les pires cas de transmission respectent les dates limites exigées.

Alors que divers travaux abordent l'intégration avec différentes spécialités, (Wang, 2012)

aborde l'intégration des systèmes avioniques d'une façon systémique. Dans ce cadre, ce travail discute les sujets essentiels relatifs à l'intégration qu'il faut les considérer à savoir : la classification de l'intégration, les plateformes et les mécanismes utilisés, les méthodes d'intégration, l'organisation de cette dernière et l'efficacité qu'elle présente pour les systèmes avioniques. Il reste que l'intégration dans des architectures IMA présente des défis qui sont étudiés dans (Garside and Pighetti, 2007). Ces défis consistent à définir correctement un système avionique et de diviser correctement les tâches et responsabilités entre l'intégrateur système, le fournisseur d'applications et le fournisseur de plateformes.

La modélisation est d'autant plus importante lors que nous parlons de co-ingénierie. C'est le cas de (Noll, 2014), qui propose une approche pour la co-ingénierie logiciel hardware pour les systèmes avioniques embarqués. Pour ce faire, les auteurs ont utilisé le langage AADL pour la spécification de leur architecture. Cette spécification sert par la suite comme point d'entrée pour leur outil de vérification de modèles qui permet de vérifier un ensemble de propriétés tel que l'exactitude, la sûreté, la dépendance et la performance. Les problèmes rencontrés par les plateformes de modélisation ont été investigués dans (Li and Xiong, 2010) où les auteurs fournissent un moyen pour évaluer les performances temps réels pour les composants logiciels et matériels dans le cadre de systèmes avioniques. Ils proposent trois modèles génériques : un modèle d'application qui décrit les fonctions avioniques, un modèle d'architecture pour décrire l'architecture matérielle et un modèle comportemental pour modéliser l'ordonnancement des tâches et la communication entre eux. Ce modèle générique fournit ainsi les lignes directives sur les lesquelles nous pouvons se reposer pour avoir une bonne modélisation. Nous récapitulons dans le tableau 3.1 les différents travaux liés à l'ingénierie de ces systèmes. Nous catégorisons dans la colonne *Type* le travail exposé dans différentes phases d'ingénierie des systèmes IMA. Les notations *Ana* (respectivement *Spé*, *Mod* et *Sim*) indiquent les types Analyse (respectivement Spécification, Modélisation et Simulation). La dernière colonne de ce tableau (*Focus*) indique l'objet principal de chaque travail.

Tableau 3.1 Travaux sur l'ingénierie des systèmes avioniques

Référence	Type				Focus
	Ana	Spé	Mod	Sim	
Watkins and Walter (2007)	✓				Considérations pour IMA
Andrillon (2008)	✓				Avantages de IMA
Bieber et al. (2012)	✓		✓		IMA vs fédérée
Farcas et al. (2010)	✓		✓		Vision modèle
Paulitsch et al. (2008)	✓	✓			Exigences non fonctionnelles
Wang (2012)	✓	✓			Intégration systémique
Garside and Pighetti (2007)	✓				Défis d'intégration des sys IMA
Noll (2014)	✓	✓	✓		Co-ingénierie logiciel hardware
Li and Xiong (2010)	✓		✓	✓	Directives de modélisation des systèmes avioniques

Nous nous intéressons dans la section suivante à l'analyse de l'ordonnançabilité de ces systèmes.

3.1.2 Ordonnançabilité des systèmes IMA

Les travaux qui portent sur l'analyse de l'ordonnançabilité des systèmes avioniques visent essentiellement deux types de problèmes. Le premier type sert à vérifier l'ordonnançabilité de chaque partition tandis que le deuxième cherche à trouver un ordonnancement pour chaque partition. Ces deux problèmes sont classés sous le type d'ordonnancement périodique et non préemptif (Al Sheikh et al., 2012) et dont la complexité est NP dur (Jeffay et al., 1991) et qu'il y a seulement une approximation linéaire pour la résolution (Baruah and Chakraborty, 2006). Dans (Lee et al., 1998), le problème d'ordonnançabilité des partitions est adressé. Ce travail fournit également un test d'ordonnançabilité pour les systèmes IMA ainsi que le théorème qui le prouve.

Dans (Lee et al., 2000a), les auteurs s'intéressent non seulement à l'ordonnancement des partitions, mais aussi à la communication. À cette fin, les auteurs présentent plusieurs algorithmes. Ils considèrent un ordonnancement IMA hiérarchique à deux niveaux et pour la communication ils considèrent une communication conforme au standard ARINC 659. Dans le premier niveau, ils s'intéressent à la planification des partitions et des serveurs de communication en garantissant une contrainte de distance minimale puis dans le deuxième niveau ils assurent l'ordonnançabilité des tâches et des messages. Les différents algorithmes développés ont permis par la suite à caractériser dans (Lee et al., 2000b) un outil d'analyse

d'ordonnancement pour ce type de systèmes.

Dans (Deroche et al., 2016), l'allocation des applications avioniques dans une architecture IMA a été étudié tout en considérant une communication suivant le protocole AFDX. Ce travail présente un algorithme qui permet de répondre à deux objectifs à la fois. Le premier consiste à étudier la faisabilité d'allocation des partitions dans les modules tandis que le deuxième se focalise plutôt sur l'analyse des pires cas des communications de bout en bout. Parmi les recherches qui ont traité ces deux défis, nous citons (Al Sheikh, 2011) qui ont considéré une approche en deux étapes. Ils ordonnancent en premier lieu les partitions dans les nœuds d'exécution puis ils font le routage des flux du système avionique. (Ekelin and Jonsson, 2002) considère aussi ces deux défis mais s'intéresse plutôt à la problématique d'allocation des partitions en minimisant le coût des communications.

Nous n'abordons pas le premier défi dans le cadre de notre problématique de recherche dans le sens où nous supposons que le plan d'allocations des partitions aux modules est donné par l'ingénieur système. Le deuxième défit ou analyse des pires cas de transmission fait partie de nos objectifs. En effet, nous cherchons à synthétiser un ordonnancement en exigeant des contraintes des délais de bout en bout sur le système.

Certains travaux se basent sur la théorie des jeux pour étudier l'ordonnançabilité des partitions IMA. Ces travaux se basent sur la définition du facteur d'évolution qui détermine de combien les fenêtres de temps des partitions peuvent être étendues. Nous citons dans ce cadre (Al Sheikh et al., 2012) qui détermine à partir d'un ensemble de partitions avioniques bien définies, leur allocation sur l'architecture IMA et le meilleur partage des ressources entre les partitions. Ce partage prend en compte le besoin applicatif entre les applications en termes de périodicité stricte, de durée, de capacité mémoire et également en termes de ségrégation (ou de redondance) des partitions et de colocalisation de différentes partitions dans un même module ou non. La meilleure solution dans ce cadre est par rapport au facteur d'évolution qui sera maximisé pour chaque partition. Le travail de (Chen and Du, 2015) utilise aussi la théorie des jeux pour décider sur l'ordonnançabilité des partitions dans le cadre d'un système IMA. Il cherche également un ordonnancement pour les différentes partitions. L'approche proposée dans ce cadre détermine le facteur d'évolution maximal pour chaque partition qui sert comme paramètre de décision de l'ordonnançabilité.

Le travail (Chen and Du, 2015) a été étendu par la suite dans (Chen et al., 2016) pour aborder le problème d'odonnançabilité des partitions dans une architecture multiprocesseurs. Les auteurs proposent une formulation en Programmation Linéaire Mixte (MILP) pour calculer le facteur d'évolution et déterminer ainsi l'ordonnançabilité des partitions. Les auteurs conduisent en plus des simulations pour comparer leur approche qui est basée sur la théorie

des jeux par rapport à une résolution exacte en suivant la représentation MILP. La comparaison est faite par rapport à deux critères qui sont le temps consommé et les facteurs d'évolution déterminés et montrent que l'analogie de théorie des jeux est une heuristique efficiente pour résoudre le problème d'ordonnancement dans le cadre d'architecture multiprocesseurs.

D'autres critères peuvent être considérés dans la synthèse d'un ordonnancement IMA. C'est le cas pour (Si et al., 2015) qui propose une approche pour assister l'intégrateur système à valider l'ordonnançabilité des applications IMA suivant trois critères : (1) le temps d'exécution des partitions (2) leurs fréquences et (2) leurs ordres respectifs dans la MAjor Frame (MAF). Ces critères permettent ainsi de valider l'ordonnancement des partitions tout en supposant qu'ils sont déjà bien formés. L'approche développée dans ce cadre s'intéresse non seulement à la validation de l'ordonnançabilité, mais aussi au processus de modélisation, d'analyse de modèle. Concernant la modélisation, les auteurs ont présenté une extension AADL dont le choix du langage a été justifié par capacité à modéliser rigoureusement les systèmes à sûreté critique.

Une classification des travaux sur l'ordonnancement des tâches périodiques dans le cadre d'une architecture IMA a été faite dans (Chen et al., 2016). Trois catégories de travaux ont été présentées dans ce cadre. La première catégorie s'adresse au problème de l'analyse de l'ordonnacabilité des tâches strictement périodiques et qui répondent à des contraintes spécifiques. La deuxième catégorie des travaux est basée sur la détermination du facteur d'évolution critique et par la suite de décider si toutes les partitions sont ordonnançables ou non sur un nombre fini de processeurs et la troisième catégorie qui se base sur le temps de calcul maximal autorisé pour chaque tâche pour vérifier l'ordonnacabilité.

D'autres objectifs peuvent être considérés dans la synthèse d'un ordonnancement IMA. Nous citons dans ce cadre (Lhachemi et al., 2016) qui s'adresse au problème de modélisation et d'optimisation des ordonnancements pour les systèmes IMA multicritiques. L'approche développée permet de déterminer et d'optimiser des configurations qui sont caractérisées par (1) la distribution des tâches entre les partitions ainsi que (2) les dates de début et de fin des partitions qui garantissent l'ordonnançabilité totale du système. Les auteurs peuvent considérer un de ces deux objectifs pour leur problème d'optimisation. Le premier objectif consiste à minimiser le temps alloué aux partitions alors que le deuxième minimise le temps de réponse. Cette approche peut être appliquée dans plusieurs contextes tels que la migration des systèmes existants vers l'architecture IMA, la modification et la mise à niveau des fonctions déjà existantes et la conception de nouveaux systèmes avioniques IMA.

Afin d'améliorer les temps de réponse des tâches dans le cadre d'un système IMA, (Tao et al., 2015) propose une approche qui permet de redéfinir l'ordonnancement IMA en effectuant deux

types d'opérations. Ces deux types sont le réajustement qui consiste à redéfinir la longueur de chaque partition et le fusionnement qui consiste à combiner les tâches de deux partitions pour former une nouvelle. Dans le cadre de notre travail, nous ne cherchons pas à redéfinir les partitions mais plutôt à trouver un ordonnancement pour ces derniers qui minimise le coût de reconfiguration de tout le système.

Nous résumons les différents travaux relatifs à l'analyse d'ordonnaçabilité dans le tableau 3.2. La première colonne de ce tableau indique si le travail présenté cherche un plan allocation des partitions IMA aux modules. La deuxième colonne indique si le travail cherche un ordonnancement des partitions et la troisième si l'ordonnançabilité des tâches dans une partition est considéré La quatrième colonne indique si une communication est considérée dans l'orodonnançabilité de ces systèmes et éventuellement son type. La dernière colonne indique si le travail présenté considère un critère d'optimisation.

Dans le cadre de notre problématique de recherche, nous ne considérons pas l'allocation des partitions aux modules ni l'ordonnançabilité des tâches au sein d'une partition. Nous supposons que les partitions sont déjà bien formées et le plan d'allocation est connu d'avance et nous cherchons plutôt à synthétiser un ordonnancement pour les différentes partitions. Concernant la partie communication, nous nous différons par rapport aux travaux présentés dans le tableau 3.2 en considérant une communication TTEthernet dont nous analysons aussi l'ordonnançabilité. Le critère que nous optimisions est aussi bien différent des critères considérés dans ce tableau. En effet, nous nous plaçons dans le cadre de systèmes évolutifs dans lesquels nous cherchons à analyser d'une manière incrémentale l'ordonnançabilité. L'objectif ainsi est de chercher à chaque étape d'intégration un ordonnancement qui minimise la reconfiguration du système intégré. Ce critère conserne aussi bien l'ordonnancement IMA que celui du réseau.

Tableau 3.2 Travaux sur l'analyse d'ordonnançabilité IMA

Référence	Allocation	Ordonnancement		Communication	Optimisation
		Partition	Tache		
Lee et al. (1998)	×	✓	✓	×	×
Lee et al. (2000a,b)	×	✓	✓	ARINC 659	×
Deroche et al. (2016)	✓	✓	×	AFDX	×
Al Sheikh (2011)	✓	✓	✓	AFDX	Temps Réponse
Ekelin and Jonsson (2002)	×	✓	✓	Bus	Coût Comm
Al Sheikh et al. (2012)	✓	✓	✓	×	Temps Réponse
Chen and Du (2015)	✓	✓	×	×	Temps Réponse
Chen et al. (2016)	✓	✓	×	×	Temps Réponse
Si et al. (2015)	×	✓	×	×	×
Lhachemi et al. (2016)	✓	✓	✓	×	Plusieurs
Tao et al. (2015)	×	✓	✓	×	×

Étroitement lié à l'ordonnançabilité, l'étude de la sureté et la multi-criticité des systèmes avioniques constitue une partie importante des travaux de recherches sur les systèmes IMA. Nous nous intéressons dans la section suivante à l'étude de la sureté et la multi-criticité de ses systèmes.

3.1.3 Sureté et multi-criticité des systèmes IMA

Les risques de point de vue ordonnancement et allocation de ressources a été étudié dans (Ren et al., 2015). Ce travail se base sur une spécification formelle de l'architecture qui permet par la suite la représentation de dépendance du système modélisé sous forme de réseau de corrélation. Ce réseau permet par la suite de dresser une stratégie d'évaluation du risque. Dans (Zheng et al., 2015), les auteurs traitent le problème de sûreté non de point de vue d'allocation, mais plutôt de point de vue des processus d'intégration des ressources. Pour ce faire, les auteurs proposent un modèle d'interactions de données entre les fonctions qui permet d'évaluer les risques de l'utilisation des fonctions par plusieurs fonctionnalités. La sûreté peut être aussi traité en se basant sur un modèle formel de propagation des fautes telles l'exemple de (Sagaspe, 2007) qui propose une approche pour décider si un ensemble de systèmes peuvent être implémentés sur une architecture IMA ou non tout en assurant des exigences de sûreté.

La sûreté des systèmes n'est pas le seul aspect qui est étudié dans les configurations des systèmes IMA. La multi-criticité constitue un autre aspect qui est fortement connexe à la sûreté et qui a été étudié par la communauté scientifique. Dans (Vestal, 2007), les auteurs

proposent une méthode pour analyser l'ordonnacabilité des tâches ayant différents niveaux de criticité et exigeant différents niveau de sûreté. Dans ce cadre, les auteurs assument qu'une tâche peut avoir différents pire-cas de temps d'exécution suivant le niveau de sûreté exigé et l'analyse de l'ordonnacabilité se base sur la modification de temps de réponse pour traiter la multi-criticité des tâches.

Un modèle pour l'ordonnancement des systèmes multi-critiques a été proposé dans (Baruah et al., 2012). Ce modèle définit aussi la faisabilité d'un scénario d'ordonnancement en considérant le temps d'exécution de chaque tâche suivant un niveau de criticité donné. Ils définissent aussi l'exactitude d'une instance d'un ordonnancement en vérifiant que tous les scénarios d'ordonnancement de cette instance sont corrects. Dans ce cadre, les auteurs supposent un seul processeur qui exécute toutes les tâches. La différence en termes d'exigences de certification est exprimée dans ce cadre par différents temps d'exécution du code. Ce temps dépend du niveau de criticalité exigé. En d'autres termes, le temps d'exécution est non décroissant avec le niveau de criticalité. Les auteurs prouvent également que le problème d'ordonnançabilité multicritique est NP-dur.

Un des problèmes qui peuvent subvenir avec les systèmes multicritiques est l'inversion de criticité (Sha et al., 1990) qui provient lors que des tâches moins critiques ratent leur date limite et infèrent par la suite sur les tâches les plus critiques. C'est dans ce cadre que la politique d'assignation des priorités suivant le niveau de criticité a été définie pour tel phénomène. Cette politique a l'inconvénient d'avoir des taux faibles d'utilisation des processeurs. Pour résoudre ce problème, (d. Niz et al., 2009) propose une autre politique qui définit un intervalle de temps pour chaque tâche durant lequel une tâche ne peut pas être interrompue par une autre moins critique.

La multicriticité peut être liée aussi aux tolérances aux fautes. En effet, en mode dégradé les performances du système sont réduites et il est utile d'assurer l'ordonnançabilité des tâches critiques même en mode dégradé. Dans cette perspective, (Guo and Baruah, 2014) définit un modèle pour lequel les processeurs exécutent les tâches avec une vitesse unitaire, plus rapide ou plus lente, mais pas en dessous d'un seuil inférieur. L'objectif est alors d'assurer que toutes les tâches dans des conditions normales respectent leurs dates limites et que toutes les tâches critiques respectent leurs date limites en mode dégradé.

Dans le cadre de cette thèse, nous supposons que les partitions sont bien définies. La multicriticité n'intervient que lors que nous analysons l'ordonnançabilité des tâches au sein d'une partition. La non-interférence entre les partitions de différentes criticités est garentie par le partitionnement spatial et temporel. Ainsi, nous n'abordons pas la multi-criticité des partitions IMA.

Autre que la synthèse d'ordonnancement et l'analyse d'ordonnançabilité, la reconfiguration des systèmes multicritiques est l'un des sujets qui ont été étudiés par la communauté scientifique. Nous citons dans ce cadre (Waez et al., 2014) qui propose une méthode pour la reconfiguration des systèmes multicœurs multicritiques. Cette méthode se base sur la représentation en automates et la synthèse de contrôleur qui permet de trouver la configuration du système tel que la fonctionnalité souhaitée est assurée même en cas d'échec

Dans notre cas, nous étudions la reconfiguration des systèmes avioniques évolutifs. Nous n'abordons pas la sureté ce ses systèmes vu que nous supposons un plan d'allocation fixe des partitions aux modules. Nous nous intéressons en plus à la minimisation des coûts qui sont associés à l'intégration itérative de ce système. Nous étudions dans la section suivante l'intégration incrémentale de ces systèmes.

3.1.4 Intégration Incrémentale des systèmes IMA

D'autres travaux s'intéressent à l'aspect évolutif de ces systèmes avioniques. C'est le cas de (Nam et al., 2014), qui définit une approche incrémentale dans le cadre d'intégration des systèmes IMA tout en assurant les contraintes temporelles des parties déjà existantes ainsi que des nouvelles. Les auteurs facilitent cette vérification en introduisant un nouveau commutateur de données qui permet de limiter la latence tout au long du réseau. Ce commutateur exécute un algorithme de commutation par horloge qui permet de donner les meilleurs pires cas de transmission pour tous les trafics faisables. Une heuristique a été associée pour déterminer si les contraintes réseaux d'un système IMA qui utilise un commutateur donné sont vérifiées ou non même si de nouvelles fonctionnalités ont été ajoutées et que ces dernières causent une augmentation de trafic. Lors que les contraintes temporelles ne sont pas vérifiées, l'approche développée détermine une nouvelle configuration du système. Dans le contexte de notre approche, nous n'introduisons pas un nouveau type de commutateur qui permet de borner les latences de plus, nous cherchons aussi à modéliser le coût de configuration et le minimiser.

Le travail présenté dans (Lauer et al., 2013), constitue une étude préliminaire à l'intégration itérative pour notre projet. Néanmoins, ce travail se limite à l'intégration d'une seule partition et ne modélise pas le trafic réseau sur les différents liens. Cette limite est due à l'utilisation de formalisme Programmation en nombre entier binaire (BIP) qui n'est pas capable de résoudre le cas le plus général.

Après l'étude des différents travaux liés aux systèmes IMA, nous étudions dans la prochaine section les différents travaux dédiés à l'analyse des réseaux à bande passante limitée.

3.2 Analyse des réseaux à bande passante limitée

Plusieurs méthodes ont été introduites pour vérifier les délais de bout en bout dans le cadre des réseaux à bande passante limitée. Dans (Charara et al., 2006), trois de ces méthodes ont été introduites. J'étends ce travail pour en présenter d'autres.

1. *Vérification de modèles (Alur and Dill, 1994)* : cette méthode détermine un pire cas de délais de bout en bout et le scénario correspondant vu qu'il explore tous les états possibles du système. Des techniques d'abstraction et d'agrégation ont été étudiés pour optimiser les performances. Les modèles utilisent essentiellement deux types d'automates. Un premier type pour présenter les évènements qui occurrent au niveau de chaque commutateur et un deuxième type pour représenter les évènements d'envoi. Parmi les travaux qui réfèrent à cette technique, nous citons (Ermont et al., 2006).
2. *Approche holistique* : c'est la première approche défini pour donner des bornes des délais de bout en bout. Introduite dans (Tindell and Clark, 1994), elle considère le pire scénario sur chaque nœud visité par un flux en prenant compte de la gigue maximale introduite par les autres nœuds visités. Cette approche n'est pas pratique vu qu'elle est trop pessimiste.
3. *Simulation des réseaux* : cette méthode donne des bornes expérimentales qui peuvent être dépassés. Ce type d'approche donne une estimation globale de la charge du réseau. Nous citons dans ce cadre le travail (Charara and Fraboul, 2005)
4. *Calcul réseau* : c'est une méthode analytique qui se base essentiellement sur l'analyse des flux entrant et sortant d'un nœud pour déterminer soit un seuil pour le retard maximal que peut avoir une trame au niveau de ce nœud ou un seuil minimal pour la taille requise pour une file. Cette méthode est pessimiste vu qu'elle repose sur des hypothèses pessimistes.
5. *Approche par trajectoire* : c'est une autre méthode analytique qui analyse le flux tout au long de sa trajectoire pour déterminer un seuil supérieur du pire cas de sa transmission. Le principe est de retarder la trame en question au maximum au niveau de chaque nœud. Ceci provoque un pessimisme.

Vu que nous cherchons dans notre contexte à vérifier que les délais de bout en bout ne dépassent pas un seuil fixé, nous écartons la méthode de simulation. La méthode de vérification de modèle ne peut pas être utilisée pour des exemples de taille industrielle vu l'explosion combinatoire qu'elle présente. Quant à la méthode holistique trop pessimiste, elle est impratique. Les méthodes de calcul réseau et l'approche par trajectoire présentent un bon compromis entre pessimisme et la maitrise de complexité de l'analyse. Ces deux méthodes sont les plus

utilisés et étudiés par la communauté scientifique. Nous consacrerons le reste de cette section à la présentation des travaux relatifs à ces deux méthodes. Nous catégorisons les travaux relatifs à l'analyse et vérification des flux à bande passante limités en deux catégories. Le classement est fait par la méthode utilisée et nous commençons notre étude par les travaux qui se base sur le calcul réseau.

Calcul réseau

Le travail présenté dans (Boyer and Fraboul, 2008), est l'un des travaux de référence qui utilise la méthode de calcul réseau pour l'analyse des délais de bout en bout dans le cadre des réseaux à bande passante limitée. Il présente de nouveaux résultats en tenant compte de l'assomption d'un trafic FIFO. La contribution formelle de ce travail est de prendre en compte l'algorithme du sceau percé dans l'analyse des pires cas de latence Ce travail combine aussi la technique d'agrégation de flux présenté dans (Grieu, 2004) avec la considération de l'algorithme de sceau percé dans l'analyse des pires cas de transmission Ceci a permis d'améliorer les bornes de latence.

Ces bornes ont été améliorées dans (Liu et al., 2012) en proposant un nouveau modèle de calcul réseau pour les réseaux AFDX. Ce travail étudie également les limites de l'ancien modèle de point de vue gestion de trafic au niveau de chaque commutateur et un nouveau paramètre compte a été défini pour exprimer le crédit que peux avoir un flux donné a un instant donné.

Dans (Zhang and Wang, 2012), afin de réduire le pessimisme les auteurs proposent une version modifiée du calcul réseau qui est basé sur la définition d'un nouveau modèle de service. Ce modèle de service fait la séparation du trafic prioritaire au trafic moins prioritaire pour la détermination des tailles requises des files d'attente.

Autre que les délais de bout en bout, le calcul réseau permet aussi d'étudier les tailles requises des files d'attente au niveau de chaque commutateur. C'est le cas de (Wu et al., 2011) qui étudie les tailles des files avec différentes priorités au niveau de chaque nœud d'un réseau AFDX. L'objectif dans ce cadre est d'avoir les meilleures tailles requises pour ces files à l'étape de conception. L'optimalité dans ce cadre réside dans le choix de l'espace minimal requis qui empêche la perte de données.

Dans (Frances et al., 2006), des résultats quantitatifs obtenus lors de l'optimisation des priorités des flux de trafic AFDX ont été présentés. Ces résultats montrent qu'avec une bonne affectation des priorités, les seuils de latence et des tailles des files peuvent être plus serrés. Ces seuils sont déterminés par l'application de la méthode de calcul réseau. L'étude

montre aussi qu'une assignation aléatoire des priorités donne des seuils moins bons que de ne pas en utiliser. Pour le choix des priorités optimales, les auteurs utilisent les techniques classiques d'optimisation comme la méthode descendante et l'approche d'élagage alphabeta pour donner de meilleures solutions. Pour expérimenter leurs résultats, les auteurs ont généré différentes configurations de priorités et ils ont remarqué que les seuils sont plus serrés de 40% pour les latence et de 30% pour les tailles des files.

Autre que le calcul réseau classique, le calcul réseau stochastique (Jiang, 2006) a été utilisé dans (Ridouard et al., 2007) pour estimer la distribution des latences dans un réseau AFDX et ainsi, évaluer le pessimisme par rapport à la borne supérieure de latence trouvée par la méthode du calcul réseau déterministe.

La méthode de calcul réseau est utilisée pour certifier les délais de bout en bout et les tailles requise des files. Dans (Bauer et al., 2012), on affirme que le dimensionnement des files d'attentes au niveau de l'A380 est fait par la méthode de calcul réseau. Le tableau 3.3 résume les différentes contributions faites par rapport à la méthode de calcul réseau dans l'ordre chronologique.

Tableau 3.3 Travaux relatifs à la méthode de calcul réseau

Cruz (1991a,b)	Calcul réseau
Le Boudec and Thiran (2000)	
Grieu (2004)	Techniques d'agrégation des flux
Frances et al. (2006)	Amélioration des performances par assignation des priorités
Jiang (2006)	Calcul réseau stochastique
Ridouard et al. (2007)	Estimation des distributions des latences
Boyer and Fraboul (2008)	Analyse des délais de bout en bout
Wu et al. (2011)	Taille des files avec niveaux de priorités
Liu et al. (2012)	Nouveau modèle avec crédit des flux
Zhang and Wang (2012)	Nouveau modèle de service avec priorités

Nous étudions dans le section suivante les différents travaux relatifs aux vérifications des flux avec quota limité en utilisant l'approche par trajectoire.

Approche par trajectoire

L'approche par trajectoire a été proposée dans (Martin, 2004). Le travail publié par le même auteur dans (Martin and Minet, 2006) constitue la référence pour l'analyse des pires cas de temps de réponse par l'approche par trajectoire. La méthode développée dans ce cadre détermine une borne pour ces pires cas en supposant une politique de transmission FIFO.

Dans le but d'améliorer davantage les bornes déterminées suivant cette approche, (Bauer et al., 2009) montrent comment la technique de groupage des flux définis dans (Grieu, 2004) dans le cadre de la méthode de calcul réseau peut être aussi exploité. L'adaptation de cette technique par l'approche par trajectoire montre une amélioration de 10% par rapport à la méthode de calcul réseau en utilisant la même technique de groupage.

Autre que les délais de bout en bout, les tailles requises pour les files peuvent être aussi étudiés par l'approche par trajectoire. Nous citons dans ce cadre (Bauer et al., 2012). La méthode développée permet un gain de 10% au niveau des tailles des files comparées à la méthode de calcul réseau. La détermination des tailles requises dans ce cadre est basée sur l'idée sur l'évaluation du pire retard que peut avoir une trame au niveau d'un nœud h. Pour résoudre ce problème, les auteurs considèrent en premier lieu les trames compétitives à une trame au niveau du nœud h puis cherchent l'ordonnancement de ces trames qui conduit au pire cas de retard de cette trame.

Plus récemment, on a découvert dans (Kemayo et al., 2013a) que la technique de groupage présenté dans (Bauer et al., 2009) conduit à un optimisme qui sous-estime les délais de bout en bout. Un contre-exemple a été défini par cette occasion pour illustrer l'optimisme. Dans (Kemayo et al., 2013b), la cause de l'optimisme a été identifiée. En effet, chaque paquet doit être compté une seule fois sauf le paquet pivot qui doit être compté deux fois vu qu'il est présent dans deux périodes d'occupation consécutives sauf que les auteurs affirment qu'ils ne disposent pas de moyen pour identifier le paquet pivot. Les auteurs détaillent aussi deux conditions couplées ensemble conduisent à une sous-estimation des délais de bout en bout.

La source de l'optimisme a été identifiée aussi dans (Li et al., 2014). Ce travail caractérise cet optimisme à travers un exemple détaillé avec les analyses. À la différence des travaux précédents, ce travail propose une solution pour corriger ce problème. Une formalisation de la correction dans le cas général a été fournie pour ce fait. L'analyse de l'origine de l'optimisme introduit par l'approche se résume par le fait que c'est vrai que la technique de groupage identifie la charge générée par des trames compétitives et qui ne retardent pas une trame donnée f_i. Néanmoins, une partie de ces trames compétitives ne sont pas prises en compte dans le calcul de retard par l'approche classique. Donc, une solution de résoudre le problème d'optimisme est d'exclure la charge de travail qui n'est pas considéré par l'approche classique considérant la technique de groupage.

Une autre proposition qui prend en compte la technique de groupage tout en évitant l'optimisme dans le cadre de l'approche par trajectoire a été introduite dans (Zhao et al., 2014) Une preuve pour l'exactitude du calcul proposé a été établie par la même occasion. Ces résultats ont été confirmés à travers des essais sur des configurations réelles de systèmes avioniques.

Les résultats montrent aussi que les bornes obtenues par l'approche par trajectoire sont en moyenne plus serrées que celles obtenues par la méthode de calcul réseau. Une évaluation du pessimisme donné par les deux approches a montré qu'il est deux fois moins élevé par l'approche par trajectoire que celui donné par la méthode du calcul réseau. Le tableau 3.4 résume les différentes contributions faites par rapport à l'approche par trajectoire dans l'ordre chronologique.

Tableau 3.4 Travaux relatifs à l'approche par trajectoire

Martin (2004)	Approche par trajectoire
Martin and Minet (2006)	Analyse des pires cas de temps de réponse
Bauer et al. (2009)	Adoption de la technique de groupage
Bauer et al. (2012)	Dimensionnement des files
Kemayo et al. (2013a)	Identification de l'optimisme du groupage
Kemayo et al. (2013b)	Identification de la cause de l'optimisme
Li et al. (2014)	Proposition d'une solution à l'optimisme
Zhao et al. (2014)	Adaptation de la technique du groupage avec preuve du non optimisme

Pour vérifier l'ordonnançabilté des flux RC, nous définissons une nouvelle méthode basée sur le formalisme de programmation par contrainte et qui cherchera à supprimer le pessimisme introduit par ces deux approches analytiques. Nous étudions dans la prochaine section les différents techniques et travaux définis pour la synthèse d'ordonnancements TT.

3.3 Synthèse et Vérification des Ordonnancement TT

Nous commençons notre étude par une revue des différents techniques utilisés pour la synthèse d'ordonnancement TT.

3.3.1 Techniques utilisés

Les formalismes utilisés pour établir un ordonnancement TT sont divers. On étend les techniques repérées dans (Amerion and Ektesabi, 2012) pour en déduire 6 catégories.

— *Programmation linéaire en nombre entier et Programmation linéaire mixte* (Schrijver, 1986) : On cite comme exemple dans ce cadre les travaux (Schmidt and Schmidt, 2010) et (Zeng et al., 2011). Cette approche est utilisée lors que les contraintes du problème sont linéaires. Son succès provient de l'utilisation de l'algorithme SIMPLEXE qui a prouvé ses performances dans la résolution de ces types de contraintes ainsi que

la possibilité d'optimiser une fonction objectif tel que la minimisation des délais de transmission.

— *Algorithmes génétiques* (Man et al., 2012) : Parmi les travaux qui définissent des algorithmes génétiques pour la recherche d'un ordonnancement, on cite (Ding et al., 2005) et (Ding et al., 2010). L'effort pour l'adoption d'une telle approche consiste d'une part à coder un ordonnancement possible sous forme de gêne appropriée et d'une autre part d'en définir dessus les opérations de sélection, de croisement et de mutation appropriées qui permettent la génération et l'optimisation des ordonnancements tout en considérant un ensemble de contraintes.

— *Recherche Tabou* (Luke, 2013) : On cite dans ce cadre le travail de (Tamas-Selicean et al., 2012a) et de (Steiner, 2011). Commençant d'un ordonnancement initial faisable. Cette méthode permet d'améliorer progressivement l'ordonnancement de départ suivant l'évaluation de l'objectif fixé et sous un ensemble de contraintes. Cette méthode permet également l'évitement des optimums locaux via le mécanisme de la liste tabou. L'adoption d'une telle méta-heuristique est utile lors que le problème à résoudre est dur et on peut accepter une solution sous-optimale pourvu que ça soit dans un temps raisonnable.

— *Recuit Simulé* (Luke, 2013) : Inspiré de la thermodynamique et plus précisément du processus de réchauffement et de refroidissement en metallurgie, l'idée de cette approche est d'exprimer la fonction objectif sous forme d'énergie qu'on essaie d'optimiser. On cite comme exemple de travail utilisant cette approche (Gavrilu et al., 2015). Ce dernier cherche pour des applications communicantes sur des réseaux TTEthernet la meilleure topologie possible qui soit tolérante aux fautes et dont son coût soit minimal.

— *Satisfiabilité Modulo Théories* : La recherche d'ordonnancement TT en utilisant le formalisme SMT a démarré avec (Steiner, 2010). Plusieurs travaux par la suite ont adopté cette technique pour différents types de problèmes d'ordonnancement TT. Nous citons dans ce cadre le travail de (Craciunas and Oliver, 2016) qui s'est intéressé à la co-synthèse des fonctions et des trames TT.

— *Programmation par contraintes* (Rossi et al., 2006) : Les concepts de base de la CP ont été introduit dans la section 2.8 et on cite (Sun et al., 2010) comme exemple de travaux qui ont utilisé le paradigme de programmation par contraintes pour la recherche d'un ordonnancement TT.

Les méthodes qu'on a présenté ci-dessus peuvent être classées en deux catégories : des méthodes où la notion de contrainte est explicite et dont ces contraintes sont exprimées directement en fonction des variables du modèle. On cite dans ce cadre la programmation en nombre

entier, la programmation linéaire mixte, la programmation par contraintes et la satisfiabilité modulo théories. Les autres méthodes sont des méta-heuristiques et reposent sur la définition d'un ensemble d'opérations qui permettent de découvrir l'espace des solutions faisables.

La première catégorie des problèmes est beaucoup plus adaptée à notre contexte étude. En effet, dans le contexte d'intégration itérative, notre modèle évolue : à chaque fois, on enrichira notre modèle par les variables et les contraintes reliées aux fonctionnalités ajoutées. Dans ce cas, il est beaucoup plus facile de gérer l'évolution de notre modèle par cette catégorie de méthodes. Nous abordons dans la section suivante les différents travaux qui se sont intéressés à l'ordonnancement TT.

3.3.2 Travaux Relatifs

La synthèse d'ordonnancement TT sur un réseau distribué constitue un point d'intérêt de plusieurs travaux de recherche. Nous citons dans ce cadre (Suethanuwong, 2012) qui propose une approche de synthèse d'ordonnancement des trames TT de TTEthernet. Dans cette approche, les auteurs supposent que les différents trames ont des périodes harmoniques. (multiple d'une plus petite) qui sera l'unité de base pour déterminer les différent ordonnancements. L'idée de cette approche est d'exprimer la relation entre deux trames successives qui appartiennent ou non au même flux pour en déduire l'ordonnancement en conséquence.

La synthèse d'ordonnancement TT peut considérer certaines contraintes spécifiques. C'est le cas du travail présenté dans (Goswami et al., 2012) qui présente une approche basée sur le formalisme Programmation linéaire en nombre entier (ILP) pour la co-synthèse d'ordonnancement TT des applications automobiles. La génération de l'ordonnancement est faite sous contraintes de stabilité tout en optimisant un caractère de performance de contrôle.

D'autres critères peuvent être optimisés lors de la synthèse d'un ordonnancement TT. Dans (Huang et al., 2012), on aborde le problème d'ordonnancement dans le cadre d'un réseau à puce. Deux critères sont optimisés dans ce cadre. En premier lieu, l'objectif est d'allouer les éléments de traitement aux commutateurs de façon à minimiser le coût de communication estimé par la distance entre le nœud source et destination. Une fois que les éléments de traitements sont alloués, l'objectif est alors de chercher un ordonnancement faisable tout en optimisant l'occupation réseau en optant pour une approche qui combine la résolution SMT avec le problème classique de bin-packing.

Dans (Lukasiewycz et al., 2009), les caractères à optimiser sont bien différents. Ce travail s'intéresse à l'optimisation d'ordonnancement TT pour le protocole FlexRay. Le problème est formulé sous la forme de bin-packing dont le but est de minimiser les fenêtres de temps

allouées. Ces fenêtres sont réorganisées par la suite pour favoriser l'ajout d'autres.

Afin de synthétiser un ordonnancement TT pour de grandes instances, d'autres travaux définissent une approche incrémentale. C'est le cas de (Steiner, 2010), qui définit en premier lieu un modèle formel pour la communication TT ainsi qu'un ensemble de contraintes modélisant le principe du transfert TT dans le cadre des réseaux TTEthernet. Ces contraintes sont résolues par la suite d'une manière incrémentale en considérant à chaque étape un sous-ensemble de trames. Dans le cadre d'une inconsistance, les auteurs procèdent par des retours en arrière puis augmentent graduellement le nombre de trames à considérer. L'approche incrémentale permet dans ce cadre de résoudre des instances larges de 100 trames et de 1000 instances de trames.

Les approches de synthèse incrémentale peuvent aussi optimiser certains critères. Dans ce contexte nous citons le travail de (Smirnov et al., 2017)qui optimise non seulement les ordonnancements établis mais aussi le routage pour minimiser la non-interférence avec les trafic non encore ordonnancé.

Une étude approfondie des performances des solveurs SMT dans la génération des ordonnancements TT a été faite dans (Pozo et al., 2015a). Différentes théories SMT pour la génération automatique des ordonnancements TT ont été étudiées dans ce cadre. Les résultats expérimentaux ont montré que la résolution suivant la théorie arithmétique linéaire en nombre entier réduit en moyenne trois fois le temps de résolution. Les auteurs ont étudié également dans le cadre d'une approche incrémentale de génération des ordonnancements TT le nombre adéquat de trames à considérer pour chaque incrément. Ils ont constaté que pour un nombre total de 1000 trames à ordonnancer, les meilleures performances en termes de temps de résolution sont obtenues avec un nombre de 6 à 22 trames par chaque incrément.

Restant dans le cadre de synthèse incrémentale, nous citons aussi le travail de (Craciunas and Oliver, 2016) qui introduit des algorithmes pour la cogénération d'ordonnancement TT aussi bien pour les messages que pour les tâches prémptives. Les auteurs définissent dans ce cadre des contraintes d'ordonnançabilité avec le formalisme SMT. L'idée adoptée par ce travail est d'ordonnancer seulement les tâches communicantes avec le solveur SMT. Un test d'ordonnançabilité basé sur la demande d'utilisation permet de vérifier lors de l'ajout de chaque incrément si l'ensemble des tâches est ordonnançable. D'une manière similaire, la cosynthèse des ordonnancements TT a été étudiée aussi dans (Craciunas and Oliver, 2014). Un test d'ordonnançabilité similaire a été défini dans ce cadre pour analyser l'ordonnançabilité des tâches asynchrones périodiques. L'évaluation a été faite par rapport à plusieurs topologies et configurations de systèmes et a montré que cette approche peut viser des problèmes de taille moyenne à large d'une manière efficiente. L'approche peut ainsi cibler des problèmes

de taille industrielle.

La définition d'une approche incrémentale peut être défini non pour maitriser la complexité comme le cas de (Sagstetter et al., 2016). En effet, l'objectif dans ce cadre est de synthétiser plusieurs ordonnancement pour une variété de configurations de systèmes qui sont destinés pour plusieurs clients.

L'adoption d'une approche incrémentale n'est pas l'unique solution pour synthétiser un ordonnancement TT pour des instances larges. Dans (Pozo et al., 2015b), une approche de décomposition est définie pour cette fin en divisant les instances des trames sur des intervalles. La décomposition faite dans ce cadre présente des avantages par rapport à celle présentée dans (Steiner, 2010). En effet, la décomposition est faite d'une part d'une manière parallèle, chaque sous-ensemble de trames est ordonnancé à part et puis la décomposition est faite de manière à considérer le minimum de dépendance entre les sous-ensembles des trames.

La décomposition est faite aussi dans (Pozo Pérez et al., 2017) en introduisant une approche de segmentation du problème d'ordonnancement TT en sous problèmes solvables facilement par les solveurs SMT. La segmentation est faite dans ce cadre en divisant l'hyperpériode en sous-intervalles de temps disjoints. L'évaluation de l'approche montre qu'elle peut générer des ordonnancements pour des systèmes avec plus d'une centaine de noeuds avec une augmentation linéaire en fonction du nombre de trames. Les auteurs ont également étudié le choix de la taille du segment et son impact sur les performances globales de l'approche. La relaxation des contraintes a été aussi envisagée et a permis une augmentation importante au niveau des performances de la synthèse tout en conservant une bonne qualité des ordonnancements générés.

La limite de l'approche présentée dans (Pozo et al., 2015b) est qu'elle n'est valide que si les segments sont indépendants. De plus, elle suppose que la longueur d'un segment d'ordonnancement est plus petite que la plus petite période de trame. Cette supposition a pour objectif d'exclure plus d'une instance de trame dans le cycle d'ordonnancement. Ce qui ne tient pas lors qu'il s'agit de réseaux larges. Dans (Pozo et al., 2016), les auteurs proposent une nouvelle approche qui prend en compte les périodes des trames en allouant ces trames plusieurs fois dans l'ordonnancement. Ces trames, étant dépendantes les unes des autres, sont gérées par un ensemble de contraintes qui expriment leurs positions relatives. Les ordonnancements des différents segments sont rassemblés par la suite pour former l'ordonnancement global.

La décomposition peut être faite suivant d'autres facteurs que le temps. C'est le cas de (Sagstetter et al., 2013) qui synthétisent séparément des ordonnancements de divers services automobiles puis les combinent ensemble. Lors de la phase de fusion des différents services et en cas de conflit, les auteurs définissent deux stratégies pour gérer ce conflit. Il s'agit soit

de déplacer l'ordonnancement d'un processus d'un service ou de décaler l'ordonnancement de l'ensemble du service. Nous résumons dans le tableau 3.5 les travaux relatifs à la synthèse d'ordonnancement TT. La deuxième colonne du tableau indique si le problème considéré est un problème d'optimisation et éventuellement le critère à optimiser. La troisième colonne indique s'il s'agit d'un problème de co-synthèse. Plus précisément, elle indique si le problème synthétise aussi l'ordonnancement des applications distribuées. Nous indiquons dans la quatrième colonne si dans le travail présenté une approche incrémentale ou une décomposition est adoptée. Pour les approches qui s'appuient sur une décomposition, nous indiquons entre parenthèses le critère de décomposition. La dernière colonne indique le formalisme adopté pour la synthèse d'ordonnancement TT.

Le problème que nous considérons est un problème de co-synthèse. En effet, à part la synthèse d'ordonnancement TT, nous synthétisons aussi l'ordonnancement des partitions IMA. Le critère que nous optimisons est bien différent des critères présentés par ces travaux. En effet, nous optimisons plutôt le coût associé à la reconfiguration. Nous adoptons également une approche incrémentale. En effet, nous considérons pour chaque étape l'ajout d'un sous-ensemble des fonctionnalités avioniques. Quant au formalisme adopté, nous optons pour une approche SMT et une autre basée sur le formalisme de programmation par contraintes.

Un problème connexe à la synthèse d'ordonnancement consiste à chercher le routage adéquat qui permet de véhiculer les trames TT d'une manière optimale. Dans (Gavrilu et al., 2015), les auteurs considèrent des applications distribuées à sureté critique et communicantes conformément au protocole TTEthernet. Les auteurs ont proposé une approche qui se base sur la méta-heuristique du recuit simulé et dont l'objectif est de chercher la topologie réseau telle que les propriétés temps-réel et de sureté des applications sont satisfaites et que le coût de l'architecture est minimisé.

La synthèse d'ordonnancement TT dépend étroitement de la qualité de synchronisation établie entre les différents noeuds distribués. Dans (Steiner and Dutertre, 2010), les auteurs s'adressent un problème de vérification formelle de la fonction de compression qui est essentielle pour la synchronisation. On note ici que la fonction de compression s'exécute au niveau du commutateur TTEthernet et sert à collecter les horloges des systèmes terminaux puis calculer une valeur moyenne quelle retourne aux systèmes terminaux. Le travail de vérification dans ce cadre consiste à modéliser le comportement de cette fonction et de vérifier moyennant les techniques de vérification de modèle la terminaison de cette fonction. La tolérance aux fautes de la fonction de synchronisation des horloges dans le cadre des réseaux TTEthernet a été faite aussi dans (Steiner and Dutertre, 2011). Les auteurs vérifient d'une part la précision des horloge qui est déterminé par le décalage maximal entre deux horloges du système ainsi

Tableau 3.5 Travaux relatifs à la synthèse d'ordonnancement TT

Reférence	Optimisation	Co-Synth	Inc/Déc	Approche
Suethanuwong (2012)	×	×	×	Cluster Cycle
Goswami et al. (2012)	Performance	✓	×	ILP
Huang et al. (2012)	Coût Comm	×	Inc	SMT
Lukasiewycz et al. (2009)	Util. Bus	×	Inc	ILP+heuristique
Steiner (2010)	×	×	Inc	SMT
Smirnov et al. (2017)	Plusieurs	✓	×	SAT
Pozo et al. (2015a)	×	×	Inc	SMT
Craciunas and Oliver (2016)	latence	✓	Inc	SMT+MIP
Craciunas and Oliver (2014)	×	✓	Inc	SMT
Sagstetter et al. (2016)	×	✓	Inc	SMT
Pozo et al. (2015b)	×	×	Déc(trame)	SMT
Pozo Pérez et al. (2017)	×	×	Déc(trame)	SMT
Pozo et al. (2016)	×	×	Déc(période)	SMT
Sagstetter et al. (2013)	×	✓	Déc(applicatif)	ILP

la qualité de la synchronisation en supposant des échecs aux niveaux de noeuds intervenant dans la synchronisation.

Afin d'évaluer les ordonnancements TT synthétisés, certains travaux s'appuient sur la simulation. C'est le cas de (Robati et al., 2016) qui présente une approche à base de transformation de modèles pour simuler les applications avioniques déployées sur des systèmes IMA avec communication TTEthernet. Les modèles en entrées ont été spécifiés moyennant une extension du langage de modélisation AADL puis transformés sous forme de modèles pour la simulation. Le but de simulation dans ce cadre est de vérifier les contraintes temporelles de TTEthernet.

Le protocole TTEthernet a été défini pour les applications critiques distribuées communicant à travers des réseaux câblés. Dans (Peon et al., 2015), les auteurs proposent trois nouveaux protocoles d'accès au canal pour une communication temps-critique sans fil. Ces trois protocoles permettent de véhiculer les différentes classes de trafic de TTEthernet. Nous nous intéressons dans la section suivante aux approches qui analysent et vérifient les flux synchrones et asynchrones de TTEthernet.

3.4 Analyse et Ordonnancement des réseaux à flux synchrones et asynchrones

Nous commençons notre étude par le travail de (Abuteir and Obermaisser, 2015), qui présente un algorithme d'ordonnancement pour des applications avec une interdépendance entre les

tâches. Il s'agit dans ce cadre d'applications multi-critiques utilisant les deux types de flux TT et RC. L'algorithme développé procède par recherche locale et optimise les chemins critiques du réseau. L'approche a été évaluée par des cas d'études larges en utilisant l'environnement de simulation OPNET (Chang, 1999).

D'autres approches s'intéressent seulement à l'analyse de l'ordonnnançabilité des flux RC tout en considérant la présence de flux TT. C'est le cas (Zhao et al., 2014) qui propose une amélioration de la courbe d'arrivée pour modéliser les flux TT en se basant sur la méthode du calcul réseau. L'approche développée considère aussi l'influence des ordonnancements TT et dérive des bornes de pire cas de latence plus strictes pour les flux RC en identifiant les ressources résiduelles en termes de temps. L'exactitude de l'approche a été analysée par des simulations en considérant une courbe d'arrivée minimale et une comparaison des bornes établies a été faite par rapport à la méthode adoptant une courbe d'arrivée classique.

D'une manière similaire. mais en prenant comme référence l'approche par trajectoire, on présente dans (TamasSelicean et al., 2015a) une méthode pour analyser les délais dans des réseaux TTEthernet tout en prenant compte l'existence de deux flux TT et RC. Pour analyser les délais des flux RC, les auteurs définissent la période d'occupation qui exprime le retard que peux subir une trame pour être transmise sur un lien. Ce retard est dû soit aux fenêtres de temps TT, la transmission d'autre trames RC, ou aux fenêtres de blocages exigés pour les trames RC. Les auteurs définissent une fonction récursive qui permet de calculer cette période d'occupation pour chaque trame sur chaque lien puis moyennant ces différents périodes, ils définissent un algorithme qui estime le pire cas de transmission de ces trames.

D'autres approches adoptent une méthodologie spécifique pour la vérification des flux TT et RC. C'est le cas de (Kermia, 2015) qui propose une approche en deux phases. Dans la première phase, l'objectif est d'analyser la faisabilité des messages TT. Pour ce fait un test de faisabilité suffisant a été proposé. Ce test est basé sur le facteur d'utilisation strictement périodique qui retourne la somme des fractions de temps utilisé pour transmettre les messages TT lorsque leurs périodes strictes sont satisfaites. Ce test permet d'aider le concepteur de réseau TTEthernet à savoir l'existence d'une table d'ordonnancement TT. Dans la deuxième étape les auteurs visent l'analyse d'ordonnançabilité des messages RC tout en prenant compte de l'existence des messages TT et les temps de silence requis pour permettre l'intégration des flux TT et RC.

En plus de la synthèse d'ordonnancement TT et la vérification des flux RC certains travaux cherchent à optimiser certains critères. Nous citons (Tamas-Selicean et al., 2012a) qui se place dans le contexte des applications temps réel dures implémentées sur des architectures hétérogènes. En considérant un ensemble de messages TT et RC et la topologie des liens

virtuels sur lesquelles les messages sont assignés, l'objectif dans ce cadre est de synthétiser des ordonnancements TT de telle sorte que les latences de bout en bout des messages RC sont optimisés. Pour explorer l'espace de solution, les auteurs adoptent un algorithme tabou qui peut soit avancer la fenêtre de temps pour la trame sélectionnée, avancer les fenêtres de ces prédécesseurs, retarder la fenêtre de la trame considérée ou retarder les fenêtres de ces successeurs.

D'une manière similaire, on introduit dans (Murshed et al., 2015) un modèle d'ordonnancement pour les processeurs multi-coeurs. avec un réseau à puce supportant à la fois le trafic déclenché par le temps ainsi que celui déclenché par les évènements. Les auteurs cherchent dans ce cadre à co-synthétiser un ordonnancement pour les tâches ainsi que pour la communication tout en minimisant la latence.

Une analyse profonde de ces travaux nous permettent de constater que certains d'entre eux ne visent qu'à la vérification des flux RC tout en prenant compte de la présence des flux TT. Généralement, on opte dans ce cadre à la réadaptation des approches célèbres dans ce domaine soit la méthode de calcul réseau ou l'approche par trajectoire et on réfère dans ce contexte aux travaux de (Zhao et al., 2014) et à (TamasSelicean et al., 2015a). D'autres travaux qui cherchent aussi à synthétiser un ordonnancement TT, adoptent généralement une heuristique. Une telle heuristique commence généralement par une solution initiale qui n'est pas nécessairement faisable, mais parcourt l'ensemble des solutions pour en découvrir une qui soit adéquate. Nous citons dans ce cadre (Tamas-Selicean et al., 2012a) qui définit une approche à base de recherche tabou. L'inconvénient d'une telle approche est qu'elle n'est pas bien adaptée au contexte d'intégration itérative où on se place dans le cadre de systèmes évolutifs. Telles approches ne modélisent pas cet aspect.

D'autres travaux de recherche vont au-delà de la vérification des flux critiques et cherchent à optimiser le trafic BE. Nous citons dans ce cadre (Suethanuwong and Hirunmutrapom, 2016) qui présente une approche alternative à la fragmentation du trafic BE. Cette approche consiste plutôt à chercher une trame de taille plus petite qui peut être véhiculée durant l'intervalle de blocage. Cette méthode présente un avantage en n'exigeant aucune modification des systèmes terminaux et des commutateurs TTEthernet pour la réception des messages BE. En plus cette approche ne requiert pas un contenu additionnel pour les entêtes Ethernet. Dans (Tamas-Selicean and Pop, 2014), les auteurs proposent plutôt à cette fin une stratégie d'optimisation de configuration des réseaux TTEthernet qui maximise la bande passante laissée pour les flux BE.

Dans (Steiner, 2011), on s'intéresse seulement aux flux critiques en abordant un problème résultant de la coexistence des deux flux TT et RC. Il s'agit de la famine des messages RC quand

les fenêtres des messages TT sont collées les unes aux autres ce qui empêche de véhiculer les messages RC. Pour résoudre ce problème, les auteurs présentent trois stratégies pour garantir une porosité suffisante entre les messages TT. La première méthode dite a priori définit des intervalles blancs avant même la recherche de l'ordonnancement TT. La deuxième, dite posteriori, modifie les ordonnancements déjà synthétisés pour garantir une distance temps entre les ordonnancements planifiés. Quant à la troisième méthode, elle consiste à planifier des fenêtres de temps suffisantes pour les messages RC de taille maximale.

Dans un contexte plus général, certains travaux ne se limitent pas à la synthèse d'ordonnancement TT et la vérification RC. Nous citons (Tămaş-Selicean et al., 2015) qui considèrent des applications temps-réel multi-critiques implémentés sur des architectures distribuées hétérogènes. Étant donné un ensemble de messages TT et RC et la topologie du réseau, les auteurs ont proposé une stratégie d'optimisation basée sur la méthode de recherche tabou qui cherche à empaqueter les messages dans des trames, assigner des trames aux liens virtuels, le routage des liens virtuels et la synthèse des ordonnancements TT. L'optimisation est faite de sorte que les trames sont ordonnançables et le degré d'ordonnançabilité est optimisé.

Nous citons aussi (Tamas-Selicean et al., 2012b) qui présente un outil pour l'analyse et l'optimisation des applications multi-critiques dans des architectures partitionnées. L'outil prend comme entrée le modèle d'application représenté par les messages et les tâches, le modèle de la plateforme système et produit la configuration du système qui garantit la non-interférence entre les applications critiques et leurs ordonnançabilité. L'optimisation dans le cadre de ce travail décide l'allocation des éléments de traitement sur l'architecture, la séquence et la longueur des partitions sur chaque élément de traitement, l'ordonnancement des messages TT de telle sorte que les applications sont ordonnançables et que le temps de réponse des messages TT et RC soit minimisé. Pour résoudre ce problème, les auteurs définissent une approche basée sur la recherche Tabou. Nous résumons dans le tableau 3.6 les travaux liés à l'ordonnançabilité des flux TT et RC. Nous indiquons dans la deuxième colonne si le problème considère un critère d'optimisation. La troisième colonne indique si, à part la vérification de l'ordonnancabilité, l'approche synthétise un ordonnancement TT pour le réseau ou si elle co-synthétise un ordonnancement pour les applications distribuées et le réseau. La troisième colonne indique le formalisme utilisé pour l'approche. La dernière colonne indique si, à part la vérification de l'ordonnancabilité, l'approche présentée considère un autre objectif.

Tableau 3.6 Travaux relatifs à la synthèse d'ordonnancements TT et l'analyse de l'ordonnançabilité des flux RC

Référence	Optimisation	Synth/Co-Synth	Approche	Autre
Abuteir and Obermaisser (2015)	×	Co-Synth	Recherche locale	
Zhao et al. (2014)	×	×	Calcul réseau	Amélioration analyse latence
Tămaş-Selicean et al. (2015)	×	Synth	Recherche Tabou	Fragmentation+Routage
Kermia (2015)	×	×	Approche dédiée	-
Tamas-Selicean et al. (2012a)	Latence RC	Synth	Recherche Tabou	
Murshed et al. (2015)	Latence TT et ET	Co-Synth	MILP	Routage TT
Suethanuwong and Hirunmutrapom (2016)	Bande passante BE	×	Shorter Frame Search	-
Tamas-Selicean and Pop (2014)	Bande passante BE	Synth	Recherche Tabou	Ordonnancabilité TT, RC
Steiner (2011)	Porosité	Synth	SMT	-
TamasSelicean et al. (2015b)	×	×	Approche par trajectoire	-
Tamas-Selicean et al. (2012b)	Temps Réponse TT+RC	Co-Synth	Recherche Tabou	Affectation tâches aux partitions

Une simulation des flux TT et RC est aussi importante dans le cadre de vérification. Elle permet de simuler le comportement réel d'un système ou réseau donné. Les actuelles plateformes de simulation réseau (ex., OPNET, OMNeT++) ne supportent pas la communication TT et RC de TTEthernet. Vu les différences entre TTEthernet et les autres protocoles TT (e.g., CANoe Option FlexRay), ces environnements ne sont pas applicables pour le cadre de TTEthernet. Malgré qu'un modèle de simulation a été développé pour étudier la synchronisation des horloges et valider ainsi les algorithmes d'ordonnancements (Suethanuwong, 2011), ce modèle n'est pas générique et ne modélise pas certains comportements et configurations des applications communicantes suivant le protocole TTEthernet. Dans (Abuteir and Obermaisser, 2013), les auteurs présentent un environnement simulation générique qui peut être configuré et étendu pour la simulation d'applications spécifiques.

Suite à l'intérêt considérable à l'adoption TTEthernet, on explore dans (Elshuber and Obermaisser, 2013) la possibilité d'étendre les routeurs Ethernet pour supporter le trafic des différents flux de TTEthernet. Ceci permet de réduire le coût de hardware et offre la possibilité d'utilisation des infrastructures anciennes pour des applications temps-réelles.

3.5 Conclusion

Nous avons présenté dans ce chapitre un aperçu des différents travaux de recherches reliés à notre problématique de recherche. Dans une première partie, nous avons présenté les différents travaux relatifs aux systèmes IMA. Nous avons étudié par la suite différents techniques et travaux relatifs à la vérification des flux à quota limité et à la synthèse d'ordonnancement synchrone. Vers la fin, nous avons exploré différents travaux visant la vérification simultanée de ses deux types de flux.

CHAPITRE 4 INTÉGRATION ITÉRATIVE DES FONCTIONNALITÉS AVIONIQUES AVEC UNE COMMUNICATION DÉCLENCHÉE PAR LE TEMPS

Nous présentons dans ce chapitre notre approche pour l'intégration itérative des fonctionnalités avioniques déployées sur une architecture IMA interconnectée par un réseau TTEthernet. Cette approche a été publié dans (Beji et al., 2014) et fait aussi l'objet de la soumission (Beji et al., 2018a). Cette dernière considère un ordonnancement IMA et le trafic TT des fonctions déjà intégrées et cherche une meilleure façon en termes de coût de reconfiguration pour intégrer une nouvelle fonctionnalité. Cette intégration consiste à co-synthétiser un nouvel ordonnancement IMA et du trafic TT tout en assurant les exigences du système et en minimisant aussi le coût associé à la reconfiguration des parties déjà intégrées. Le coût ici peut exprimer les ressources utilisées pour la reconfiguration et/ou pour la recertification.

Pour résoudre ce problème, nous adoptons la méthodologie suivante. Nous présentons dans la section 4.1 un modèle formel pour l'architecture avionique et dans la section 4.2 un modèle de configuration associé. En nous basant sur ces deux modèles, nous définirons dans la section 4.3 un ensemble de contraintes pour modéliser les exigences temporelles d'un système avionique. L'ensemble des contraintes serviront dans la section 4.4 à définir notre approche d'intégration. À travers deux cas d'études, nous analyserons dans la section 4.5 notre approche et nous dégagerons quelques directives qui peuvent aider l'ingénieur système dans le choix de sa meilleure stratégie d'intégration et nous clôtureroms cette section par une étude de l'évolutivité de notre approche.

4.1 Modèle Formel d'architecture avionique

Le but de cette partie est de décrire d'une manière formelle et générique une architecture avionique. Ce modèle est construit en deux parties. Une première partie décrira l'architecture alors que la deuxième partie modélisera les applications avioniques déployées sur cette architecture.

4.1.1 Modèle d'Architecture

Dans cette section, nous introduisons le modèle d'architecture des modules distribués IMA interconnectés par un réseau TTEthernet. Concernant la communication déclenché par le temps, nous étendrons un modèle similaire à celui introduit dans (Steiner, 2010).

Un système avionique, comme le montre la figure 4.1, est une *grappe TTEthernet* de modules IMA généralement referrés par *End Systems* (ESes), interconnectés moyennant des liens physiques et des *commutateurs réseaux* TT notés (NSes). Chaque grappe TTEthernet est modélisée par un graphe non orienté $\mathcal{G}(\mathcal{V}, \mathcal{E})$. L'ensemble des sommets $\mathcal{V} = \mathcal{ES} \cup \mathcal{NS}$ est composé d'un ensemble de systèmes terminaux ($\mathcal{ES}$) et un ensemble de commutateurs réseaux ($\mathcal{NS}$). L'ensemble des arêtes $\mathcal{E}$ sont les liens physiques connectant les systèmes terminaux et les commutateurs réseau. Dans l'exemple illustratif de la figure 4.1, on a $\mathcal{ES} = \{\mathrm{M}_1, \mathrm{M}_2, \cdots, \mathrm{M}_5\}$ et $\mathcal{NS} = \{\mathrm{NS}_1, \mathrm{NS}_2\}$.

Chaque système terminal (module IMA) est composé d'un ensemble fini non vide de partitions et chaque arête (lien physique) est un lien de données bidirectionnel Nous notons par $\mathcal{L}$ l'ensemble des liens de données orientés du réseau. Nous notons aussi par (u, v) le lien physique qui connecte les nœuds et par $[u, v]$ le lien de donné du nœud u au nœud v. L'ensemble des arêtes et les liens de données du réseau sont ainsi reliés comme suit.

$$\forall u, v \in \mathcal{V} : (u, v) \in \mathcal{E} \Rightarrow [u, v] \in \mathcal{L} \text{ and } [v, u] \in \mathcal{L} \tag{4.1}$$

Une séquence de liens de données adjacents et orientés $p = [[v_1, v_2], ..., [v_{r-1}, v_r]]$ connectant un module $v_1 \in \mathcal{ES}$ à un autre $v_r \in \mathcal{ES}$ à travers un chemin non cyclique de commutateurs réseaux $v_i \in \mathcal{NS}\,(2 \leq i \leq r-1)$ forme un *chemin de donnée*. La figure 4.1 illustre, moyennant les lignes pointillées, un exemple de chemin de donnée du module M_1 au module M_3.

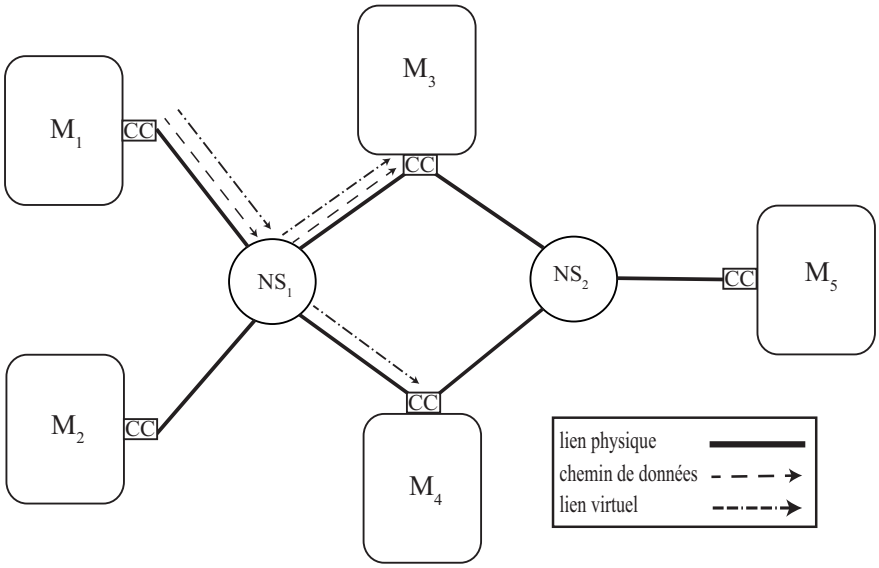


Figure 4.1 Exemple d'une architecture avionique

L'ensemble de tous les chemins de données et notés par $\mathcal{DP}$.

TTEthernet utilise le concept de *lien virtuel* (ARINC, 2009) pour définir le partitionnent spatial des messages ayant différents niveaux de criticité et transmis sur le même lien physique. Un lien virtuel noté par vl, transporte un message unique d'un module source à un ou plusieurs modules destinations. Ainsi, il représente une *structure d'arbre* où la racine est le module émetteur, les modules destinations sont les feuilles et les nœuds internes de l'arbre sont les commutateurs réseau. Chaque chemin de la racine à une feuille forme un chemin de données. De plus, la racine a *un seul fils* (qui doit être un nœud *interne*) et les feuilles sont des modules disjoints deux à deux. En d'autres termes, un lien virtuel est un ensemble de chemins de données transportant un message d'une source à une ou plusieurs destinations. La figure 4.1 illustre un lien virtuel qui est composé par deux chemins de données.

$$vl_1 = \{[[M_1, NS_1], [NS_1, M_3]], [[M_1, NS_1], [NS_1, M_4]]\} \tag{4.2}$$

Nous notons par $\mathcal{VL}$ l'ensemble des liens virtuels et pour chaque lien virtuel $vl \in \mathcal{VL}$, nous utilisons la notation fonctionnelle suivante pour exprimer :

- *first*(vl) : le premier lien de donnée dont lequel le message est transmis. Ce lien est bien défini puisque la racine a un seul fils (ex. *first*$(vl_1) = [M_1, NS_1]$) ;
- *last*(vl) : l'ensemble de dernier liens de données sur le lien virtuel (e.g. *last*$(vl_1) = \{[NS_1, M_3], [NS_1, M_4]\}$) ;
- *root*(vl) : la racine du lien virtuel. (e.g. *root*$(vl_1) = M_1$) ;
- *leaves*(vl) : l'ensemble des feuilles du lien virtuel (e.g. *leaves*$(vl_1) = \{M_3, M_4\}$).
- *links*(vl) : l'ensemble des liens de donnés dans le lien virtuel (e.g. *links*$(vl_1) = \{[M_1, NS_1], [NS_1, M_3], [NS_1, M_4]\}$).

4.1.2 Modèle d'application

Nous définissons dans cette section notre modèle formel d'application pour le problème d'intégration itérative. Nous commençons par formaliser les concepts IMA de modules et de partitions. Par la suite, nous présentons un modèle formel pour le flux de communication TT. Finalement, nous définissons la notion d'une fonctionnalité avionique et nous concluons la section par la formalisation de l'intégration d'une nouvelle fonctionnalité dans un système existant.

Concepts IMA

Nous notons un système IMA distribué par $\mathcal{M} = \{M_1, ..., M_n\}$. $\mathcal{M}$ représente le système au complet et il est composé d'un ensemble fini de modules. Chaque module M est un ensemble fini de partitions $M = \{P_1, P_2, \cdots, P_m\}$, où P_i désigne la $i^{\text{ème}}$ partition de M. Dans le cadre de ce travail, nous considérons que les partitions sont des *unités atomiques d'exécution d'applications temps réel* modélisant une fonction de traitement, lecture d'un capteur, actionnement des composants mécaniques et hydro-électriques, etc. Chaque partition P, implémentée dans un module M, est exécutée périodiquement. Ainsi, nous caractérisons temporellement une partition comme suit :

$$(P.C, P.T)$$

où $P.C$ désigne le pire temps d'exécution d'une partition et $P.T$ sa période. Ces deux paramètres temporaux sont connus *a priori* et ils sont données par l'ingénieurs système. Le plus petit multiple commun de toutes les périodes de partitions exécutées au sein du même module M est appelés le MAF, connu aussi sous *Hyper Période* du module (HP_M). Nous notons qu'un système IMA distribué $\mathcal{M}$ est déployé sur une grappe avionique $\mathcal{G}(\mathcal{ES} \cup \mathcal{NS}, \mathcal{E})$ alors $\mathcal{ES} = \mathcal{M}$.

Modèle d'application réseau

L'information communiquée entre la source et les destinations est sous forme de messages appelés *trames*. Chaque trame déclenchée par le temps noté f est aussi envoyé périodiquement et elle est temporellement caractérisée suivant le modèle de (Steiner, 2010) par l'ensemble de paramètres suivants :

$$(f.C, f.T),$$

où $f.C$ représente la durée de transmission d'une trame et $f.T$ la périodicité de son envoi. Nous exprimons dans le cadre de ce travail $f.C$ en termes d'unité temporelle pour représenter le temps nécessaire pour une trame pour qu'elle soit complètement transmise sur un lien donné. Le plus petit multiple commun de toutes les périodes sur le réseau est appelé le cycle d'application, T_{cycle}. Les messages que les partitions produisent sont communiqués sous forme de trames périodiques. Vu que multiples messages avec différents niveaux de criticité peuvent être transmises sur le même lien et pour assurer un partitionnement spatial de ces messages, chaque trame est mise en correspondance avec son lien virtuel. Étant donné un ensemble

de partitions $\mathcal{P}$ et un ensemble de trames $\mathcal{F}$ échangés par ces partitions, nous utilisons les fonctions auxiliaires suivantes pour lier les partitions qui produisent ou consomment les messages avec les liens virtuels associés à leurs trames.

— $\alpha : \mathcal{P} \rightarrow \mathcal{M}$, une allocation de partition qui associe les partitions aux systèmes terminaux (modules).

— $\Omega : \mathcal{P} \rightarrow 2^{\mathcal{F}}$, une fonction qui associe chaque partition P à l'ensemble des trames produites par P, de sorte $P \neq P'$ implique que $\Omega(P) \cap \Omega(P') = \emptyset$ et $\cup_{P \in \mathcal{P}} \Omega(P) = \mathcal{F}$. En d'autres termes, chaque trame est produite par une partition unique. Étant donné une trame f, nous utilisons la notation fonctionnelle $source(f)$ pour désigner la partition qui produit les messages transportés par f. Nous notons que $source(f) = P$ si et seulement si $f \in \Omega(P)$.

— $\Delta : \mathcal{F} \rightarrow 2^{\mathcal{P}}$, la fonction qui associe chaque trame f à l'ensemble (non-vide) des partitions destinations qui consomment les messages transportés par la trame f.

— $\Pi : \mathcal{F} \rightarrow \mathcal{VL}$, une association des trames aux liens virtuels. Nous notons aussi que deux trames différentes peux être produites par des fonctions (partitions) implémentées au sein du même module et peut être consommées par des partitions implémentées aussi sur le même module. Dans ce cas, leurs liens virtuels associés ont le même ensemble de chemins. Ainsi, nous utilisons la fonction Π pour identifier d'une manière unique les liens virtuels puisque chaque trame de même chaque message est produit par une unique partition.

Une fonctionnalité avionique est un ensemble de fonctions distribuées et communicantes pour réaliser un sous-système avionique.

Définition 1 (Fonctionnalité Avionique) *Étant donné une grappe TTEthernet $\mathcal{G}(\mathcal{V}, \mathcal{E})$, une fonctionnalité avionique déployée sur $\mathcal{G}$ est défini par le 6-uplet func $\stackrel{def}{=} (\mathcal{P}, \mathcal{F}, \alpha, \Omega, \Delta, \Pi)_{\mathcal{G}}$ où*

— *$\mathcal{P}$ est l'ensemble (distribué) des partitions communicantes ;*

— *$\mathcal{F}$ est l'ensemble de trames échangées par ces partitions ;*

— *α est la fonction qui alloue les partitions aux systèmes terminaux ;*

— *Ω est la fonction qui associe chaque partition à l'ensemble des trames que les partitions produisent ;*

— *Δ est la fonction qui associe chaque trame à l'ensemble de ces destinations ;*

— *Π est la fonction qui associe chaque trame au lien virtuel qui transporte ses messages ;*

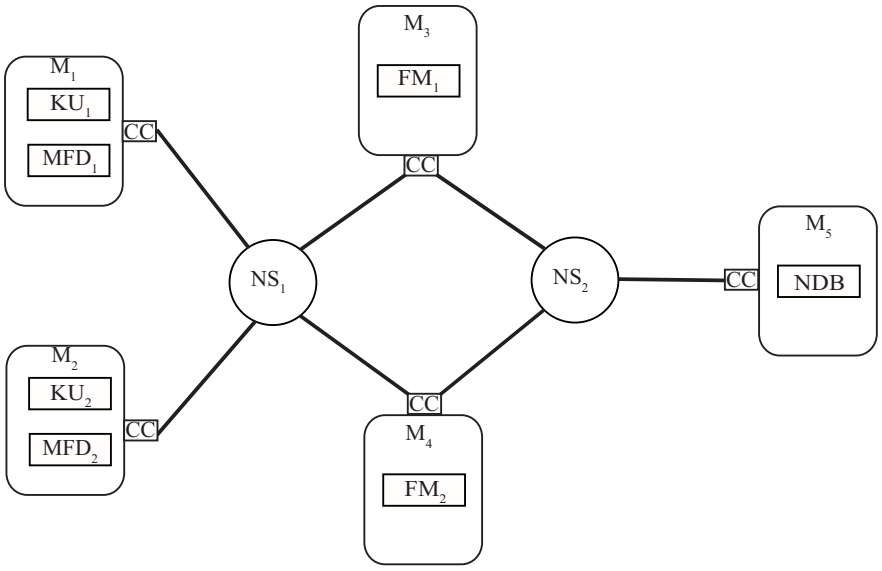


Figure 4.2 Un sous-système du système gestion de vol

telle que pour chaque f dans $\mathcal{F}$, la source ainsi que les destinations de f appartiennent à $\mathcal{P}$ ainsi, source$(f) \in \mathcal{P}$ et $P \in \mathcal{P}$.

De plus, pour chaque trame f dans $\mathcal{F}$, les propriétés suivantes doivent être vérifiés : *source*$(f) \in$ *root*$(\Pi(f))$ et si $P \in \Delta(f)$ alors $\alpha(P) \in$ *leaves*$(\Pi(f))$. En d'autre termes, la partition émettrice doit appartenir à la racine du lien virtuel et les partitions réceptrices doivent être dans les feuilles. Pour des raisons de lisibilité, nous enlèverons la notation de grappe TTEthernet $\mathcal{G}$ sauf si ce n'est pas clair du contexte.

La figure 4.2 montre un exemple de déploiement d'un sous-système du système de gestion de vol sur la grappe TTEthernet donnée par la figure 4.1. Ce sous-système contrôle l'affichage de l'information du waypoint sur les écrans des pilotes.

La fonctionnalité décrivant ce sous-système noté par $func_1$ est comme suit. Deux partitions KU_i $(i \in \{1, 2\})$, pour *Keyboard and Cursor Unit*, envoient l'information de la requête du waypoint par le pilote et le copilote. Un *waypoint Id*, $wpId_i$ $(i \in \{1, 2\})$, est alors transmis comme un identifiant de requête aux deux partitions *Flight Manager* FM_1 et FM_2. Chaque FM_i réagit en conséquence et envoie $query_i$ à *NDB*, le *Navigation DataBase*. Cette dernière envoie les informations nécessaires pour le calcul du waypoint comme $answer_i$. Par la suite, chaque FM_i $(i \in \{1, 2\})$ calcule l'information du waypoint $wpInfo_i$ et la transmets pour son associé MFD_i, le *Multi Functional Display*. Ce dernier affiche $wpInfo_i$ sur l'écran correspondant. L'affectation des partitions aux modules est aux liens virtuels associés est décrite par

le tableau 4.1. Par exemple, la partition KU_1 est allouée au module M_1 et elle produit une seule trame $wpId_1$. Cette trame a deux destinations FM_1 et FM_2 et son associé lien virtuel est $\{[[M_1, NS_1], [NS_1, M_3]], [[M_1, NS_1], [NS_1, M_4]]\}$.

Tableau 4.1 Communication de la fonctionnalité $func_1$

P	$\alpha(P)$	$\Omega(P)$	$\Delta(f)$	$\Pi(f)$
KU_1	M_1	$wpId_1$	FM_1	$[[M_1, NS_1], [NS_1, M_3]]$
			FM_2	$[[M_1, NS_1], [NS_1, M_4]]$
FM_1	M_3	$query_1$	NDB	$[[M_3, NS_2], [NS_2, M_5]]$
		$wpInfo_1$	MFD_1	$[[M_3, NS_1], [NS_1, M_1]]$
KU_2	M_2	$wpId_2$	FM_1	$[[M_2, NS_1], [NS_1, M_3]]$
			FM_2	$[[M_2, NS_1], [NS_1, M_4]]$
FM_2	M_4	$query_2$	NDB	$[[M_4, NS_2], [NS_2, M_5]]$
		$wpInfo_2$	MFD_2	$[[M_4, NS_1], [NS_1, M_2]]$
NDB	M_5	$answer_1$	FM_1	$[[M_5, NS_1], [NS_1, M_3]]$
		$answer_2$	FM_2	$[[M_5, NS_1], [NS_1, M_4]]$
MFD_1	M_1	$\emptyset$		
MFD_2	M_2	$\emptyset$		

Nous définissons un système avionique complet comme un ensemble de fonctionnalités.

Définition 2 (Système Avionique) *Étant donné une grappe TTEthernet notée $\mathcal{G}(\mathcal{V}, \mathcal{E})$, un système avionique $sys \stackrel{def}{=} \{(\mathcal{P}_r, \mathcal{F}_r, \alpha_r, \Omega_r, \Delta_r, \Pi_r)_{\mathcal{G}} : 1 \leq r \leq n\}$ est un ensemble fini de fonctionnalités avioniques déployées sur la même grappe $\mathcal{G}$ telle que pour tout $r, r' \leq n$ $(r \neq r')$, $\mathcal{F}_r \cap \mathcal{F}_{r'} = \emptyset$ et si $P \in \mathcal{P}_r \cap \mathcal{P}_{r'}$ alors $\alpha_r(P) = \alpha_{r'}(P)$.*

Nous désignons par $\mathcal{P}^{sys} \stackrel{\text{def}}{=} \cup_{1 \leq r \leq n} \mathcal{P}_r$ et par $\mathcal{F}^{sys} \stackrel{\text{def}}{=} \cup_{1 \leq r \leq n} \mathcal{F}_r$ l'ensemble de toutes les partitions (respectivement toute les trames) dans *sys*. En nous basant sur ce modèle formel d'architecture IMA, nous modélisons formellement dans la section 4.2 les configurations avioniques ainsi que le processus d'intégration. Par la suite, nous formalisons dans la section 4.3 l'ensemble des contraintes qui doivent être vérifié par les différentes configurations avioniques.

4.2 Modèle de configuration avionique

Étant donné le modèle formel d'application défini dans la section 4.1.2, la configuration d'un système avionique peut être subdivisée en deux sous-parties : l'ordonnancement des partitions sur des modules IMA distribués et l'ordonnancement des trames sur le réseau. Ainsi, étant donné un système avionique *sys*, on désigne par $(\psi, \phi)_{sys}$ la configuration avionique où,

ψ est la configuration des modules IMA et ϕ est celles des trames. L'objectif de la fonction d'ordonnancement ψ (respectivement ϕ) est de réserver pour chaque partition (respectivement trame) déployée sur le système une fenêtre de temps durant laquelle une instance de la partition (respectivement de la trame) peut être enclenché. Cette fenêtre de temps est caractérisée par son temps de début généralement appelé *temps d'offset*.

Configuration des modules

Une partition P déployée sur un module M peut être exécutée plusieurs fois durant l'hyper-période de M. Le nombre d'exécutions de P durant HP_M est $Instances(P) = \frac{HP_M}{P.T}$. Nous désignons par P_{ik}^M la $k^{\text{ème}}$ instance du $i^{\text{ème}}$ partition of M et par $\mathcal{P}_{inst}^{sys}$ l'ensemble de toutes les instances de partition pour un système avionique donné *sys*. Nous considérons le cas d'ordonnancement le plus général où chaque instance d'exécution durant HP_M est ordonnancée d'une manière indépendante. Ceci nous mène vers la définition suivante.

Définition 3 (Ordonnancement des Partitions) *Un ordonnancement statique d'une partition est une fonction qui associe les instances de partitions au temps d'offsets (temps d'instances)* $\psi : \mathcal{P}_{inst}^{sys} \rightarrow \mathbb{N}$.

Configuration réseau

La configuration du réseau ϕ suit le même principe que celle des modules ψ. D'une manière similaire aux partitions au sein du même module, les trames transmises sur le même lien doivent être temporellement séparées. Plusieurs instances d'une trame TT peuvent se produire durant le cycle d'application d'un système avionique *sys*, $T_{cycle}^{sys} = LCM\Big(\{f.T : f \in \mathcal{F}^{sys}\}\Big)$. Le nombre des instances d'envoi de la trame f au sein du cycle de système est $Instances(f) = \frac{T_{cycle}^{sys}}{f.T}$. À la différence d'une partition qui est allouée à un module fixe, une trame est véhiculée tout au long des chemins de données d'un lien virtuel. Nous dénotons par f_{ik}^l la $k^{\text{ème}}$ instance de f_i dans le lien l et par $\mathcal{F}_{inst}^{sys}$ l'ensemble de toutes les instances des trames d'un système avionique *sys*.

Définition 4 (Ordonnancement de trames) *Un ordonnancement déclenché par le temps (TT) d'un système avionique sys est une fonction qui permet d'associer pour toutes les instances d'une trame un temps d'offset (temps d'instance)* $\phi : \mathcal{F}_{inst}^{sys} \rightarrow \mathbb{N}$.

Intégration d'une fonctionnalité avionique

Soit *sys* un système avionique et $(\psi, \phi)_{sys}$ une configuration du système. Soit aussi *func* une nouvelle fonctionnalité à intégrer dans le système *sys*. Nous cherchons à trouver la meilleure façon d'intégrer cette nouvelle fonctionnalité. Ceci revient à synthétiser une nouvelle configuration $(\psi', \phi')_{sys'}$ du nouveau système $sys' = sys \cup func$, tout en assurant toutes les exigences des systèmes et en minimisant le coût de reconfiguration des fonctions déjà intégrées dans *sys*.

Nous notons par $I(sys,\ func)$ l'ensemble des configurations possibles du nouveau système après l'intégration de la nouvelle fonctionnalité *func*.

$$I(sys,\ func) = \{(\psi^{'}, \phi^{'}) : \psi^{'} \in \mathbb{N}^{\mathcal{P}_{inst}^{sys \cup func}} \wedge\ \phi^{'} \in \mathbb{N}^{\mathcal{F}_{inst}^{sys \cup func}}\} \tag{4.3}$$

4.3 Contraintes d'intégration de fonctionnalités avioniques communicantes en mode TT

Dans cette section, nous modélisons les exigences temporelles d'un système avionique par un ensemble de contraintes. Ses contraintes sont classées en trois catégories. La première catégorie concerne l'ordonnancement IMA et veille à ce que ce dernier soit conforme à la spécification IMA. La deuxième catégorie vérifie plutôt les exigences d'un ordonnancement TTEthernet. Pour définir cette catégorie, nous étendons l'ensemble des contraintes formalisées dans (Steiner, 2010) pour en considérer les contraintes de compatibilité des deux ordonnancements IMA et TTEthernet. Nous redéfinissons également ces contraintes en utilisant les notations de notre modèle de configuration avionique. La dernière catégorie de contraintes spécifie les exigences de latence qui ont pour but de garantir le déterminisme des délais de bout à bout.

Soit $\mathcal{G}(\mathcal{ES} \cup \mathcal{NS}, \mathcal{E})$ une grappe TTEthernet et $sys = \{(\mathcal{P}_r, \mathcal{F}_r, \alpha_r, \Omega_r, \Delta_r, \Pi_r)_{\mathcal{G}} : 1 \leq r \leq n\}$ un système avionique déployé sur $\mathcal{G}$. Afin d'alléger la formalisation des contraintes, nous utilisons les notations suivantes pour désigner :

- $\mathcal{L}_{sys}(f) \stackrel{\text{def}}{=} links(\Pi_r(f))$: l'ensemble des liens sur lesquels la trame $f \in \mathcal{F}_r$ est envoyée $(1 \leq r \leq n)$;
- $first(f) \stackrel{\text{def}}{=} first(\Pi_r(f))$: le premier lien sur lequel la trame $f \in \mathcal{F}_r$ est envoyée $(1 \leq r \leq n)$;
- $last(f_i) \stackrel{\text{def}}{=} last(\Pi_r(f_i))$: l'ensemble de derniers liens du lien virtuel de $f_i \in \mathcal{F}_r$ $(1 \leq r \leq n)$;
- $dest(f_i) \stackrel{\text{def}}{=} \Delta_r(f_i))$: l'ensemble des destinations de la trame $f_i \in \mathcal{F}_r$ $(1 \leq r \leq n)$.

Pour chaque P in $\mathcal{P}^{sys}$ et pour chaque M in $\mathcal{ES}$, on écrit aussi $P \in M$ si P est alloué à M, c'est à dire, il existe r $(1 \leq r \leq n)$ tel que $\alpha_r(P) \in M$.

4.3.1 Contraintes IMA

Afin de vérifier qu'un ordonnancement IMA est valide, nous spécifions dans cette section un ensemble de contraintes qui doivent être satisfaites par chaque ordonnancement ψ d'un système avionique $sys = \{(\mathcal{P}_r, \mathcal{F}_r, \alpha_r, \Omega_r, \Delta_r, \Pi_r)_{\mathcal{G}} : 1 \leq r \leq n\}$ déployé sur la grappe TTEthernet $\mathcal{G}(\mathcal{ES} \cup \mathcal{NS}, \mathcal{E})$.

Périodicité des partitions

Chaque partition doit être exécutée une seule fois durant sa période. La périodicité des partitions est exprimée comme suit.

$$\begin{aligned} &\forall M \in \mathcal{ES},\ \forall P_i \in M,\ \forall k \in [1..Instances(P_i)], \\ &(k-1) \times P_i.T \leq \psi(P_{ik}^M) < k \times P_i.T. \end{aligned} \qquad \text{(Périodicité\_P)}$$

Non-Chevauchement des exécution des partitions

Seulement une seule partition est autorisée à s'exécuter sur un module à un instant donné. Formellement, ceci peut être vérifié en exigeant que pour chaque deux partitions distinctes déployées sur un module, soit la première partition termine son exécution avant le début d'exécution de la deuxième ou vice versa.

$$\begin{aligned} &\forall M \in \mathcal{ES},\ \forall P_i, P_j \in M\ (i \neq j),\ \forall k \in [1..Instances(P_i)],\ \forall k' \in [1..Instances(P_j)], \\ &\Big(\Big(\psi(P_{jk'}^M) \geq \psi(P_{ik}^M) + P_i.C\Big)\ \vee\ \Big(\psi(P_{ik}^M) \geq \psi(P_{jk'}^M) + P_j.C\Big)\Big). \end{aligned}$$

(Non_Chevauchement_P)

Synchronisation des partitions

Pour les applications à sureté critique où quelques modules sont dupliqués, il est exigé dans certains cas que chaque partition est synchronisée avec sa dupliquée. Ceci signifie que les offsets pour ces deux partitions doivent être égales.

$$\begin{aligned} &\forall M_i, M_{i'} \in \mathcal{ES}\ (i \neq i'),\ \forall P_j \in M_i,\ \forall k \in [1..Instances(P_j)], \\ &Duplicate(M_i,\ M_{i'}) \Rightarrow\ \psi(P_{jk}^{M_i}) = \psi(P_{jk}^{M_{i'}}). \end{aligned} \qquad \text{(Synchronisation)}$$

où le prédicat *Duplicate*(M_i, $M_{i'}$) est évalué à vrai si M_i est le dupliqué de $M_{i'}$.

4.3.2 Contraintes TTEthernet

Nous spécifions dans cette section d'une manière formelle un ensemble de contraintes sur l'ordonnancement ϕ pour qu'il soit conforme à la spécification TTEthernet. Les contraintes définies dans les sections 4.3.2,4.3.2, 4.3.2 et 4.3.2 ont été déjà spécifié pour les réseaux déclenchés par le temps dans (Beji et al., 2014; Steiner, 2010). Nous le reformulons dans le cadre de ce travail au modèle de configuration et nous le généralisons pour les différentes instances des trames.

Contraintes de non-chevauchement

Nous considérons l'exclusion mutuelle des trames transmises sur le même lien de données. À chaque instant, une seule trame peut être transmise sur un lien donné. Étant donné un système avionique *sys* et sa configuration $(\psi, \phi)_{sys}$, on vérifie pour chaque paire de trames appartenant à *sys* et transmises sur le même lien que l'envoi de la deuxième se déroule après la fin de transmission de la première ou l'inverse. Formellement, les contraintes de non-chevauchement sont exprimées comme suit.

$$\forall f_i, f_{i'} \in \mathcal{F}^{sys},\ \forall l \in \mathcal{L}_{sys}(f_i) \cap \mathcal{L}_{sys}(f_{i'}),\ \forall k \in \Big[1..Instances(f_i)\Big],\ \forall k' \in \Big[1..Instances(f_{i'})\Big],$$
$$(f_i \neq f_{i'}) \Rightarrow \Big(\Big(\phi(f^l_{ik}) \geq \phi(f^l_{i'k'}) + f_{i'}.C\Big) \vee\ \Big(\phi(f^l_{i'k'}) \geq +\phi(f^l_{ik}) + f_i C\Big)\Big) \qquad \text{(Non\_Chevauchement\_T)}$$

Contraintes de formation correcte des chemins

La formation correcte des chemins de données peut être assurée en vérifiant l'ordre d'évènements de relai d'une trame f_i. Cet ordre doit être comme suit : (1) la réception de la trame f_i, (2) le traitement de f_i par le commutateur et (3) l'envoi de f_i sur le prochain lien. Cet ordre se traduit formellement par l'ensemble des contraintes suivantes.

$$\forall f_i \in \mathcal{F}^{sys}, \forall [u,v], [v,w] \in \mathcal{L}_{sys}(f_i), \forall k \in \Big[1..Instances(f_i)\Big],$$
$$\Big(\phi(f^{[v,w]}_{ik}) - \phi(f^{[u,v]}_{ik})\Big) \geq max(hopdelay) \qquad \text{(Dépendance\_Chemin)}$$

où $max(hopdelay)$ est un seuil maximal configurable en mode déconnecté et qui borne la latence maximale sur un saut unique du réseau. Nous notons que $max(hopdelay)$ doit être plus grand que $f_i.C$ pour tout f_i dans $\mathcal{F}^{sys}$.

Contraintes de limite de mémoire d'un commutateur

Compte tenu de la limite de la mémoire tampon et afin d'éviter les scénarios de congestion, chaque trame doit rester pour une durée limitée au niveau de chaque commutateur tout au long du lien virtuel correspondant. Nous notons cette durée limite par *membound* et nous exprimons cet ensemble de contraintes de la façon suivante.

$$\forall f_i \in \mathcal{F}^{sys}, \forall [u,v],[v,w] \in \mathcal{L}_{sys}(f_i), \forall k \in \Big[1..Instances(f_i)\Big],$$
$$\Big(\phi(f_{ik}^{[v,w]}) - \phi(f_{ik}^{[u,v]})\Big) \leq membound \qquad \text{(Limite\_Mémoire)}$$

Contraintes de relai simultanée

Quelques applications avioniques exigent que les messages soient diffusés simultanément sur tous les liens sortants. Ceci se traduit formellement par l'ensemble des contraintes suivant.

$$\forall f_i \in \mathcal{F}^{sys}, \forall [u,v],[u,w] \in \mathcal{L}_{sys}(f_i), \forall k \in \Big[1..Instances(f_i)\Big],$$
$$\Big(SimRelay(f_i)\Big) \Rightarrow \Big(\phi(f_{ik}^{[u,v]}) = \phi(f_{ik}^{[u,w]})\Big) \qquad \text{(Relai\_Simultané)}$$

où $SimRelay(f_i)$ est un prédicat qui est évalué à vrai si f_i doit être diffusée.

Compatibilité IMA-TTEthernet

Théoriquement, les ordonnancements IMA et TTEthernet ψ et ϕ peuvent être synthétisés séparément. Toutefois, de point de vue application, ces deux ordonnancements sont dépendants. En effet, les trames transmettent les informations produites par les partitions. D'où, le temps d'offset d'une instance de trame ne peut pas être configuré avant la fin de l'exécution de la partition source. La relation entre les instances de la partition source de l'information et celle de la trame qui le transporte peut être déterminée par le ratio de la division de la période de la partition source par la période de la trame. Prenant l'exemple d'une trame qui est envoyée trois fois plus rapidement que l'exécution de la partition source alors, les instances de trames de 1 à 3 communiquent le résultat produit par la première instance de la partition source. Ainsi, les temps d'offset pour les trois premières instances de trame doivent être entre la fin d'exécution de la première et la seconde instance de la partition source. Nous notons aussi que la période de la partition source P doit être plus grande ou égale aux périodes des trames produites par P pour éviter la congestion au niveau du nœud source. Soit k une instance de la partition source P et k' une instance de la trame transportant le

message produit par la $k^{\text{ème}}$ instance de P, alors $k^{'}$ doit être dans l'intervalle

$$\Big[\Big((k-1)\times\frac{source(f_i).T}{f_i.T}+1\Big)..k\times\frac{source(f_i).T}{f_i.T}\Big].$$

Ainsi la compatibilité des deux ordonnancements ψ et ϕ peut être exprimée comme suit.

$$\begin{aligned}
&\forall f_i\in\mathcal{F}^{sys},\forall k\in\Big[1..Instances(source(f_i))\Big],\\
&\forall k^{'}\in\Big[\Big((k-1)\times\frac{source(f_i).T}{f_i.T}+1\Big)..k\times\frac{source(f_i).T}{f_i.T}\Big],\\
&\Big(\psi(source(f_i)_k)+source(f_i).C\leq\phi(f_{ik'}^{first(f_i)})\Big)\wedge\\
&\Big(\phi(f_{ik'}^{first(f_i)})<\psi(source(f_i)_{k+1})+source(f_i).C\Big)\qquad\text{(Compatibilité_IMA_TTE)}
\end{aligned}$$

où $source(f_i)_k$ désigne $k^{\text{ème}}$ instance de trame de $source(f_i)$.

4.3.3 Contraintes de latence

Afin d'assurer que les fonctions avioniques sont complètement terminées dans les limites de temps, nous définissons deux types de contraintes de latence. Ces deux types sont illustrés par la figure 4.3.

Le premier type, connue sous le nom de *latence de bout-en-bout*, permet de contraindre la durée maximale écoulée depuis le début de l'exécution de la partition source jusqu'à la fin d'exécution de la partition destination. Les contraintes de latence de bout-en-bout, assurent

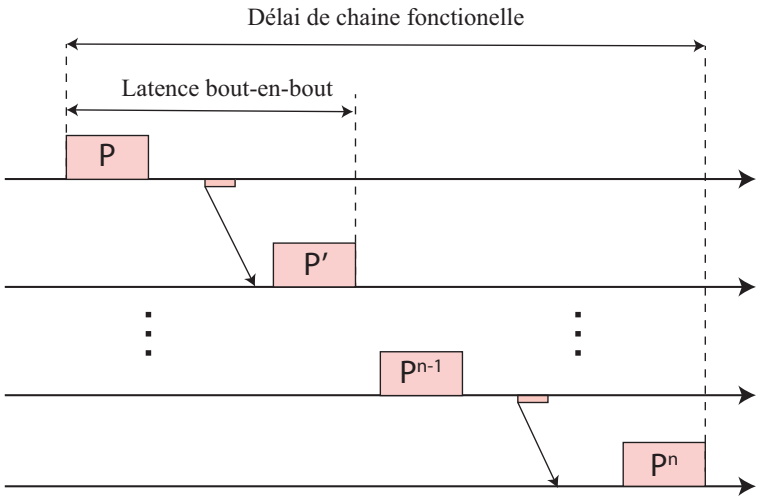


Figure 4.3 Latence de bout-en-bout et délai de chaîne fonctionnelle

l'ordonnancabilité des trames du système. En effet, elles garantissent que chaque trame peut atteindre sa destination dans les délais impartis.

Le deuxième type appelé *délai d'une chaîne fonctionnelle* a pour but de contraindre le temps global de réponse d'une fonction avionique. Un exemple de délai de bout en bout d'une chaîne fonctionnelle serait le processus d'extension/rétraction du train d'atterrissage qui fait intervenir plusieurs trames.

Latence de bout-en-bout

Les éléments suivants sont nécessaires pour calculer la latence : l'instance d'exécution de la partition émettrice ; l'instance de la trame transportant le message communiqué ; et l'instance de la partition réceptrice.

Pour chaque destination P_d^v dans $dest(f_i)$, la latence de communication entre $source(f_i)$ et P_d^v est périodique de période $LCM\,(source(f_i).T, f_i.T, P_d^v.T)$. Il est ainsi suffisant de considérer les instances de a de la partition source dans l'intervalle suivant

$$\left[1..\frac{LCM\,(source(f_i).T, f_i.T, P_d^v.T)}{source(f_i).T}\right].$$

Le calcul de latence dépend aussi de l'instance b de communication de la trame f_i et l'instance c de la partition destination P_d^v. Si la $b^{\text{ème}}$ instance de la trame f_i transmet le message produit par la $a^{\text{ème}}$ instance de $source(f_i)$, alors cette trame doit être acheminée entre la $a^{\text{ème}}$ et la $(a+1)^{\text{ème}}$ instance de la partition $source(f_i)$. De la même façon, si la $c^{\text{ème}}$ instance de partition de P_d^v consomme le message transmis par la $b^{\text{ème}}$ instance de trame de f_i, alors elle doit être enclenché entre la réception de la $b^{\text{ème}}$ et la $(b+1)^{\text{ème}}$ instance de trame de f_i.

Soit le système avionique $sys = \{(\mathcal{P}_k, \mathcal{F}_k, \alpha_k, \Omega_k, \Delta_k, \Pi_k)_{\mathcal{G}} : 1 \leq k \leq n\}$ et $(\psi, \phi)_{sys}$ une configuration de sys, nous formalisons la latence de bout-en-bout comme suit :

$$
\begin{aligned}
&\forall f_i \in \mathcal{F}^{sys},\ \forall [u,v] \in last(f_i),\ \forall P_d^v \in dest(f_i),\\
&\forall a \in \left[1..\frac{LCM\left(source(f_i).T, f_i.T, P_d^v.T\right)}{source(f_i).T}\right],\\
&\forall b \in \left[1..\frac{LCM\left(source(f_i).T, f_i.T, P_d^v.T\right)}{f_i.T}\right],\\
&\forall c \in \left[1..\frac{LCM\left(source(f_i).T, f_i.T, P_d^v.T\right)}{P_d^v.T}\right],\\
&\Bigg(\Big(\psi(source(f_i)_{a \mod Instances(source(f_i))}) + source(f_i).C \leq \phi(f^{first(f_i)}_{i(b \mod Instances(f_i))})\Big) \wedge\\
&\Big(\phi(f^{first(f_i)}_{i(b \mod Instances(f_i))}) < \psi(source(f_i)_{a \mod Instances(source(f_i))+1}) + source(f_i).C\Big) \wedge\\
&\Big(\phi(f^{[u,v]}_{i(b \mod Instances(f_i))}) + f_i.C \leq \psi(P^v_{d(c \mod Instances(d_s))})\Big) \wedge\\
&\Big(\psi(P^v_{d(c \mod Instances(d_s))}) < \phi(f^{[u,v]}_{i(b \mod Instances(f_i))+1}) + f_i.C\Big)\Bigg) \Rightarrow\\
&\Big(\Big(\psi(P^v_{d(c \mod Instances(d_s))}) + P_d^v.C - \psi(source(f_i)_{a \mod Instances(source(f_i))})\Big) \leq maxLatency(f_i)\Big)
\end{aligned}
$$

(Latence_Bout)

Nous notons que nous ne considérons que le modulo car ψ est seulement défini pour $a \leq Instances(source(f_i))$ et $c \leq Instances(P_d^v)$ et ϕ est seulement défini pour $b \leq Instances(f_i)$. La première ligne de la contrainte vérifie que la fin de l'exécution de $source(f_i)$ est avant le début de transmission de f_i sur son premier lien. La deuxième ligne vérifie plutôt que le début de transmission de f_i sur son premier lien est bien avant la fin d'exécution de la prochaine instance de $source(f_i)$. D'une manière similaire, nous vérifions dans la troisième ligne que la fin de transmission de la trame sur le dernier lien est avant le début d'exécution de l'instance de la partition destination P_d^v. Nous vérifions également dans la quatrième ligne que l'arrivé de la prochaine instance de trame est bien après le début d'exécution de la même instance de partition destination. Ces quatre lignes permettent ainsi d'identifier l'instance de la source, l'instance de la trame communiquée et la l'instance de la partition destination. Il reste donc à borner dans la cinquième ligne de cette contrainte la durée du début d'exécution de l'instance de partition source à la fin d'exécution de l'instance de la partition destination.

Délai d'une chaîne fonctionnelle

Une autre propriété importante de latence des systèmes avioniques est le *délai d'une chaîne fonctionnelle* qui fait intervenir plusieurs systèmes terminaux communicants. Outre que l'ordonnançabilité de chaque trame d'une chaîne fonctionnelle, il est exigé que le délai bout-en-bout de toute la chaîne fonctionnelle soit borné par un seuil fixé qu'on le note *maxDelay*. À

titre exemple, nous citons le délai de bout-en-bout du processus d'extension du système du train d'atterrissage qui est défini d'une manière informelle comme suit :

> Quand les lignes de commandes sont fonctionnelles (c.-à-d. absence d'échecs), si la poignée de la commande du train d'atterrissage a été poussée vers le bas et reste vers le bas alors les trains doivent être verrouillés vers le bas et les portes sont fermés en moins de 15 secondes après que les commandes sont actionnées (Boniol and Wiels, 2014).

Nous spécifions la latence de toute la chaîne en bornant la somme des pires cas délais (Worst-Case Delay (WCD)) des trames intervenant dans la chaîne fonctionnelle. Le pire cas délai pour une configuration $(\psi, \phi)_{sys}$ est défini comme suit :

$$WCD(f_i, \psi, \phi) \stackrel{\text{def}}{=} \max_{[u,v] \in last(f_i), P_d^v \in dest(f_i), a, c} \Big(\psi(P^v_{d(c \mod Instances(d_s))}) + P_d^v.C - \psi(source(f_i)_{a \mod Instances(source(f_i))}) \Big) \tag{4.4}$$

avec a et c satisfont les même conditions comme le cas des contraintes d'ordonnançabilité.

Soit $sys = \{(\mathcal{P}_k, \mathcal{F}_k, \alpha_k, \Omega_k, \Delta_k, \Pi_k)_{\mathcal{G}} : 1 \leq k \leq n\}$ un système avionique et $(\psi, \phi)_{sys}$ une configuration de *sys*. Pour chaque fonctionnalité *func* de *sys*, la chaîne fonctionnelle en intérêt est donnée par l'ingénieur système et spécifiée par l'ensemble $\mu(func)$. Nous spécifions l'exigence de latence de bout-en-bout pour *sys* comme suit :

$$\forall func \in sys,\ \forall [f_{i_1}, f_{i_2}, .., f_{i_n}] \in \mu(func),$$
$$\sum_{k=1}^{n} WCD(f_{i_k}, \psi, \phi) \leq maxDelay([f_{i_1}, f_{i_2}, .., f_{i_n}]) \qquad \text{(Delai\_Chaine)}$$

Compte tenu de l'ensemble des contraintes définies dans cette section, on caractérise dans la prochaine section notre approche SMT pour l'intégration optimale des systèmes avioniques.

4.4 Approche SMT pour l'intégration optimale de système avionique

Nous définissons dans cette section notre approche Weighted Partial Max-SMT (WPMS) pour l'intégration optimale des systèmes avioniques. Pour ce faire, nous caractérisons formellement le problème d'intégration itérative. Cette caractérisation va nous permettre par la suite de définir notre algorithme WPMS pour la résolution de ce problème.

4.4.1 Caractérisation de l'approche d'intégration itérative

Dans le cadre du problème d'intégration itérative, le coût qu'on considère est seulement relatif aux trames et aux partitions des fonctionnalités déjà intégrées. Soit $sys = \{(\mathcal{P}_k, \mathcal{F}_k, \alpha_k, \Omega_k, \Delta_k, \Pi_k)_{\mathcal{G}} : 1 \leq k \leq n\}$ un système avionique, $(\psi, \phi)_{sys}$ sa configuration courante, et $func = (\mathcal{P}_{n+1}, \mathcal{F}_{n+1}, \alpha_{n+1}, \Omega_{n+1}, \Delta_{n+1}, \Pi_{n+1})_{\mathcal{G}}$ une nouvelle fonctionnalité à intégrer. Soit $sys' = sys \cup func$ le système après l'intégration de $func$ et $(\psi^{'}, \phi^{'})_{sys'} \in I(sys, func)$ une nouvelle configuration de sys'. Bien évidemment, nous supposons que la configuration courante $(\psi, \phi)_{sys}$ est valide, c'est à dire, elle satisfait toutes les contraintes dures de la section 4.3. Nous avons choisi de définir une approche WPMS pour résoudre le problème d'intégration itérative vu que la modification des offsets d' une instance de trame ou bien d'une partition déjà ordonnancée, peut être modélisée par des contraintes douces. Il reste donc à faire la correspondance entre le coût de violation d'une contrainte et le coût de reconfiguration de l'instance (de partition ou de trame) considérée.

1. *Fonction Coût :* La fonction coût d'intégration à partir de la configuration courante $(\psi, \phi)_{sys}$ à une nouvelle configuration $(\psi', \phi')_{sys'}$ est :

$$Cost((\psi, \phi), (\psi^{'}, \phi^{'})) = \sum_{\substack{M \in \mathcal{ES} \\ P_i \in M \cap \mathcal{P}^{sys} \\ k \in [1...Instances(P_i)]}} \omega(P_{ik}^M) \times \delta_{i,k}^M(\psi', \psi) + \sum_{\substack{f_i \in \mathcal{F}^{sys} \\ l \in \mathcal{L}^{sys}(f_i) \\ k \in [1..Instances(f_i)]}} \kappa(f_{ik}^l) \times \sigma_{i,k}^l(\phi', \phi) \quad (4.5)$$

   où

   — $\omega(P_{ik}^M)$ est le coût de re-ordonnancer l'instance de partition P_{ik}^M.

   — $\kappa(f_{ik}^l)$ est le coût de re-ordonnancer l'instance de trame f_{ik}^l.

   — $\delta_{i,k}^M((\psi', \psi)$ est la fonction indicateur retournant 1 si est seulement si la contrainte douce $(\psi'(P_{ik}^M) = \psi(P_{ik}^M))$ est violée.

   — $\sigma_{i,k}^l((\phi', \phi)$ est la fonction indicateur retournant 1 si est seulement si la contrainte douce $(\phi'(f_{ik}^l) = \phi(f_{ik}^l))$ est violée.

2. *Objectif :* Notre objectif est de minimiser la fonction coût. La formulation globale du problème d'intégration itérative est comme suit.

minimiser : $Cost((\psi,\phi),(\psi^{'},\phi^{'})),$ with $(\psi^{'},\phi^{'}) \in I(sys,func)$

sous contraintes de :

$$\bigwedge_{\substack{M\in\mathcal{ES}\\ P_i\in M\cap\mathcal{P}^{sys}\\ k\in[1..Instances(P_i)]}} \Big(\delta^M_{i,k}(\psi',\psi) \iff \psi'(P^M_{ik}) \neq \psi(P^M_{ik})\Big)\wedge$$

$$\bigwedge_{\substack{f_i\in\mathcal{F}^{sys}\\ l\in\mathcal{L}^{sys}(f_i)\\ k\in[1..Instances(f_i)]}} \Big(\sigma^l_{i,k}(\phi',\phi) \iff \phi'(f^l_{ik}) \neq \phi(f^l_{ik})\Big) \wedge \mathcal{C}_{hard}(\psi',\phi') \qquad \text{(P)}$$

où $\mathcal{C}_{hard}(\psi',\phi')$ est la conjonction de toutes les contraintes dures définies dans la section 4.3 pour la configuration $(\psi^{'},\phi^{'})_{sys'}$ du nouveau système après intégration. Nous désignons par $Cost(I(sys,func),(\psi,\phi)_{sys})$ le coût d'intégrer la fonction *func* au système *sys* avec la configuration courante (ψ,ϕ).

Maintenant, étant donné un système initial sys_0, sa configuration courante (ψ_0,ϕ_0) et un ensemble fini de fonctions avioniques $\{func_1, func_2, \cdots, func_n\}$, nous définissons le coût d'une intégration itérative des fonctions $func_i$ par la somme des coûts des étapes d'intégration. Ceci peut être formalisé comme suit :

$$Cost(I(sys_0,func_1,...,func_n),(\psi_0,\phi_0)) = \sum_{i=1}^{n} Cost(I(sys_{i-1},func_i),(\psi_{i-1},\phi_{i-1})) \qquad (4.6)$$

où $sys_i = sys_{i-1} \cup func_i$, et (ψ_{i-1},ϕ_{i-1}) est la configuration obtenue en intégrant $func_{i-1}$.

4.4.2 Approche Max-SMT pondérée et partielle pour la reconfiguration Weighted Partial Max-SMT-WPMS

En se basant sur les contraintes dures définies dans la section 4.3 et les contraintes douces définies ci-dessous, notre approche d'intégration présentée par l'algorithme 1, génère les assertions et leurs poids pour le contexte logique du solveur. Les variables des formules sont les paramètres d'ordonnancement des instances des partitions et des trames. Si le contexte est satisfiable, c'est-à-dire toutes les contraintes dures sont satisfiables, alors l'ensemble des valeurs des variables, *modèle de solution* et le poids total des contraintes non satisfaites est retourné.

L'algorithme prend comme paramètres d'entrée l'architecture physique $\mathcal{G}(\mathcal{V}, \mathcal{L})$, l'ensemble de toutes les fonctions avioniques déjà intégrées *sys*, leurs configurations actuelles $(\psi, \phi)_{sys}$, la nouvelle fonction à intégrer *func*, l'ensemble des chaînes fonctionnelles concernées par les contraintes de latence μ et les fonctions poids ω et κ. L'algorithme commence par ajouter les contraintes dures $\mathcal{C}_{hard}$ pour le nouveau système sys' (qui combine l'ancien système *sys* avec la nouvelle fonction *func*) au solveur de contexte ($AssertHard$). Si l'ancien système est vide (première intégration) alors le solveur SMT ($SMTSolve$) est invoqué et un modèle , qui est une configuration $(\psi^{'}, \phi^{'})_{sys'}$ est retourné. Autrement, l'algorithme ajoute aussi les clauses pondérées des contraintes douces

$$\mathcal{C}_{soft} = \{(c_1, \theta_1), (c_2, \theta_2), \cdots, (c_s, \theta_s)\} \tag{4.7}$$

au contexte du solveur et invoque le solveur ($MaxSMTSolve$). Dans ce cas, une solution (si elle existe) est la configuration $(\psi^{'}, \phi^{'})_{sys'}$ du nouveau système qui minimise le coût de reconfigurer les paramètres d'ordonnancement.

ALGORITHM 1: Approche d'intégration itérative basé sur le formalisme Max-SMT

Input: $\mathcal{G}(\mathcal{V}, \mathcal{L}), sys, (\psi, \phi)_{sys}, func, \mu, \omega, \kappa$
Output: $(\psi^{'}, \phi^{'}), cost$
Begin
  $cost \leftarrow 0$
  $sys' \leftarrow sys \cup func$
  $\mathcal{C}_{hard} \leftarrow AssertHard(\mathcal{G}(\mathcal{V}, \mathcal{L}), sys', \mu)$
  if $sys \neq \emptyset$ **then**
    $\mathcal{C}_{soft} \leftarrow AssertSoft(\mathcal{G}(\mathcal{V}, \mathcal{L}), sys, (\psi, \phi)_{sys}, sys', \omega, \kappa)$
    $[(\psi^{'}, \phi^{'}), cost] \leftarrow MaxSMTSolve(\mathcal{C}_{hard}, \mathcal{C}_{soft})$
  else
    $(\psi^{'}, \phi^{'}) \leftarrow SMTSolve(\mathcal{C}_{hard})$
  fi
return $[(\psi^{'}, \phi^{'}), cost]$

4.5 Cas d'études

Nous analysons dans cette section à travers deux cas d'étude notre approche d'intégration itérative. Le premier cas d'étude illustre un système de gestion de vol alors que le deuxième illustre d'autres fonctionnalités avioniques à travers l'exemple du système de contrôle de carburant et le système de train d'atterrissage.

4.5.1 Premier cas étude : Système de gestion de vol

Présentation

Le système de gestion de vol contrôle l'affichage des informations statiques de navigation dans les écrans des pilotes. Son fonctionnement est décrit dans la section 4.1.2 et dont l'architecture est donnée par la figure 4.2. Nous considérons pour ce système la chaîne fonctionnelle donnée par la figure 4.4 et les trames spécifiées par le tableau 4.2.

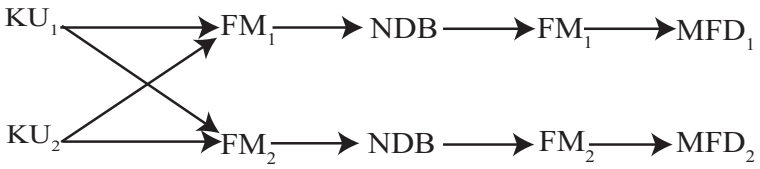


Figure 4.4 Chaîne fonctionnelle du système de gestion de vol

L'intégration des éléments du sous-système illustré par la figure 4.2 est effectuée en trois étapes. Nous intégrons en premier lieu les partitions KU_1, FM_1 et NDB et la trames f_1 sur les liens de données $\left[[M_1\text{-}SW_1], [SW_1\text{-}M_3]\right]$ et f_2. Dans une deuxième étape, nous intégrons les parties dupliquées de celles de l'étape 1. Plus précisément, nous intégrons les partitions KU_2 et FM_2 et les trames f_1 sur les liens $\left[[M_1\text{-}SW_1], [SW_1\text{-}M_4]\right]$, ainsi que les trames f_3 et f_4. Finalement, la dernière étape intègre les parties manquantes soit les partitions MFD_1 et MFD_2 et les trames f_5, f_6 et f_7.

Les résultats présentés ci-dessous sont obtenus avec l'instance suivante du système de gestion de vol. Les paramètres temporaux des partitions et leurs coûts de reconfiguration associés sont donnés par le tableau 4.3. Nous utilisons les mêmes paramètres *Période=50*, *Longueur=3* et le même coût de reconfiguration (*Coût=7*) pour toutes les trames. Le délai de traitement de chaque trame par un commutateur est fixé à 1.

Dans la première étape d'intégration, nous notons que les contraintes dures sont trivialement satisfaites excepté les suivantes : contraintes *périodicité des partitions* pour l'ordonnancement IMA, contraintes *formation correcte des chemins*, contraintes *limite de mémoire d'un commutateur* pour l'ordonnancement TTEthernet, les contraintes de *compatibilité IMA-TTE* et les contraintes de *latence de bout-en-bout* pour la communication de M_1 à M_5. Cette latence doit être inférieur à un seuil $lat1$. Vu que rien n'a été initialement intégré, nous notons qu'à cette étape nous n'avons pas de contraintes douces et que le coût d'intégration est nul.

Tableau 4.2 Les trames du système de gestion de vol

Trame	Lien
f_1	[M_1-SW_1]
f_1	[SW_1-M_3], [SW_1-M_4]
f_2	[M_3-SW_2]
f_2	[SW_2-M_5]
f_3	[M_1-SW_1]
f_3	[SW_1-M_3], [SW_1-M_4]
f_4	[M_3-SW_2]
f_4	[SW_2-M_5]
f_5	[M_2-SW_2]
f_5	[SW_2-M_3], [SW_2-M_4]
f_6	[M_3-SW_1]
f_6	[SW_1-M_1]
f_7	[M_4-SW_1]
f_7	[SW_1-M_2]

Tableau 4.3 Paramètres d'ordonnancement des partitions

Module	Partition	Période	Longueur	Coût
M_1	KU_1	50	25	1
M_1	MFD_1	50	25	3
M_2	KU_2	50	25	1
M_2	MFD_2	50	25	3
M_3	FM_1	60	30	5
M_4	FM_2	60	30	5
M_5	NDB	100	20	5

Dans cette étape d'intégration, les choses commencent à être de plus en plus intéressantes. Outre que les contraintes non-triviales de l'étape précédente, nous devons considérer aussi les contraintes de *relai simultané* des trames communiquées à FM_1 et FM_2, les contraintes de *non-chevauchement* sur les liens [SW_1-M_3], [SW_1-M_4], [SW_2-M_5] et la contrainte de *minimisation du coût d'intégration* pour les parties déjà intégrées dans la première étape.

Finalement, la dernière étape ajoute les contraintes de *non-chevauchement des exécutions* des partitions sur M_1 et M_2 et un deuxième seuil de latence $lat2$ pour borner la réponse de M_5 aux deux modules M_1 et M_2.

Analyse des résultats

Nous analysons dans cette partie les résultats obtenus suite à l'intégration de l'instance du système FMS décrit dans la section précédente.

Analyse des performances Nous avons implémenté les contraintes spécifiées dans la section 4.3 pour cette instance de système en utilisant le solveur SMT *YICES*. En fixant les paramètres de latence $lat1$ et $lat2$ égaux à 300, nous avons eu automatiquement les paramètres d'ordonnancement pour les partitions et les trames TT reportées respectivement par les tableaux 4.4 et 4.5. Chaque étape d'intégration retourne les résultats dans les 100 ms. Le coût total de l'intégration est zéro, c'est-à-dire qu'aucune étape d'intégration ne modifie les valeurs d'offset des parties déjà intégrés. Nous notons que le processus d'intégration nécessite environ 120 contraintes et environ mille lignes de code.

Performance vs Coût d'intégration Nous relevons que, avec un seuil de 300, le coût d'intégration est nul. Par conséquent, nous avons déterminé le seuil de latence le plus bas et le plus haut pour avoir une intégration faisable avec un coût non nul. Le seuil de latence le plus bas est 184 et le plus haut est 250. L'impact du seuil sur le coût d'intégration est donné par la figure 4.5. Comme attendu, cette figure montre que le coût d'intégration augmente quand le seuil de latence diminue. Nous notons aussi que pour certains intervalles (ex. [136..149]), le coût est constant. Il est donc plus approprié de choisir le seuil le plus bas pour cet intervalle soit 136 pour optimiser la latence avec le même coût. Cette étude peut aider à faire une balance entre la performance et le budget (c-à-d. coût).

Coût d'intégration vs périodicité des trames TT Finalement, nous constatons que quand les partitions sources ont une périodicité différente que leurs trames associées alors le coût d'intégration augmente. Ainsi, s'il n'est pas exigé par l'application, il est mieux d'avoir la

Tableau 4.4 Offsets des Partitions

Module	Partition	Offset
M_1	KU_1	0
M_1	MFD_1	25
M_2	KU_2	0
M_2	MFD_2	25
M_3	FM_1	30
M_4	FM_2	30
M_5	NDB	75

Tableau 4.5 Offsets des trames

Trame	Lien	Offset
f_1	$[M_1\text{-}SW_1]$	25
f_1	$[SW_1\text{-}M_3]$, $[SW_1\text{-}M_4]$	37
f_2	$[M_3\text{-}SW_2]$	109
f_2	$[SW_2\text{-}M_5]$	122
f_3	$[M_1\text{-}SW_1]$	37
f_3	$[SW_1\text{-}M_3]$, $[SW_1\text{-}M_4]$	41
f_4	$[M_3\text{-}SW_2]$	109
f_4	$[SW_2\text{-}M_5]$	126
f_5	$[M_2\text{-}SW_2]$	100
f_5	$[SW_2\text{-}M_3]$, $[SW_2\text{-}M_4]$	104
f_6	$[M_3\text{-}SW_1]$	109
f_6	$[SW_1\text{-}M_1]$	122
f_7	$[M_4\text{-}SW_1]$	60
f_7	$[SW_1\text{-}M_2]$	64

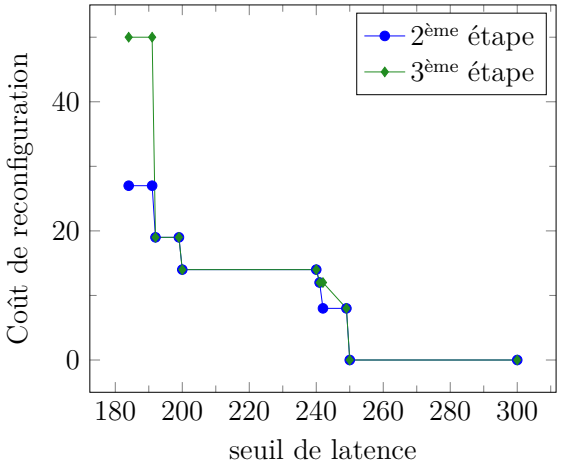


Figure 4.5 Le coût d'intégration en fonction du seuil de latence

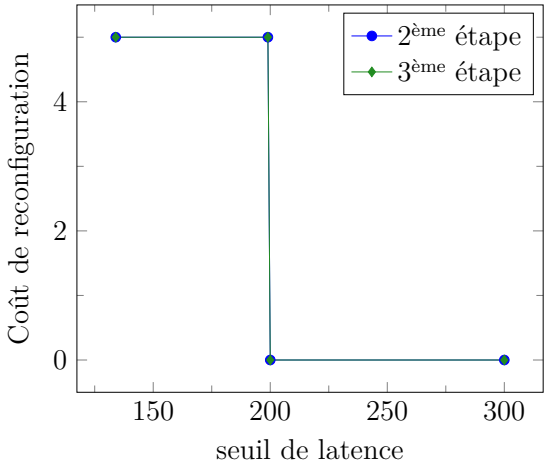


Figure 4.6 Coût d'intégration avec des périodicités des trames égale à 100

même périodicité pour une partition source avec ces trames associées. Ce résultat est illustré par les figures 4.6, 4.7 et 4.8 où la périodicité de KU_1 et KU_2 est fixé à 100.

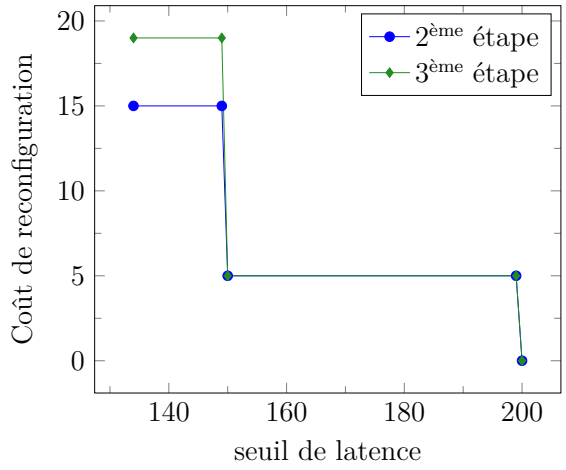


Figure 4.7 Coût d'intégration avec des périodicités des trames égale à 50

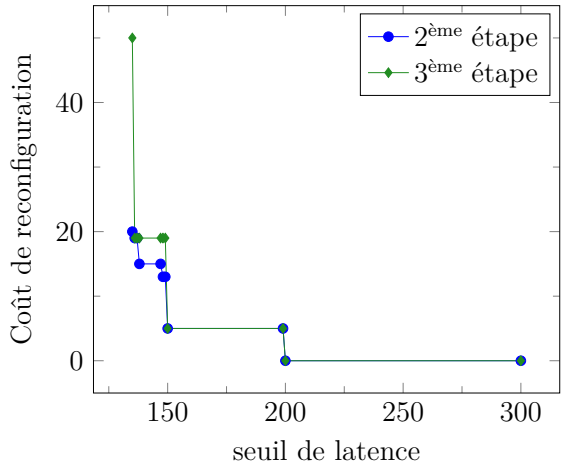


Figure 4.8 Coût d'intégration avec des périodicités des trames égale à 25

4.5.2 Deuxième cas d'étude : Système du train d'atterrissage et Système de contrôle de carburant

Nous étudions dans cette section l'intégration itérative de deux autres sous-systèmes avioniques : le *système de train d'atterrissage* et le *système du contrôle de carburant*. Ces deux sous-systèmes sont conçus conformément à la spécification IMA et communiquent avec des flux TT. Nous commençons par une briève description des deux sous-systèmes à intégrer. Par la suite, nous décrivons l'architecture et les applications. Nous concluons cette section par la présentation de nos résultats d'intégration et quelques observations.

Description du système

Système de train d'atterrissage Le système du train d'atterrissage est un système avionique critique qui a deux rôles cruciaux dans le sol. Il supporte le poids total et la charge de l'avion et donne aussi la mobilité de l'avion en mode taxi. Il est composé par trois parties : système avionique et hydraulique, système numérique et interface cockpit.

Pour la partie mécanique, elle est composée de trois ensembles d'atterrissage : un ensemble d'atterrissage avant (nose landing set) localisé dans le nez de l'avion et deux trains principaux (main landing set) situés dans les côtés gauches et droite. La partie numérique est un logiciel de contrôle qui fournit les commandes pour l'extension/rétraction des actionneurs hydrauliques en causant le mouvement des ensembles d'atterrissage.

Les trains avant et principaux sont commandés séparément et respectivement par les composants logiciels $NoiseLandingGear(NLG)$ et $MainLandingGear(MLG)$

Système de contrôle de carburant Le système de contrôle de carburant surveille la consommation et le transfert du carburant dans l'avion. Il est composé par trois parties : les réservoirs de carburant, le système numérique et l'interface cockpit.

Des capteurs sont installés dans les réservoirs donnent des mesures sur le niveau du carburant et d'autres caractéristiques physiques comme la température, la densité et la permittivité. Les systèmes numériques sont composés par trois sortes de composants logiciels. Le premier composant est le *Gauging Channels* GCA et GCB. Ils donnent les mesures nécessaires pour le système. Ces mesures sont utilisées par le *Fuel Mass* (FM) pour calculer le volume et la masse du carburant dans chaque réservoir. Le troisième composant, le *Fuel Transfer Component* (FTC) enclenche le transfert du carburant pour alimenter les engins gauche et droit de l'avion et gardent la position latérale du centre de gravité dans la ligne centrale de l'avion.

Les actions du pilote à travers l'interface du cockpit sont émulées avec un composant logiciel : le *Flight Deck Emulator* (FDE). Les capteurs du système de train d'atterrissage et du système de contrôle du carburant sont aussi émulés par le *Landing Gear and Fuel Aircraft Emulator* (LGFAE).

Description de l'architecture et de l'application

L'architecture physique de ce cas d'étude est illustrée par la figure 4.9. Elle est composée de 6 modules (M_1 à M_6) hébergeant les composants logiciels (partitions IMA) décrits ci-dessous. Nous supposons que le système initial contient déjà les partitions (GUI, $SP1$, SPE et $Handover$) et le FDE.

Nous intégrons les fonctionnalités du train d'atterrissage et du contrôle du carburant décrits dans la section 4.5.2. La fonctionnalité du train d'atterrissage est représentée par les partitions en couleur bleue (NLG et MLG). Il partage les partitions émulateurs FDE et $LGFAE$ qui sont utilisés aussi par le système de contrôle du carburant illustré en rouge (FM, GCA, GCB, et FTC).

Les paramètres temporaux fixes des partitions ci-dessus sont donnés par le tableau 4.10 tandis que celles des communications sont résumées par le tableau 4.6 où les liens de données sont identifiés par des nombres de 1 à 12. L'identification de ses liens est donné par le tableau 4.7.

Expériences et résultats d'intégration

Stratégies d'intégration Nous voulons identifier la meilleure stratégie pour intégrer le système du train d'atterrissage et le système de contrôle de carburant. La meilleure stratégie dans notre cas est l'ordre d'intégration qui garantit le coût le plus faible. Dans une première

Tableau 4.6 Caractéristiques temporelles des trames

$func_k$	*frame*	*source*(f)	*dest*(f)	$f.T$	$f.C$	*links*(f)
1	101	GUI	FDE	60	1	9-4
	102	FDE	GUI	30	1	3-10
	103	$Handover$	everywhere	60	11	11-{2,4,6,8,10}
	104	$Handover$	everywhere	60	5	11-{2,4,6,8,10}
	105	$Handover$	everywhere	60	2	11-{2,4,6,8,10}
	106	SPE	$SP1$	60	4	3-6
	107	$SP1$	SPE	60	1	5-4
2	108	FDE	NLG	30	1	3-2
	109	FDE	MLG	30	1	3-8
	110	$LGFAE$	NLG	60	2	3-2
	111	$LGFAE$	MLG	60	4	3-8
	112	NLG	$LGFAE$	60	1	1-4
	113	MLG	$LGFAE$	60	3	7-4
	114	NLG	FDE	60	2	1-4
	115	MLG	FDE	60	1	7-4
3	116	FDE	FTC	30	2	3-8
	117	$LGFAE$	GCA	60	3	3-2
	118	$LGFAE$	GCB	60	7	3-8
	119	$LGFAE$	FM	60	3	3-8
	120	GCB	FM	30	4	7-2
	121	FM	FTC	60	1	1-8
	122	FTC	$LGFAE$	60	2	7-4
	123	GCA	FDE	30	1	1-4
	124	GCB	FDE	30	2	7-4
	125	FM	FDE	60	1	1-4
	126	FTC	FDE	60	1	7-4

Tableau 4.7 Identification des liens de données

Id	Lien de donnée	Id	Lien de donnée
1	[M_1,SW]	7	[M_2,SW]
2	[SW,M_1]	8	[SW,M_2]
3	[M_3,SW]	9	[M_4,SW]
4	[SW,M_3]	10	[SW,M_4]
5	[M_5,SW]	11	[M_6,SW]
6	[SW,M_5]	12	[SW,M_6]

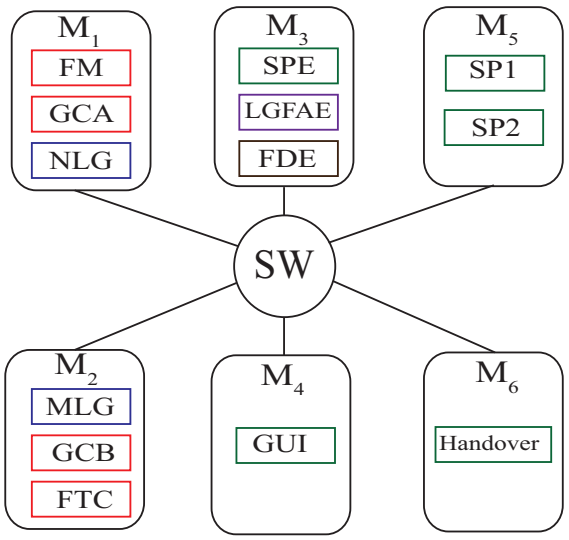


Figure 4.9 Système Avionique : Architecture Physique

Partition	Period	Length	Module
FM	120	10	M_1
GCA	30	5	
NLG	120	10	
MLG	120	10	M_2
GCB	30	5	
FTC	120	10	
LGFAE	120	10	M_3
FDE	30	5	
SPE	60	7	
GUI	60	7	M_4
SP1	60	7	M_5
SP2	60	7	
Handover	60	7	M_6

Figure 4.10 Caractéristiques temporelles des partitions

expérimentation, nous fixons le coût de reconfiguration de chaque instance de partition ou de trame à une unité. La latence de bout-en-bout de chaque trame est égale à sa période. Nous avons considéré aussi les chaînes fonctionnelles suivantes :

1. Le temps de réponse du moment de l'activation du processus d'extension/rétraction du train d'atterrissage avant par le pilote jusqu'à l'actualisation de la nouvelle position affichée sur l'interface du cockpit. Ce temps de réponse est équivalent à étudier le pire cas de latence de la chaîne fonctionnelle $FDE \rightarrow NLG \rightarrow FDE$. Nous dénotons aussi par $lat1$ le seuil fixé pour le pire cas
2. Le temps de réponse du moment de l'activation du transfert de carburant entre les deux réservoirs jusqu'à la confirmation du transfert affiché au pilote. Ce temps de réponse est équivalent à étudier le pire cas de latence de la chaîne fonctionnelle $FDE \rightarrow FTC \rightarrow FDE$. Nous dénotons par $lat3$ le seuil fixé pour ce pire cas de latence

Nous avons commencé par l'intégration du système de contrôle de carburant avant celui du train d'atterrissage. Le coût d'intégration du système du contrôle de carburant est donné par le tableau 4.8 alors que le coût de la deuxième itération est donné par le tableau 4.9. Nous écrivons $[a..b]$ pour l'intervalle de a à b. Par exemple, dans le tableau 4.8, quant nous bornons le pire cas de latence pour la chaîne fonctionnelle du transfert de carburant par un seuil dans l'intervalle [78..104], alors le coût d'intégration du système de transfert de carburant reste à 3 unités.

Maintenant, nous inversons l'ordre d'intégration en commençant par le système du train d'at-

Tableau 4.8 Coût d'intégration du système de contrôle de carburant

$lat3$	74	75	[76..77]	[78..104]	[105..150]
Coût	∞	6	5	3	0

Tableau 4.9 Coût d'intégration du système de transfert de carburant

$lat1$ \ $lat3$	[75..134]	[135..170]
[75..170]	5	2

terrissage avant le système de contrôle de carburant. Les coûts sont illustrés respectivement par les tableaux 4.10 et 4.11 pour la première et la deuxième étape.

Tableau 4.10 Coûts d'intégration du système de train d'atterrissage

$lat1$	74	[75..76]	77	[78..104]	[105..170]
Coût	∞	6	5	4	0

Tableau 4.11 Coûts d'intégration du système de contrôle de carburant après le système du train d'atterrissage

$lat3/lat1$	[75..170]
[75..134]	5
[135..170]	3

Nous notons que dans les deux cas, plus la performance exigée est plus grande (c-à-d. seuil de latence est plus bas) plus le coût d'intégration est élevé. En d'autres termes, il est nécessaire de trouver le bon seuil pour la performance et le coût. Autrement dit, quand nous intégrons une nouvelle fonctionnalité avionique sur un système déjà existant, nous avons besoin d'équilibrer soigneusement la performance globale que nous voulons obtenir et le budget à notre disposition pour la reconfiguration/recertification. Nous notons aussi que la meilleure stratégie d'intégration dépend aussi du niveau de performance requis pour les deux fonctionnalités. À titre d'exemple, exigeant une meilleure performance pour le LGS (ex. $lat1 = 75$ et $lat3 = 135$) mène à $FCS \rightarrow LGS$ comme meilleure stratégie. Plus généralement, pour ce cas d'étude, la fonctionnalité qui exige le moins de performance doit être intégrée en premier lieu.

Impact du nombre de chaînes fonctionnelles intégrés sur le coût Comme nous l'avons mentionné dans la section 4.5.2, le système du train d'atterrissage contient deux

Tableau 4.12 Coût des deux stratégies d'intégration

	$FCS \rightarrow LG$	$LG \rightarrow FCS$
$lat1 = 75$ $lat3 = 135$	(0 ; 2) =2	(6 ; 3) = 9
$lat1 = 135$ $lat3 = 75$	(6 ; 5) =11	(0 ; 5) = 5

ensembles d'atterrissage (celui d'avant et les deux principaux). Il est important de contrôler le temps requis pour la retraction/extension des deux ensembles. Ainsi, nous avons borné le temps de réponse de la chaîne fonctionnelle $FDE \rightarrow MLG \rightarrow FDE$. Nous désignons par $lat2$ le seuil requis pour le pire cas pour ce temps de réponse.

Nous avons refait la même expérience en intégrant le système du train d'atterrissage puis le système de contrôle de carburant en ajoutant l'exigence de latence pour la chaîne fonctionnelle $FDE \rightarrow MLG \rightarrow FDE$. Les résultats sont illustrés par le tableau 4.13 et le tableau 4.14. Nous notons aussi que pour cette expérience, nous exigeons que $lat1 = lat2$. En effet, fixant à titre d'exemple $lat1$ nettement inférieur à $lat2$ fait que le train avant réagit beaucoup plus rapidement que les trains principaux.

Dans la première étape d'intégration, nous remarquons que l'ajout de l'exigence de latence $lat2$ n'a pas d'impact sur les coûts d'intégration. Ceci est dû au fait qu'à ce niveau d'intégration, les modules IMA et les trames ne sont pas saturées. Toutefois dans la seconde étape, l'exigence $lat2$ à un impact important sur le coût global. Ainsi, il est judicieux d'exiger des performances moindres si l'objectif est d'intégrer plus de fonctionnalités.

Tableau 4.13 Coût d'intégration du LGS en bornant le WCD de la chaîne fonctionnelle $FDE \rightarrow MLG \rightarrow FDE$

$lat1$,$lat2$	74	[75..76]	77	[78..104]	[105..170]
Coût	∞	6	5	4	0

Tableau 4.14 Coûts d'intégrer FCS aprés LGS en bornant WCD de la chaîne fonctionnelle $FDE \rightarrow MLG \rightarrow FDE$

	75	76	[77..80]	[81..134]	[135..194]	[195..200]
75	∞	∞	∞	∞	7	6
76	∞	18	18	18	7	6
[77..80]	∞	18	11	11	7	6
[81..134]	∞	18	11	10	7	6
[135..200]	3	3	3	3	3	3

Sensibilité des coûts La dernière expérience qu'on mène dans le cadre de ce travail vise à déterminer lequel du réseau ou des modules IMA contribuent le plus dans le coût d'intégration. Pour ce cas d'étude, nous avons fixé le coût de reconfiguration de chaque partition à 1. Nous avons fait varier par la suite le coût de reconfiguration d'une trame TT. Les résultats sont illustrés par la figure 4.11 pour la première étape d'intégration et par la figure 4.12 pour la deuxième.

Comme le montre la figure 4.11 et la figure 4.12, le coût total est proportionnel au coût de reconfiguration des trames TT. Ainsi, pour ce cas d'étude le réseau a un impact plus gros sur les coûts. Ceci est dû au fait que le réseau est partagé par plusieurs fonctionnalités et chaque reconfiguration peut avoir un impact direct sur les autres fonctionnalités. Toutefois, la reconfiguration d'une partition peut être localisée au niveau du module distribué considéré.

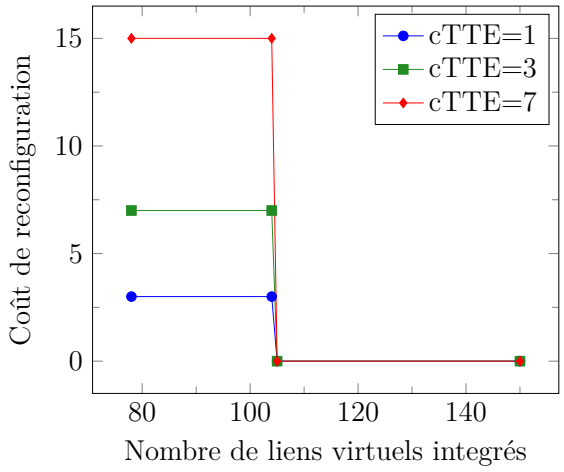


Figure 4.11 Coût de la 1ère étape d'intégration en fonction des coûts TTEthernet

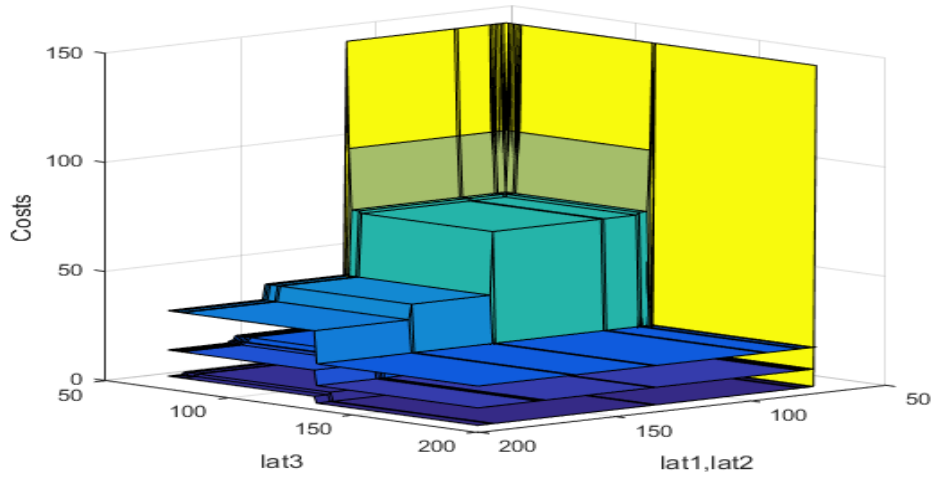


Figure 4.12 Coût de la 2ème étape d'intégration en fonction des coûts TTE

4.5.3 Évolutivité de l'approche

Après l'étude du cas du système du train d'atterrissage et du système de contrôle de carburant, nous visons dans cette partie à étudier l'évolutivité de notre approche avec différentes tailles de systèmes. Pour se faire, nous considérons une topologie en mesh avec 40 commutateurs. En supposant à titre d'exemple des commutateurs de 48-Ports, cette architecture physique large peut connecter 360 systèmes terminaux. Nous étudions l'évolutivité de notre approche en variant le nombre de systèmes terminaux connectés ainsi que le nombre de liens virtuels configurés.

Afin de réaliser nos expérimentations, nous avons développé un générateur de spécifications pour système avionique. Cet outil prend comme paramètres le nombre de systèmes terminaux et le nombre de liens virtuels et produit un système avionique en connectant uniformément les systèmes terminaux sur l'architecture et en distribuant d'une manière uniforme les liens virtuels sur le réseau. Chaque lien virtuel a au plus 4 chemins. Le nombre de chemins par lien virtuel est choisi d'une manière aléatoire de 1 (unidiffusion) à 4 (multidiffusion). Chaque lien virtuel est associé à une trame. Pour chaque trame, une nouvelle partition est ajoutée sur chaque module destination. Les partitions sources sont choisies avec une grande probabilité (0.9) entre les partitions du module destination. Ceci permet de créer une dépendance entre les trames. Si le module est vide, une nouvelle partition source est créée. Ainsi, le nombre de partitions attendues est de $(2.6|\mathcal{VL}|)$. Il est égal au nombre de partitions destinations attendues $(2.5|\mathcal{VL}|)$ plus le nombre de partitions nouvellement créées $(0.1|\mathcal{VL}|)$. Ainsi, nous pouvons estimer le seuil de saturation des systèmes terminaux (respectivement du réseau)

par le ratio $\frac{2.6|\mathcal{VC}|}{|\mathcal{ES}|}$ (respectivement $\frac{2.5|\mathcal{VC}|}{|\mathcal{ES}|}$).

Nous utilisons les mêmes paramètres temporaux pour les partitions et les trames que l'exemple de 4.5.2 (voir tableau 4.15). Nous définissons le coût de reconfiguration de chaque instance de partition à 7 et le coût de reconfiguration d'une instance de trame à 1. Les résultats reportés dans cette section sont une moyenne statistique de 4 à 10 exécutions des mêmes expériences. Nous fixons un délai limite de 6 heures pour chaque expérience et on utilise pour nos expérimentations une configuration typique d'ordinateur (Processeur i7 2.8GHz et 12GHz de RAM). Nous reportons dans cette section deux types d'expériences. Le premier type sert à illustrer l'utilité de notre approche pour maitriser les coûts. Le deuxième type d'expérience sert à montrer son impact sur la capacité d'intégration des systèmes larges.

Tableau 4.15 Paramètres temporaux du système

Paramètres	Valeurs
Périodicité des partitions	120,60,40
Longueur des partitions	[5,10]
Périodicité des trames	120,60,40
Longueur de trames	1,2
Seuils de latence	300
Coûts de reconfiguration des partitions	7
Coûts de reconfiguration des trames	1

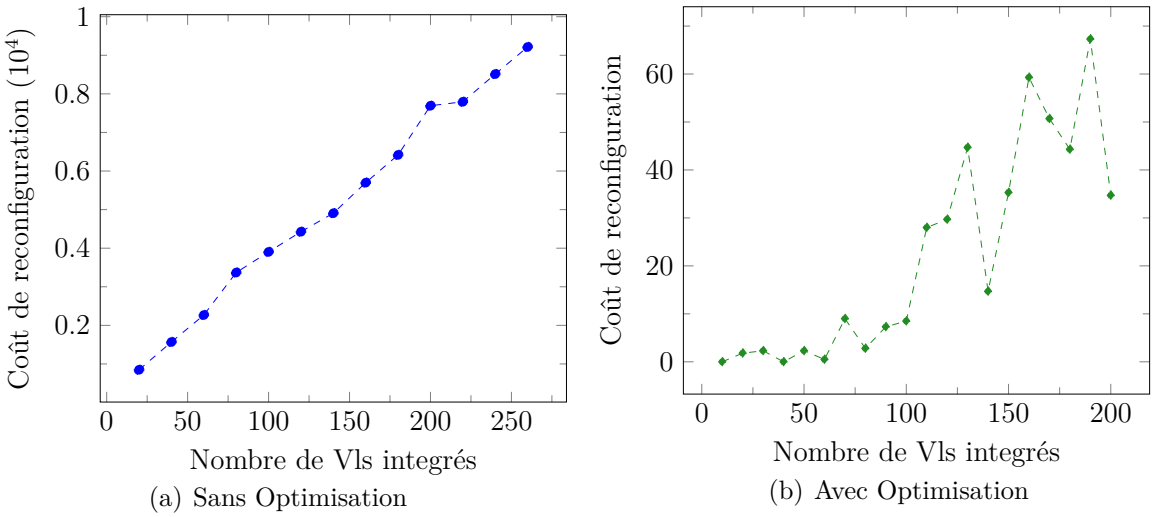
Utilité de l'approche d'optimisation de coût

Pour montrer l'importance de notre approche, nous la comparons avec deux cas extrêmes : (1) intégration sans optimisation de coût de reconfiguration (c.-à-d budget illimité) et (2) intégration sans reconfiguration (c-à-d. budget nul).

Les résultats de comparaison avec la stratégie (1) sont illustrés par les figures 4.13(a) et 4.13(b). Nous notons que le coût sans optimisation monte d'une manière spectaculaire à 8000 pour les systèmes larges dont nous n'optimisons pas les coûts. Ce coût reste raisonnablement bas avec notre approche d'optimisation vu que le coût le plus élevé est environ 60. Nous notons aussi que sans optimisation, nous sommes capable d'intégrer des systèmes plus larges. Il reste que l'approche d'optimisation donne des performances pas si mal dans une limite de temps de 6 heures.

Pour la comparaison avec la stratégie 2, nous détaillons les coûts reportés par la figure 4.13(b) en montrant les coûts obtenus lors de la deuxième et troisième étape d'intégration. Ces résultats sont illustrés par la figure 4.14. Nous notons que quand le système commence

à s'agrandir, les coûts sont non nuls même pour la deuxième étape d'intégration. Ainsi, il devient rapidement impossible d'intégrer le système sans reconfiguration. Ainsi, l'approche d'optimisation est une bonne alternative vu qu'elle aide à maitriser efficacement le coût tout en offrant la possibilité d'intégrer des systèmes larges dans un temps raisonnable.



(a) Sans Optimisation (b) Avec Optimisation

Figure 4.13 Coût de reconfiguration (un réseau avec 150 systèmes terminaux)

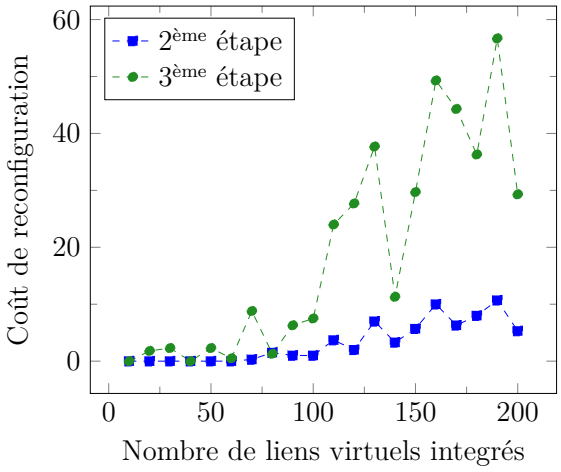


Figure 4.14 Coût optimisé (réseau avec 150 systèmes terminaux)

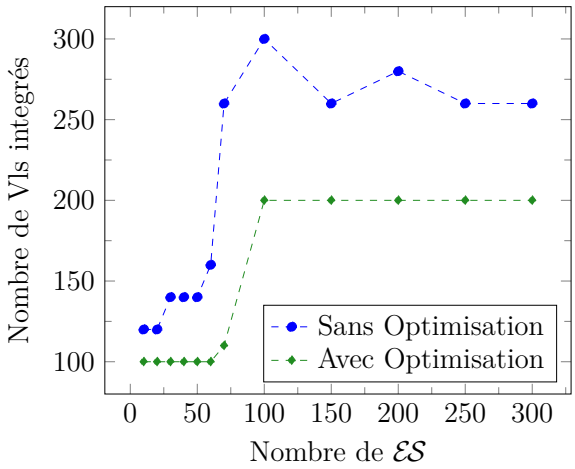


Figure 4.15 Perte en capacité d'intégration

Capacité d'intégration

Nous étudions dans cette section l'impact de l'optimisation de coût sur la capacité d'intégration. La capacité d'intégration est estimée par le nombre de liens virtuels intégrés pour

chaque nombre de systèmes terminaux connectés.

Nous estimons la perte en capacité par la figure 4.15 en divisant le nombre de liens virtuels intégrés en optimisant les coûts par le nombre de liens virtuels intégrés sans optimisation de coût. Nous notons que ce ratio n'excède pas un facteur de deux tiers.

Pour évaluer les coûts de réduction, nous référons plutôt à la figure 4.16. En utilisant le Test de Fisher (Fisher, 1935) avec un taux d'erreur de 5%, nous avons confirmé que les coûts évoluent linéairement en fonction du nombre de liens virtuels et ceci indépendamment du nombre des systèmes terminaux connectés. Nous pouvons ainsi étudier les coûts par les pentes de leurs modèles linéaires illustrés par la figure 4.16 et que nous notons par coût dérivé. Le gain en termes de réduction de coût est si élevé comme le montre la figure 4.16. Il est toujours supérieur à un facteur de 120 comme le montre la figure 4.17 où nous présentons le gain sous forme de ratio.

Ces deux expériences montrent que tandis que l'approche d'optimisation nous permet de maitriser d'une manière efficiente les coûts d'intégration, nous perdons un peu en termes de capacité d'intégration des systèmes larges. Ces résultats confirment que l'approche d'optimisation constitue une bonne alternative dans le contexte d'intégration des systèmes avioniques évolutifs.

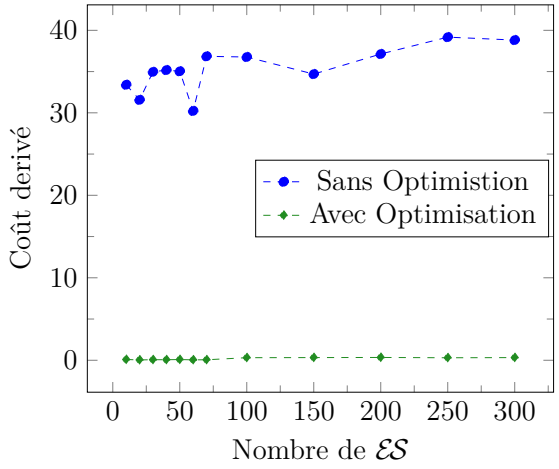


Figure 4.16 Gain en réduction de coût

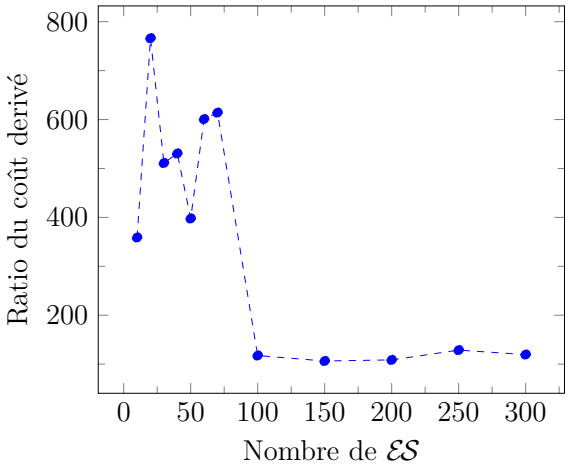


Figure 4.17 Gain en réduction de coût

4.6 Conclusion

Dans ce chapitre, nous avons présenté notre approche d'intégration itérative des fonctions avioniques déployées sur une architecture IMA et communicants à travers des flux TT du réseau TTEthernet. L'ultime objectif de cette approche et de trouver une nouvelle configu-

ration optimale en termes de coût de reconfiguration et qui vérifie les exigences temps réels qu'on a formalisé moyennant un système de contraintes.

Pour résoudre ce problème, nous avons adopté la méthodologie suivante. Nous avons présenté en premier lieu un modèle formel pour l'architecture avionique ainsi qu'un modèle de configuration associé. Ces deux modèles, nous en servi par la suite à définir un ensemble de contraintes qui modélisent les exigences temporelles d'un système avionique. Nous avons classé ses contraintes en contraintes dures et contraintes souples pour en définir par la suite notre approche WPMS pour l'intégration itérative des fonctionnalités avioniques. À travers deux cas d'études, nous avons illustré et analysé notre approche et nous avons dégagé quelques directives pour aider l'ingénieur système aux meilleurs choix d'intégration. Nous avons par la suite étudié l'évolutivité de notre approche en considérant des systèmes plus larges.

Nous nous intéresserons dans le chapitre suivant à considérer lors du processus d'intégration non seulement les flux TT mais aussi les flux RC de TTEthernet. Ceci a pour objectif de rendre notre approche générique en considérant les différents types de communication pour des fonctions avioniques critiques.

CHAPITRE 5 INTÉGRATION ITÉRATIVE DES FLUX DÉCLENCHÉS PAR LE TEMPS ET À QUOTAS LIMITÉS

Nous présentons dans ce chapitre notre approche d'intégration itérative pour les applications multi-critiques déployées sur un réseau TTEthernet. Cette approche a été accepté comme publication dans (Beji et al., 2018b). À la différence du chapitre précédent, les applications dans le cadre de ce chapitre peuvent communiquer à travers deux types de flux TT et RC. Plus précisément, ayant des fonctionnalités avioniques déjà intégrées et communiquant moyennant ces deux types de flux, l'objectif dans ce cadre et de considérer l'ajout d'autres flux résultant de l'intégration de nouvelles fonctionnalités. Nous cherchons ainsi à synthétiser un nouvel ordonnancement pour la totalité des flux TT et que l'ensemble des flux TT et RC partageant le même réseau et vérifient l'ensemble des contraintes temporelles que nous spécifierons tout au long de ce chapitre.

Pour résoudre cette problématique, nous opterons pour la méthodologie suivante. Nous présenterons dans la section 5.1 le modèle d'architecture et le modèle d'application d'un réseau TTEthernet et nous formulerons par la suite le problème d'intégration. Les sections 5.2 et 5.3 présentent la partie majeure des contributions techniques. Dans la section 5.2, nous présentons les exigences moyennant un ensemble de contraintes. La section 5.3 détaille notre approche d'intégration à deux phases et notre heuristique. Finalement, nous illustrons dans la section 5.4 l'applicabilité de notre approche à travers un cas d'étude.

5.1 Modèle Formel

Après un bref rappel du modèle de l'architecture TTEthernet présenté dans 4.1.1, nous étondons notre modèle d'application TTEthernet présenté dans le paragraphe 4.1.2 pour prendre en considération les flux RC.

5.1.1 Modèle des concepts généraux de TTEthernet

Une architecture typique d'un réseau TTEthernet est composée d'un ensemble de modules distribués appelés *Systèmes Terminaux* notés $\mathcal{ES}$. Ces systèmes sont interconnectés par un ensemble de liens physiques et un ensemble de *commutateurs réseaux* notés $\mathcal{NS}$. En suivant les notations de (Steiner, 2010), un réseau TTEthernet peut être représenté par un graphe $\mathcal{G}(\mathcal{V}, \mathcal{E})$ où l'ensemble des nœuds est $\mathcal{V} = \mathcal{ES} \cup \mathcal{NS}$. Quant à l'ensemble des arêtes $\mathcal{E}$, il représente plutôt l'ensemble des liens physiques de l'architecture. Chaque lien physique connecte

deux nœuds notés u et v. Ce lien permet une transmission dans les deux sens soit de u vers v et de v vers u formant ainsi deux liens de donnés. Nous notons le lien physique par (u, v), le lien de données de u vers v par $[u, v]$ et par $\mathcal{L}$ l'ensemble des lien de données du réseau.

Une séquence adjacente de liens de données est notée par $p = [[v_1, v_2], ..., [v_{r-1}, v_r]]$. Cette séquence connecte le système terminal v_1 à un autre distant v_r ($v_1, v_r \in \mathcal{ES}$) à travers un chemin non cyclique appelé chemin de données. Nous notons par $\mathcal{DP}$ l'ensemble de tous les chemins du réseau

La communication TTEthernet est basée sur le concept de lien virtuel. Chaque lien virtuel vl transporte une trame d'une source unique à une ou plusieurs destinations. Ainsi, $vl = \bigcup_{k=1..n} p_k$, où p_k est le chemin de données du lien virtuel et n est le nombre de systèmes terminaux récepteurs.

Un lien virtuel a la structure d'un arbre où l'élément racine est le système terminal. Il envoie aux systèmes terminaux récepteurs qui constituent les feuilles de l'arbre. Un chemin de la racine à une feuille forme un chemin de données. Plus encore, la racine a un seul fils (qui doit être un commutateur réseau) et les feuilles sont des systèmes terminaux deux à deux disjoints.

5.1.2 Modèle d'application

Suivant la convention de TTEthernet, l'information communiquée de l'expéditeur au receveur est sous forme d'un message appelé *trame*. TTEthernet intègre à la fois les trames TT et RC sur le même réseau. Chaque trame a sa *date limite* qui n'est qu'un seuil représentant le pire cas de délai de transmission toléré pour cette trame sur tous les chemins de son lien virtuel. Nous avons caractérisé dans la section 4.1.2 une trame TT moyennant trois paramètres. Dans ce chapitre, afin de borner les délais de transmission des trames TT sur leurs liens virtuels, nous ajoutons un quatrième paramètre *deadline*. Ainsi, la caractérisation temporelle d'une trame TT se redéfinit comme suit.

Définition 5 (Trame TT) *Une trame TT f_i est périodique et elle est paramétrée par le tuple $f_i = (f_i.T, f_i.C, f_i.deadline)$ où $f_i.T$ est sa période, $f_i.C$ la durée de transmission de f_i sur un seul lien et $f_i.deadline$ est la date limite pour f_i.*

Les trames RC ne sont pas nécessairement périodiques, mais un intervalle de temps minimum est exigé entre deux instances de chaque trame. Cet intervalle est connu sous le nom de BAG. Nous dénotons par $f_i.Rate$ le quota réel d'une trame notée f_i de type RC. Le BAG d'une trame doit être plus petit ou égal à $1/f_i.Rate$ pour assurer l'ordonnancabilité de chaque instance

de trame. À titre d'exemple, si chaque $10ms$ une instance valide de trame est produite ($f_i.Rate = 1/10$) alors $f_i.BAG \leq 10ms$. Par conséquent, nous caractérisons une trame RC comme suit.

Définition 6 (Trame RC) *Une trame RC notée f_i est paramétrée par le 4-tuplet $f_i = (f_i.T,\ f_i.BAG,\ f_i.C,\ f_i.deadline)$ où $f_i.T = 1/f_i.Rate$.*

Un lien virtuel vl_i transporte multiples instances de la trame f_i de sa source à ses destinations. L'intégration des trames TT et RC est complètement définie par l'allocation des fenêtres de temps pour la transmission de chaque instance de trame TT notée f_i sur chaque lien de son lien virtuel associé noté vl_i. Chaque instance de trame doit atteindre sa destination dans les délais impartis. Nous ordonnançons les instances de trame au lieu des trames. Ainsi, dans la suite de ce travail, nous nous focalisons plutôt sur les instances de trames. Nous dénotons par $f^l_{i,k}$ la $k^{\text{ème}}$ instance de trame f_i sur son lien l du lien virtuel vl_i. Ainsi, nous avons $f^l_{i,k}.T = f_i.T$ idem pour BAG, C et $deadline$.

En se basant sur les périodes T de toutes les trames TT et RC, nous définissons l'hyper-période (HP) de la communication réseau comme suit.

$$HP = PPCM(\{f_i.T : vl_i \in \mathcal{VL}\}) \tag{5.1}$$

où $PPCM$ dénote la fonction du plus petit multiple commun. Nous pouvons ainsi limiter notre étude à HP et déterminer le nombre des instances pour une trame f_i comme suit :

$$Instances(f_i) = \frac{HP}{f_i.T} \tag{5.2}$$

Nous notons que chaque lien virtuel vl_i induit des structures d'arbres. Chaque arbre est noté par $f_{i,k}$ et caractérise la transmission du $k^{\text{ème}}$ instance de f_i. Les nœuds de $f_{i,k}$ sont les instances de trame $f^l_{i,k}$. Ainsi, d'une manière similaire aux notations introduites dans la section 4.1.1 pour les liens virtuels, nous utilisons les notations fonctionnelles suivantes pour désigner :

- $first(f_{i,k})$: la $k^{\text{ème}}$ instance de f_i sur le premier lien de $f_{i,k}$;
- $leaves(f_{i,k})$: l'ensemble des feuilles de $f_{i,k}$;
- children($f^l_{i,k}$) : l'ensemble des fils de $f^l_{i,k}$;
- pred($f^l_{i,k}$) : le prédécesseur (père) de $f^l_{i,k}$.

La $k^{\text{ème}}$ transmission de f_i dépend de l'exécution d'une tâche sur le système terminal. En effet, cette tâche rend l'information à communiquer disponible. Ainsi, outre que les paramètres temporaux caractérisant une trame f_i, nous caractérisons chaque $f_{i,k}$ par les deux paramètres suivants

— $f_{i,k}.min\_avail$: la première date pour laquelle la $k^{\text{ème}}$ instance de cette tâche termine son exécution ;

— $f_{i,k}.max\_avail$: la dernière date pour laquelle la $k^{\text{ème}}$ instance de cette tâche termine son exécution.

Nous utilisons les notations suivantes pour indiquer :

— $\mathcal{F}_{TT}$: l'ensemble de toutes les instances des trames TT.

— $\mathcal{F}_{RC}$: l'ensemble de toutes les instances des trames RC.

— Γ : l'ensemble de toutes les structures d'arbres induites par tous les liens virtuels définies dans le réseau.

— Γ_{TT} : l'ensemble de toutes les structures d'arbres induites par les liens virtuels associés aux trames TT.

— Γ_{RC} : l'ensemble de toutes les structures d'arbres induites par les liens virtuels associés aux trames RC.

La transmission d'une trame RC sur le réseau est contrôlé par deux types d'évènements : (1) *availability* qui indique la date de mise en disponibilité de l'information à transmettre par l'application distribuée. (2)*regulation* qui indique la date où le régulateur de trafic débloque la transmission des instances des trames $first(f_{i,k})$ pour $k \in Instances(f_i)$ au niveau de chaque système terminal. Nous modélisons ces deux types d'évènements par les fonctions suivantes.

Définition 7 (Disponibilité de l'information RC) *Une disponibilité de l'information est une fonction partielle qui associe pour chaque structure d'arbre générée par une trame RC un temps. Cette fonction est notée comme suit :* $\text{avail} : \Gamma_{RC} \rightarrow \mathbb{N}$

Définition 8 (Régulation de trafic RC) *Une régulation de trafic est une fonction partielle qui associe pour chaque structure d'arbre générée par une trame RC un temps. La fonction régulation est notée comme suit :* $\text{regul} : \Gamma_{RC} \rightarrow \mathbb{N}$

La figure 5.1 illustre un exemple de transmission des évènements RC. L'exemple contient un système terminal ES hébergeant deux applications logicielles (ex. SA_1,SA_2). Ces deux applications envoient trois trames (ex. $f_i, i = 1..3$). Les différentes instances de chaque trame sont stockées dans une mémoire tampon dédiée indiquant la disponibilité de l'information

produite. Au niveau du contrôleur TTEthernet, trois régulateurs de trafic ($TR_i, i = 1..3$) est associé avec différentes dates de régulation et de disponibilités de trames f_i. Nous modélisons dans cette figure les différentes dates de disponibilité et de régulation de la trame f_2 notés par avail($f_{2,k}$) et regul($f_{2,k}$). Une fois que les trames RC sont produites par différentes applications logicielles hébergées dans ES, un contrôleur TTEthernet noté SC contrôle l'envoi de ces trames sur le réseau

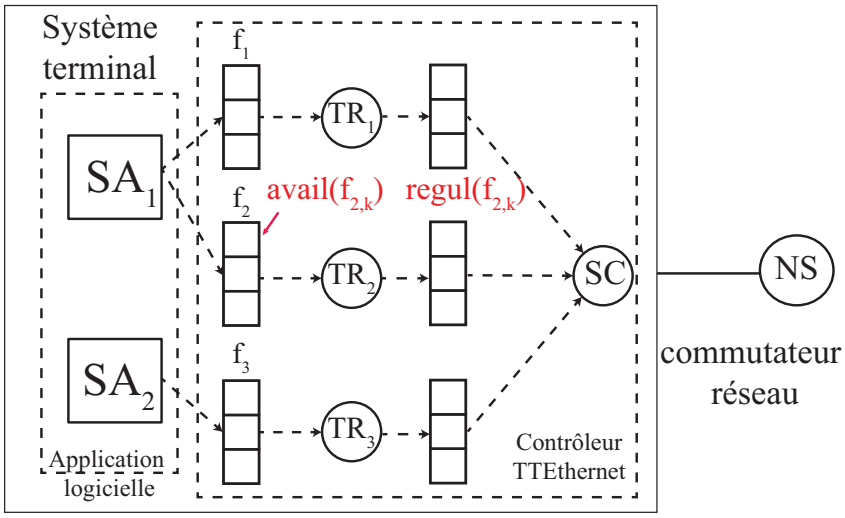


Figure 5.1 Exemple d'évènements de trafic

Comme nous l'avons mentionné ci-dessus, un *ordonnancement TT* est statique et complètement défini en spécifiant les dates d'activation d'envoi pour les différentes instances de trame TT. Synthétisant un tel ordonnancement se réduit à résoudre un ensemble de contraintes qui définissent les propriétés temporelles du tout le réseau. Nous pouvons définir un ordonnancement TT comme suit.

Définition 9 (Ordonnancement TT) *Un ordonnancement TT est une fonction partielle qui associe pour chaque instance de trame TT un temps :* $\sigma : \mathcal{F}_{TT} \rightarrow \mathbb{N}$.

La fonction partielle σ attribue un *offset*, qui est un temps de début pour une instance de trame. Nous dénotons par dom(σ) le domaine de σ qui est l'ensemble d'instances de trames TT pour lesquels σ est défini. Étant donné l'ensemble des instances de trame $F \subseteq \mathcal{F}_{TT}$, nous dénotons par $\mathbb{N}^F = \{\sigma \,|\, \text{dom}(\sigma) = F\}$ l'ensemble des ordonnancements pour les instances de trame F.

L'*ordonnancement RC*, a une priorité plus faible que celle du TT. Il est défini avec les principes suivants. La transmission d'une trame RC seulement si l'information communiquée est rendue

disponible. Au niveau de chaque lien de données, la transmission de cette trame est activée seulement si elle peut terminer sa transmission avant les instances de trames TT planifiées d'avance. Cette stratégie d'intégration des trames TT et RC connue sous le nom de stratégie de blocage. C'est la stratégie adaptée pour les systèmes temps-critiques. Les trames RC sont ordonnancées conformément à la politique FIFO. Étant un ordonnancement TT noté σ et la disponibilité de l'information avail, Nous caractérisons un ordonnancement RC comme suit.

Définition 10 (Ordonnancement RC) *Un ordonnancement RC est une fonction partielle qui associe pour chaque instance de trame RC un temps :* $\alpha : \mathcal{F}_{RC} \rightarrow \mathbb{N}$

La fonction partielle α attribue un instant d'envoi aux instances de trames RC. Un commutateur TTEthernet traite une trame TT (respectivement une trame RC) pour un délai *tt_switch_delay* (respectivement *rc_switch_delay*). Dans la section 5.2, nous présentons l'ensemble des contraintes qui doivent être satisfaites par un ordonnancement valide d'un réseau TTEthernet.

5.2 Contraintes d'intégration des flux TT et RC

Nous modélisons dans cette section moyennant un ensemble de contraintes les exigences temporelles pour la transmission des flux TT et RC. Ces contraintes peuvent être classées en quatre catégories. La première catégorie est liée aux trames, la seconde est liée plutôt aux liens de données et le troisième aux liens virtuels. Nous réservons également une quatrième catégorie pour les contraintes de latence.

5.2.1 Contraintes de trames

Contraintes de périodicité

Chaque instance de trame TT doit être ordonnancée une fois durant sa période. La logique de cette exigence est formalisée par l'ensemble suivant de contraintes :

$$\forall f^l_{i,k} \in \mathcal{F}_{TT},\ (k-1) \times f^l_{i,k}.T \leq \sigma(f^l_{i,k}) \leq k \times f^l_{i,k}.T \qquad (Périodicité)$$

Contraintes de Régulation de trafic

La transmission de la $k^{\text{ème}}$ instance de trame f_i est débloqué par le régulateur de trafic seulement si l'information transmise est rendue disponible par le système terminal et qu'une distance minimale de $f_i.BAG$ est assuré à partir de la disponibilité de l'instance $k-1$ de

la même trame. Pour l'instance de trame *first*($f_{i,1}$)), la date de déblocage est aussitôt que l'information est rendue disponible par le système terminal. Ce type de contraintes régule le trafic de chaque instance de trame RC notée $f^l_{i,k}$ à une chaque $f_i.BAG$ et peut être formalisé comme suit.

$$\begin{aligned}
&\forall f_{i,k} \in \Gamma_{RC}, \\
&\Big(k = 1 \Rightarrow regul(f_{i,k}) = avail(f_{i,k})\Big) \wedge \\
&\Big(k > 1 \Rightarrow regul(f_{i,k}) = max(avail(f_{i,k}), regul(f_{i,k-1}) + f_i.BAG)\Big) \qquad (\textit{Régulation})
\end{aligned}$$

5.2.2 Contraintes de lien

Contraintes de non-chevauchement

Nous considérons dans cette section l'exclusion mutuelle des instances de trames transmises sur le même lien. Étant donné un lien de données l, nous considérons deux instances de trames différentes sur le lien l notées par $f^l_{i,k}$ et $f^l_{j,k'}$. Étant donné aussi l'ordonnancement TT σ et la fonction de disponibilité avail, l'exclusion mutuelle peut être assuré en exigeant que la fin de transmission de $f^l_{i,k}$ se produit avant le début de $f^l_{j,k'}$ et vice versa. Nous avons déjà défini dans la section 4.3.2 la contrainte de non-chevauchement dans le cas d'un réseau véhiculant seulement des flux TT. Pour des réseaux qui peuvent inclure les deux flux, ces contraintes se redéfinissent comme suit.

$$\begin{aligned}
&\forall f^l_{i,k}, f^l_{j,k'} \in \mathcal{F}_{TT} \cup \mathcal{F}_{RC}, \\
&\Big(\Big((f^l_{i,k}, f^l_{j,k'} \in \mathcal{F}_{TT}) \wedge (i \neq j)\Big) \Rightarrow \\
&\quad \Big(\Big(\sigma(f^l_{j,k'}) + f^l_{j,k'}.C\Big) \leq \sigma(f^l_{i,k})\Big) \vee \Big(\Big(\sigma(f^l_{i,k}) + f^l_{i,k}.C\Big) \leq \sigma(f^l_{j,k'})\Big)\Big) \\
&\wedge \Big((f^l_{j,k} \in \mathcal{F}_{TT}, f^l_{j,k'} \in \mathcal{F}_{RC}) \Rightarrow \\
&\quad \Big(\alpha(f^l_{j,k'}) + f^l_{j,k'}.C) \leq \sigma(f^l_{i,k})\Big) \vee \Big((\sigma(f^l_{i,k}) + f^l_{i,k}.C) \leq \alpha(f^l_{j,k'}))\Big)\Big) \\
&\wedge \Big(\Big((f^l_{i,k}, f^l_{j,k'} \in \mathcal{F}_{RC}) \wedge (i \neq j)\Big) \Rightarrow \\
&\quad \Big(\alpha(f^l_{j,k'}) + f^l_{j,k'}.C) \leq \alpha(f^l_{i,k})\Big) \vee \quad \Big((\alpha(f^l_{i,k}) + f^l_{i,k}.C) \leq \alpha(f^l_{j,k'}))\Big)\Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (Non_Chevauchement)
\end{aligned}$$

Les deux premières lignes de la contrainte coïncident avec le cas de non-chevauchement de deux trames TT. C'est le cas d'exclusion mutuelle de transmission étudié dans la section 4.3.2.

Nous étendons cette contrainte en exigeant le non-chevauchement de deux trames RC dans la deuxième et troisième ligne de la contrainte. Nous exigeons également le non-chevauchement TT et RC dans la cinquième et sixième ligne.

Contraintes FIFO

Deux instances de trames différentes $f^l_{i,k}$ et $f^l_{j,k'}$ partageant le même lien de données l, doivent être transmises conformément à la politique FIFO. Deux cas se présentent dans ce cadre. Dans le premier cas, l est le premier lien de données sur lequel f_i et f_j sont transmises. Nous considérons dans ce cas les dates $regul(f_{i,k})$ et $regul(f_{j,k'})$ pour déterminer quelle communication doit se libérer en premier lieu. Ainsi, $f^l_{i,k'}$ et $f^l_{j,k''}$ sont ordonnancés dans cet ordre. Dans le deuxième cas, $f^l_{i,k'}$ et $f^l_{j,k''}$ sont relayés par un commutateur. Leurs ordres de transmission sont donnés par leurs ordres d'arrivées au niveau de commutateur. Nous pouvons exprimer cet ensemble de contraintes comme suit.

$$
\begin{aligned}
&\forall f^l_{i,k}, f^l_{j,k'} \in \mathcal{F}_{RC}, \\
&\Bigg((f^l_{i,k} = \mathit{first}(f_{i,k})) \wedge (i \neq j) \Rightarrow \\
&\Big(\Big(regul(f_{i,k}) < regul(f_{j,k'}) \Big) \Leftrightarrow \Big(\alpha(f^l_{i,k}) < \alpha(f^l_{j,k'}) \Big) \Big) \Bigg) \\
&\wedge \Bigg((f^l_{i,k} \neq \mathit{first}(f_{i,k})) \wedge (i \neq j) \Rightarrow \\
&\Big(\Big(\alpha(\mathrm{pred}(f^l_{i,k})) < \alpha(\mathrm{pred}(f^l_{j,k'})) \Big) \Leftrightarrow \Big(\alpha(f^l_{i,k}) < \alpha(f^l_{j,k'}) \Big) \Big) \Bigg)
\end{aligned}
\qquad (FIFO)
$$

Contrainte d'envoi le plus tôt possible

Une instance de trame RC $f^l_{i,k}$ est envoyée sur un lien l dès que ce dernier est libre et qu'il existe assez de temps suffisant pour finir la transmission de cette instance avant la transmission d'une autre instance de trame TT. Pour exprimer cet ensemble de contraintes, nous soulevons deux cas.

Dans le premier cas, le nœud qui envoie la trame est un système terminal. Le lien de donnée en sortie l communique seulement l'instance de la trame f_i ou une autre trame f_j partage avec f_i le lien l. Dans la première alternative, $f^l_{i,k}$ est envoyée dés qu'elle est libérée par le régulateur de trafic.

Dans la seconde alternative, nous assurons que $f^l_{i,k}$ ne peut pas être ordonnancé plus tôt

(i.e un ordonnancement plut tôt viole la contrainte de non-chevauchement sur le lien l). Le premier cas peut être formalisé comme suit.

$$\begin{aligned}
&\forall f^l_{i,k} \in \mathcal{F}_{RC}, \exists f^l_{j,k'} \in \mathcal{F}_{TT} \cup \mathcal{F}_{RC}, \quad (f^l_{i,k} = \mathit{first}(f_{i,k})) \Rightarrow \\
&\left(\alpha(f^l_{i,k}) = \text{regul}(f_{i,k})\right) \\
&\vee \left(\left(\alpha(f^l_{i,k}) > \mathit{regul}_i(k)\right) \wedge (i \neq j)\right. \\
&\quad \wedge \left(f^l_{j,k'} \in \mathcal{F}_{TT} \Rightarrow \left(\sigma(f^l_{j,k'}) \leq \alpha(f^l_{j,k}) - 1 \leq \sigma(f^l_{j,k'}) + f^l_{j,k'}.C\right)\right) \\
&\quad \left.\wedge \left(f^l_{j,k'} \in \mathcal{F}_{RC} \Rightarrow \left(\alpha(f^l_{j,k'}) \leq \alpha(f^l_{j,k}) - 1 \leq \alpha(f^l_{j,k'}) + f^l_{j,k'}.C\right)\right)\right)
\end{aligned} \tag{$Envoi\_Plus\_T\hat{o}t_1$}$$

Dans le deuxième cas, nous utilisons le même raisonnement pour les liens intérieurs. Dans la première alternative où l est dédié à la transmission de $f^l_{i,k}$. Ce dernier est relayé directement après l'écoulement d'un délai de rc_switch_delay commençant de son arrivée au niveau du commutateur. Le deuxième cas peut être formalisé comme suit.

$$\begin{aligned}
&\forall f^l_{i,k} \in \mathcal{F}_{RC}, \exists f^l_{j,k'} \in \mathcal{F}_{TT} \cup \mathcal{F}_{RC}, \left(f^l_{i,k} \neq \mathit{first}(f_{i,k})\right) \Rightarrow \\
&\left(\alpha(f^l_{i,k}) = \alpha(\text{pred}(f^l_{i,k})) + \text{pred}(f^l_{i,k}).C + rc_sw_delay\right) \\
&\vee \left(\left(\alpha(f^l_{i,k}) > \alpha(\text{pred}(f^l_{i,k})) + \text{pred}(f^l_{i,k}).C + rc_sw_delay\right)\wedge\right. \\
&\quad (i \neq j) \wedge \left(f^l_{j,k'} \in \mathcal{F}_{TT} \Rightarrow \left(\sigma(f^l_{j,k'}) \leq \alpha(f^l_{j,k}) - 1 \leq \sigma(f^l_{j,k'}) + f^l_{j,k'}.C\right)\right) \\
&\quad \left.\wedge \left(f^l_{j,k'} \in \mathcal{F}_{RC} \Rightarrow \left(\alpha(f^l_{j,k'}) \leq \alpha(f^l_{j,k}) - 1 \leq \alpha(f^l_{j,k'}) + f^l_{j,k'}.C\right)\right)\right)
\end{aligned} \tag{$Envoi\_Plus\_T\hat{o}t_2$}$$

5.2.3 Contrainte du lien virtuel

Contraintes de dépendance de chemin

Nous introduisons cet ensemble de contraintes pour décrire la transmission séquentielle d'une instance de trame TT tout au long d'un chemin du lien virtuel vl_i. Un chemin est bien formé si nous avons l'ordre d'évènements suivant sur deux liens de données adjacents. (1) l'arrivé de l'instance de trame $f^l_{i,k}$ au niveau de commutateur. (2) un délai de traitement de tt_sw_delay d'une instance de trame TT par le commutateur. (3) l'envoi de l'instance de trame dans le prochain lien de données. La dépendance de chemin TT a déjà été définie dans la section 4.3.2. Elle se redéfinit dans notre modèle étendu comme suit.

$$\begin{aligned}
&\forall f^l_{i,k} \in \mathcal{F}_{TT}, \\
&\Big(\Big(f^l_{i,k} = \mathit{first}(f_{i,k})\Big) \Rightarrow \Big(f_{i,k}.max_avail \leq \sigma(f^l_{i,k})\Big)\Big) \\
&\wedge \Big(f^l_{i,k} \neq \mathit{first}(f_{i,k})\Big) \Rightarrow \Big(\sigma(\mathrm{pred}(f^l_{i,k})) + \mathrm{pred}(f^l_{i,k}).C + tt_sw_delay\Big) \leq \sigma(f^l_{i,k})\Big)
\end{aligned} \tag{$D\acute{e}pendance\_Chemin_1$}$$

D'une manière similaire, la transmission séquentielle des flux RC sur leurs chemins de données peuvent être exprimée comme suit.

$$\begin{aligned}
&\forall f^l_{i,k} \in \mathcal{F}_{RC}, \\
&\Big(\Big(f^l_{i,k} = \mathit{first}(f_{i,k})\Big) \Rightarrow \Big(regul(f_{i,k}) \leq \alpha(f^l_{i,k})\Big)\Big) \\
&\wedge \Big(f^l_{i,k} \neq \mathit{first}(f_{i,k})\Big) \Rightarrow \Big(\alpha(\mathrm{pred}(f^l_{i,k})) + \mathrm{pred}(f^l_{i,k}).C + rc_sw_delay\Big) \leq \alpha(f^l_{i,k})\Big)
\end{aligned} \tag{$D\acute{e}pendance\_Chemin_2$}$$

Contraintes de limite de mémoire

Les commutateurs réseau ont des capacités de mémoire limitées. Ainsi, le temps écoulé entre la transmission de deux instances de trames $f^l_{i,k}$ et $f^{l'}_{i,k}$ avec $f^{l'}_{i,k} \in \mathrm{children}(f^l_{i,k})$ ne doit pas dépasser un seuil de temps noté mem_bound_i. La limite de mémoire a déjà été définie dans la section 4.3.2. Elle se redéfinit dans notre modèle étendue comme suit.

$$\forall f^l_{i,k} \in \mathcal{F}_{TT}, \forall f^{l'}_{i,k} \in \mathrm{children}(f^l_{i,k}), \Big(\sigma(f^{l'}_{i,k}) - \sigma(f^l_{i,k})\Big) \leq mem\_bound_i \quad (Limite\_M\acute{e}moire)$$

Contraintes de relai simultanée

Dans certaines applications, il est exigé que les instances de trame TT sont relayées simultanément sur plusieurs liens. Nous devons s'assurer dans ce cadre que les temps offsets sont égaux sur ces liens. Le relai simultané a déjà été défini dans la section 4.3.2. Il se redéfinit dans notre modèle étendu comme suit.

$$\forall f_{i,k}^{l} \in \mathcal{F}_{TT}, \forall \in f_{i,k}^{l'}, f_{i,k}^{l''} \in \text{children}(f_{i,k}^{l}), SimCasted(f_i) \Rightarrow \left(\sigma(f_{i,k}^{l'}) = \sigma(f_{i,k}^{l''})\right) \quad (Relai\_Simultan\acute{e})$$

où $SimCasted(f_i)$ est un prédicat qui retourne *vrai* si les instances de trame $f_{i,k}^{l}$ doivent être diffusés simultanément sur les liens sortants au niveau d'un commutateur.

5.2.4 Contraintes de latence

Afin de s'assurer que la communication entre des applications distribuées s'achève dans les délais impartis, nous définissons les contraintes de latence. Moyennant ces contraintes, nous déterminons pour chaque $k^{\text{ème}}$ instance de trame f_i le temps écoulé entre l'envoi de $first(f_{i,k})$ et la fin de transmission de chaque instance de trame $f_{i,k}^{l} \in leaves(f_{i,k})$. Nous nous assurons aussi que cette durée est inférieure à un seuil $f_{i,k}.deadline$. Pour la communication des trames TT, les contraintes de latence peuvent être définies comme suit.

$$\forall f_{i,k} \in \Gamma_{TT}, \forall f_{i,k}^{l} \in leaves(f_{i,k}), \left(\sigma(f_{i,k}^{l}) + f_{i,k}^{l}.C - \sigma(first(f_{i,k}))\right) \leq f_{i,k}.deadline \quad (Latence)$$

Pour la communication RC, nous ne définissons pas un ensemble de contraintes pour assurer les exigences de latence. Nous montrons dans la section 5.3 notre méthode pour prouver que les flux RC ne violent pas les exigences de latence. En se basant sur cet ensemble de contraintes, nous caractérisons formellement dans la section 5.3 le problème d'intégration itérative et nous proposons une approche pour le résoudre.

5.3 Approche d'intégration itérative des flux TT et RC

Nous considérons dans cette section la définition d'une approche CP qui résout le problème d'intégration itérative. Nous définissons dans une première partie notre approche à base de contraintes pour résoudre le problème d'intégration itérative. Dans la dernière partie, nous définissons notre heuristique pour l'amélioration des performances de résolution.

5.3.1 Caractérisation de l'approche d'intégration itérative

Un problème d'intégration itérative considère un ensemble d'instances de trames TT déjà configurées $\mathcal{F}_{TT/old}$ et leur ordonnancement $\sigma_{old} \in \mathbb{N}^{\mathcal{F}_{TT/old}}$ et nous cherchons un nouvel ordonnancement $\sigma \in \mathbb{N}^{\mathcal{F}_{TT}}$ qui minimise le coût de reconfiguration définie comme suit.

$$g(\sigma) = \sum_{f^l_{i,k} \in \mathcal{F}_{TT/old}} \partial^l_{i,k}(\sigma, \sigma_{old}) cost(f^l_{i,k}) \tag{5.3}$$

où

— $\mathcal{F}_{TT} = \mathcal{F}_{TT/old} \cup \mathcal{F}_{TT/new}$ et $\mathcal{F}_{TT/new}$ représente le nouveau trafic TT que nous voulons intégrer.

— $cost(f^l_{i,k})$, est le coût de modifier le temps d'offset d'une trame.

— $\partial^l_{i,k}(\sigma, \sigma_{old})$ est la fonction indicateur retournant 1 si $\sigma(f^l_{i,k}) \neq \sigma_{old}(f^l_{i,k})$ et 0 autrement.

Bien évidemment, nous supposons que σ_{old} satisfait les contraintes définies dans la section 5.2 et σ doit aussi satisfaire ces contraintes. Malgré que le coût de reconfiguration porte seulement sur l'ordonnancement TT noté σ, nous devons néanmoins considérer l'ordonnancement des flux RC. En effet, les deux trafics partagent les mêmes liens de communication et tandis que l'ordonnancement TT est statique (avec une priorité haute), l'ordonnançabilité des trames RC dépend des fenêtres de temps réservés pour les trames TT.

Pour résoudre le problème d'intégration itérative, nous optons pour une approche à deux phases illustrée par la figure 5.2. Le but de la première phase est de synthétiser un ordonnancement σ optimal et nous nous concentrons dans la deuxième phase sur la détermination de *Pire Cas de Latence* ou Worst-Case Latency (WCL) de chaque trame RC.

Nous notons que dans la deuxième phase nous calculons itérativement pour chaque $f_{i,k} \in \Gamma_{RC}$ le *pire cas de latence* $WCL(f_{i,k})$ et nous testons si chacun est inférieur à $f_{i,k}.deadline$. Dans le cas où $WCL(f_{i,k}) > f_{i,k}.deadline$, nous concluons que pour l'ordonnancement fixé σ, le pire cas de latence viole l'exigence de latence requise pour $f_{i,k}$. Nous devons, ainsi, ignorer l'ordonnancement produit σ et nous cherchons un autre ordonnancement optimal. En considérant la formalisation des différentes contraintes de la section 5.2, nous caractérisons le problème d'optimisation de la première phase comme suit.

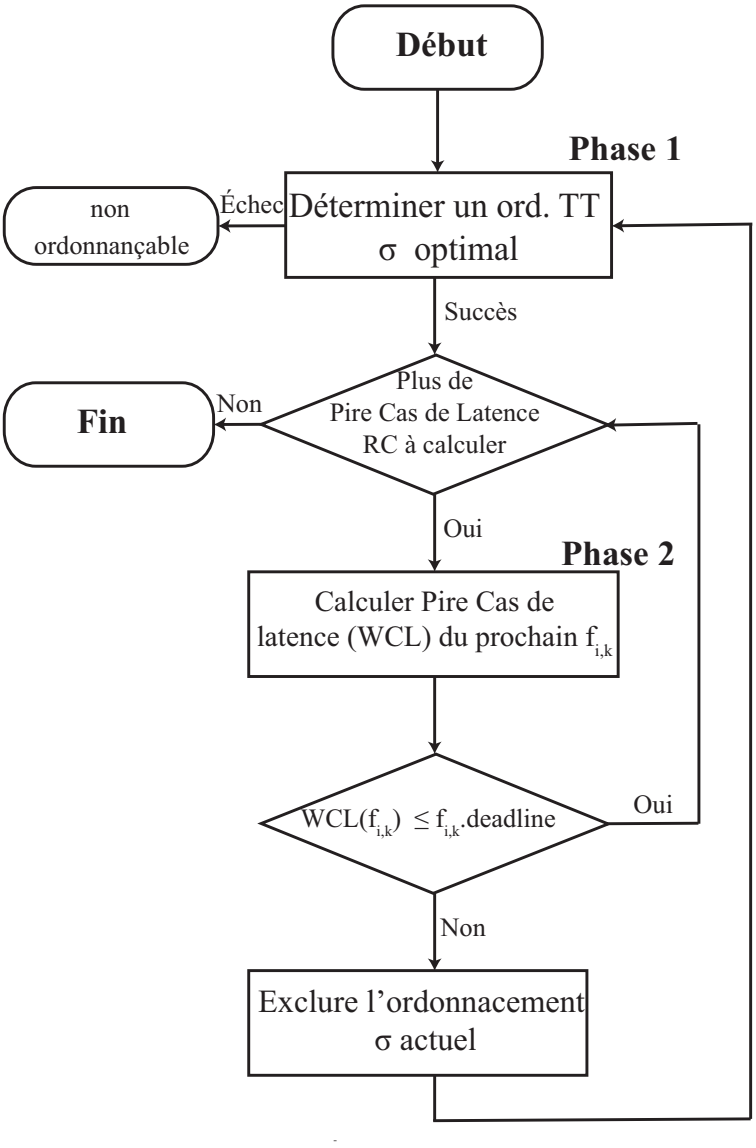


Figure 5.2 Approche d'intégration itérative

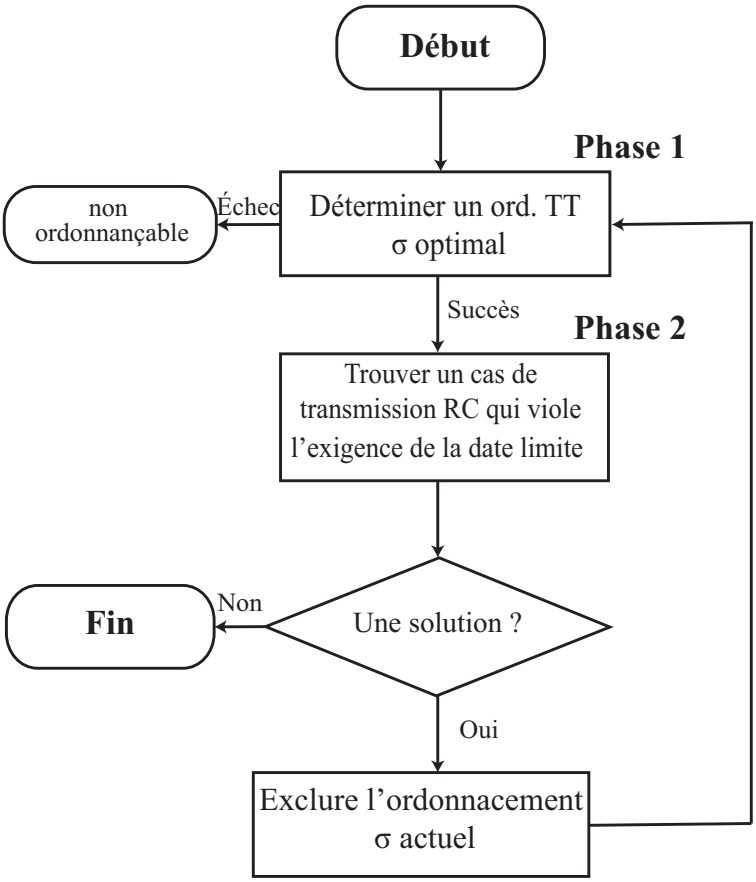


Figure 5.3 Approche d'intégration itérative améliorée

$$\textbf{minimiser } g(\sigma) \textbf{ avec}$$
$$\sigma \in \mathbb{N}^{\mathcal{F}_{TT/old} \cup \mathcal{F}_{TT/new}}, \text{avail} \in \mathbb{N}^{\Gamma_{RC}}, \text{regul} \in \mathbb{N}^{\Gamma_{RC}} \text{ et } \alpha \in \mathbb{N}^{\mathcal{F}_{RC}}$$
$$\textbf{sous contraintes}$$
$$P\acute{e}riodicit\acute{e} \wedge R\acute{e}gulation \wedge Non\_Chevauchement \wedge FIFO \wedge$$
$$\bigwedge_{i=1,2} Envoi\_Plus\_T\hat{o}t_i \wedge \bigwedge_{i=1,2} D\acute{e}pendance\_Chemin_i \wedge$$
$$Limite\_M\acute{e}moire \wedge Relai\_Simultan\acute{e} \wedge Latence \qquad \text{(P.1)}$$

Dans la deuxième phase, nous fixons l'ordonnancement optimal σ obtenu dans la phase précédente. L'objectif de cette phase est de déterminer pour chaque $f_{i,k}$, la fonction avail qui donne le pire cas de transmission noté $WCL(f_{i,k})$. En d'autres termes, nous cherchons les pires scénarios de disponibilité d'information donnée par les systèmes terminaux qui maximisent les délais de transmission. Étant donné un ordonnancement TT noté σ et la fonction avail, nous définissons pour chaque $f_{i,k} \in \Gamma_{RC}$ une fonction $g_{i,k}$ qui mesure la durée requise pour f_i pour atteindre les destinations pour la $k^{\text{ème}}$ instance comme suit.

$$g_{i,k}(\alpha) = \max_{f^l_{i,k} \in leaves(f_{i,k})} \Big(\alpha(f^l_{i,k}) + f^l_{i,k}.C - \alpha(first(f_{i,k})) \Big)$$

Considérons l'ordonnancement TT σ synthétisé dans la phase 1, la phase 2 consiste à résoudre une classe de problème qui détermine chaque $WCL(f_{i,k})$ pour chaque $f_{i,k} \in \Gamma_{RC}$ en résolvant ce problème.

$$\textbf{maximiser } g_{i,k}(\alpha) \textbf{ avec}$$
$$\alpha \in \mathbb{N}^{\mathcal{F}_{RC}}, \text{avail} \in \mathbb{N}^{\Gamma_{RC}} \text{ et regul} \in \mathbb{N}^{\Gamma_{RC}} \textit{ avec}$$
$$\textbf{sous contraintes}$$
$$P\acute{e}riodicit\acute{e} \wedge R\acute{e}gulation \wedge Non\_Chevauchement \wedge FIFO \wedge$$
$$\bigwedge_{i=1,2} Envoi\_Plus\_T\hat{o}t_i \wedge \bigwedge_{i=1,2} D\acute{e}pendance\_Chemin_i \wedge Limite\_M\acute{e}moire \wedge$$
$$Relai\_Simultan\acute{e} \wedge Latence \qquad \text{(P.2 : Ver 1)}$$

Le calcul du pire cas de transmission pour chaque flux RC présente un problème majeur de scalabilité. Le nombre de problèmes à considérer dans la phase 2 est proportionnel au nombre des flux RC. Ainsi, le plus de trames RC intégrées, le plus de problèmes qu'on a à résoudre dans la phase 2. Ceci s'accompagne bien évidemment d'une résolution plus complexe pour chaque problème de la phase 2. En effet, les instances deviennent plus larges en ajoutant davantage de flux. Nous notons aussi qu'il n'est pas exigé de calculer le pire cas de latence pour les flux RC. Ainsi, il est suffisant de déterminer un cas de transmission qui viole l'exigence de latence pour en déduire que l'ordonnancement TT dit σ n'est pas valide. Pour ces différentes raisons, nous réduisons les problèmes d'optimisation de la phase 2 en un seul problème de satisfaction de contraintes. Nous considérons ainsi l'approche d'intégration itérative améliorée donnée par la figure 5.3.

Pour déterminer un cas de transmission qui viole les exigences de latence et étant donné l'ordonnancement TT σ, nous considérons la contrainte suivante.

$$\exists \text{avail} \in \mathbb{N}^{\Gamma_{RC}}, \bigvee_{i,k} \Big(g_{i,k}(\alpha) > f_{i,k}.deadline \Big) \qquad (Dépassement\_Delai)$$

Le problème de la phase 2 pour l'approche d'intégration itérative améliorée peut ainsi être formalisé comme suit :

$$\begin{gathered} \textbf{résoudre} \\ Périodicité \wedge Régulation \wedge Non\_Chevauchement \wedge FIFO \wedge \\ \bigwedge_{i=1,2} Envoi\_Plus\_Tôt_i \wedge \bigwedge_{i=1,2} Dépendance\_Chemin_i \wedge \\ Limite\_Mémoire \wedge Relai\_Simultané \wedge Latence \wedge Dépassement\_Delai \\ \textbf{avec } \text{avail}, \text{regul} \in \mathbb{N}^{\Gamma_{RC}} \text{ et } \alpha \in \mathbb{N}^{\mathcal{F}_{RC}} \end{gathered} \qquad \text{(P.2 : Ver 2)}$$

En programmation par contraintes, l'approche d'intégration itérative peut être implémentée par deux programmes (c.-à-d. un par phase) que nous caractérisons comme suit :

Pour la première phase, nous considérons le problème $\mathcal{P}_1 = (\mathcal{X}_1, \mathcal{D}_1, \mathcal{C}_1, f)$ défini comme suit.

$$
\begin{aligned}
\mathcal{X}_1 =& \{\sigma(f^l_{i,k}) : f^l_{i,k} \in \mathcal{F}_{TT} \wedge \sigma \in \mathbb{N}^{\mathcal{F}_{TT}}, \\
& avail(f_{i,k}), regul(f_{i,k}), \alpha(f^l_{i,k}) : \\
& f_{i,k} \in \Gamma_{RC} \wedge f^l_{i,k} \in \mathcal{F}_{RC} \wedge avail, regul \in \mathbb{N}^{\Gamma_{RC}}\} \\
D(avail(f_{i,k})) =& [f_{i,k}.min\_avail, f_{i,k}.max\_avail] \\
D(x) =& [0..2 \times HP], \forall x \in \mathcal{X}_1 \backslash \{avail(f_{i,k}) : f_{i,k} \in \Gamma_{RC}\} \\
\mathcal{C}_1 =& Périodicité \bigcup Régulation \bigcup Non\_Chevauchement \bigcup FIFO \bigcup \\
& \bigcup_{i=1,2} Envoi\_Plus\_Tôt_i \bigcup \bigcup_{i=1,2} Dépendance\_Chemin_i \bigcup \\
& Limite\_Mémoire \bigcup Relai\_Simultané \bigcup Latence \\
f =& g(\sigma) \qquad (\mathcal{P}_1)
\end{aligned}
$$

Nous notons aussi que nous pouvons limiter $D(x)$ à l'intervalle $[0..2 \times HP]$ au lieu de $\mathbb{N}$ qui est suffisant pour représenter l'espace de solution sans autant éliminant des solutions candidates. Pour la deuxième phase, nous considérons le problème $\mathcal{P}_2 = (\mathcal{X}_2, \mathcal{D}_2, \mathcal{C}_2)$ défini comme suit.

$$
\begin{aligned}
\mathcal{X}_2 =& \{avail(f_{i,k}), regul(f_{i,k}), \alpha(f^l_{i,k}) : \\
& f_{i,k} \in \Gamma_{RC} \wedge f^l_{i,k} \in \mathcal{F}_{RC} \wedge avail, regul \in \mathbb{N}^{\Gamma_{RC}}\} \\
D(avail(f_{i,k})) =& [f_{i,k}.min\_avail, f_{i,k}.max\_avail] \\
D(x) =& [0..2 \times HP], \forall x \in \mathcal{X}_1 \backslash \{avail(f_{i,k}) : f_{i,k} \in \Gamma_{RC}\} \\
\mathcal{C}_2 =& Périodicité \bigcup Régulation \bigcup Non\_Chevauchement \bigcup FIFO \bigcup \\
& \bigcup_{i=1,2} Envoi\_Plus\_Tôt_i \bigcup \bigcup_{i=1,2} Dépendance\_Chemin_i \bigcup \\
& Limite\_Mémoire \bigcup Relai\_Simultané \bigcup Latence \bigcup Dépassement\_Délai \\
& \qquad (\mathcal{P}_2)
\end{aligned}
$$

Nous notons que pour les variables du problème $\mathcal{P}_2$, nous ne considérons plus les variables $\sigma(f^l_{i,k})$. L'ordonnancement optimal σ obtenu dans la phase 1 est fixé dans $\mathcal{P}_2$. Pour les contraintes, nous considérons en plus la contrainte E pour assurer l'ordonnançabilité des trames RC.

5.3.2 Approche d'intégration itérative : Heuristique de branchement

Nous introduisons dans cette section une heuristique de branchement pour améliorer les performances de résolution de la première phase de notre approche. L'heuristique est illustrée par la figure 5.4. Elle est basée sur les choix de branchement suivants.

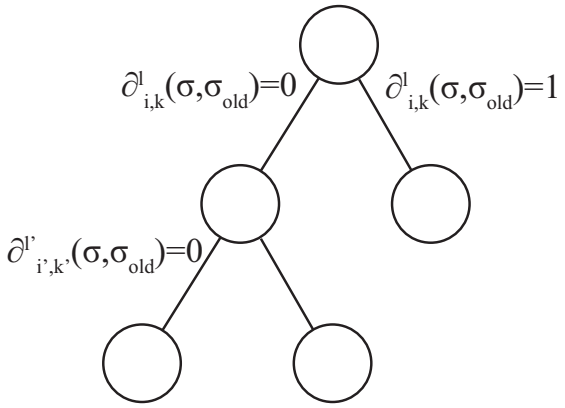


Figure 5.4 Heuristique d'intégration itérative

— *Structure d'arbre :* arbre binaire.

— *Variable de branchement :* variables de reconfiguration de l'ancien ordonnacement TT noté σ_{old}.

$$\{\partial^{l}_{i,k}(\sigma,\sigma_{old}) : \forall\sigma \in \mathbb{N}^{\mathcal{F}_{TT}}, \forall f^{l}_{i,k} \in \mathcal{F}_{TT/old}\} \tag{5.4}$$

— *Sélection des valeurs :* Pour chaque $\partial^{l}_{i,k}(\sigma,\sigma_{old})$, nous choisissons la valeur 0 avant la valeur 1.

Construisant notre arbre de recherche avec seulement les variables de reconfiguration a du sens dans notre approche. En effet, ce choix réduit énormément l'espace de recherche. Plus encore, les variables de reconfiguration sont les seules variables qui définissent notre fonction de minimisation de coût. La sélection d'une valeur pour chaque variable de reconfiguration enclenche la propagation des contraintes et réduit les valeurs possibles pour les variables d'ordonnancement. (c.-à-d. ordonnancement des instances de trames TT et RC). En explorant les valeurs 0 avant les valeurs 1 pour les variables de reconfiguration, nous favorisons en premier lieu l'exploration des cas de non-reconfiguration. Ceci permet d'orienter le solveur dès le début vers la partie d'espace de solution qui contient potentiellement la solution optimale. En effet, la fonction objectif augmente pour chaque reconfiguration.

La fonction de reconfiguration est définie sur l'ordonnancement TT noté σ. Plus précisément, elle est définie sur les instances de trame TT notées $f^l_{i,k}$. Ainsi, pour définir correctement notre heuristique de recherche et pour désigner la reconfiguration d'une instance de trame spécifique, nous devons considérer la sélection de (1) l'index de trame i, (2) l'instance de la trame k, (3) le lien de données l. Les index i et k peuvent être déterminés par la désignation d'une structure d'arbre $f_{i,k}$ dans $\mathcal{T}_{TT/old}$. Le lien de données l est plutôt déterminé par la sélection d'un nœud dans la structure d'arbre $f_{i,k}$. Une fois que ces trois éléments sont fixés, nous pouvons faire le branchement avec la variable $\partial^l_{i,k}(\sigma, \sigma_{old})$. Ceci signifie que nous ajoutons un autre niveau dans l'arbre de recherche. Évidemment, les différents nœuds de ce niveau correspondent aux valeurs possibles de $\partial^l_{i,k}(\sigma, \sigma_{old})$.

Notre heuristique est définie par l'algorithme 2. Il prend comme paramètre en entrée $\Gamma_{TT/old}$ et l'ordonnancement avant l'intégration σ_{old}. Dans la *ligne* 1, nous sélectionnons une structure d'arbre $f_{i,k}$ de $\Gamma_{TT/old}$. Dans la *ligne* 4 nous faisons le branchement suivant la structure d'arbre $f_{i,k}$ en utilisant la procédure DF_Branch dont la définition est donnée par l'algorithme 3. L'algorithme de branchement a comme premier paramètre la structure d'arbre G, second paramètre le nœud de branchement et comme dernier paramètre l'ordonnancement σ_{old}. Cet algorithme définit récursivement un parcours en profondeur de la structure $f_{i,k}$. Chaque nœud parcouru correspond à la sélection d'un lien de données l et par la suite la sélection d'une seule instance de trame $f^l_{i,k}$. Nous attribuons la valeur 0 à la variable $\partial^l_{i,k}(\sigma, \sigma_{old})$ qui correspond au choix de branchement défini dans la *ligne* 3 de l'algorithme 3. Nous attribuons la valeur 1 à la même variable $\partial^l_{i,k}(\sigma, \sigma_{old})$ dans la *ligne* 6 seulement si la sélection de la valeur 0 conduit à une inconsistance. En d'autres termes, ceci signifie que malgré que nous avons parcouru récursivement les différents liens de données fils dans la *ligne* 6, nous sommes pas capable de trouver l'ordonnancement optimal σ en attribuant que les valeurs 0 aux $\partial^l_{i,k}(\sigma)$.

ALGORITHM 2: Heuristique d'intégration itérative

IterativeIntegrationHeuristic $\Gamma_{TT/old}, \sigma_{old}$

```
1  forall f_{i,k} ∈ Γ_{TT/old} do
2      G ← f_{i,k}
3      l ← root(G)
4      DF_Branch(G, l, σ_old)
   end
```

En suivant l'heuristique définie par l'algorithme 2, nous considérons dans l'opération de recherche les structures d'arbre définies par $\Gamma_{TT/old}$. Ce choix permet une propagation efficace de l'ensemble des contraintes reliées au lien virtuel $(\cup_i C.i)$. Pour être plus précis, fixant les temps d'offset des instances de trame sur un lien donné, réduit le domaine des valeurs possibles des temps d'envois sur les prochains liens.

ALGORITHM 3: Algorithme de branchement

DF_Branch G, l, σ_{old}

1 $Mark(l)$
2 $Branch(\partial^{l}_{i,k}(\sigma, \sigma_{old}), 0)$
3 **forall** $c \in children(G, l)$ **do**
4 **if** *not Marked(c)* **then**
5 $DF\_Branch(G, c, \sigma_{old})$
end
end
6 $Branch(\partial^{l}_{i,k}(\sigma, \sigma_{old}), 1)$

5.4 Cas d'étude

Afin de mettre notre approche en évidence, nous présentons dans une première partie un cas d'étude d'intégration des flux TT et RC. Dans une deuxième partie, nous analyserons les résultats d'intégration et nous évaluerons les performances de notre heuristique.

5.4.1 Présentation

Nous considérons le même exemple de la section 4.5.2 dont l'architecture physique est donnée par la figure 4.9. Nous reprenons aussi pour ce cas d'étude les mêmes flux, mais en redéfinissant certains flux TT en flux RC La caractérisation temporelle des nouvelles trames est donnée par le tableau 5.1

Les trames sont groupées aussi en trois groupes. Chaque groupe correspond à la communication d'une application qui est intégrée en une seule étape indiquée par la première colonne. Les trames dont les identifiants sont de 1 à 21 correspondent aux trames TT et sont indiquées par la couleur bleue. Nous réservons les identifiants de 100 à 104 pour les trames RC. Ces trames sont indiquées par la couleur en rouge. La périodicité de la production de l'information est indiquée par la troisième colonne. Nous réservons la quatrième colonne pour indiquer leurs périodicités. La durée de transmission de chaque trame sur chaque lien de donnée est indiquée par la cinquième colonne. De la même façon que l'exemple 4.5.2, Nous caractérisons dans la sixième colonne nos liens virtuels (Nous utilisons la forme $l_s - \{l_{d_1}, ..., l_{d_n}\}$ où l_s est le premier lien virtuel et l_{d_1} à l_{d_n} sont les liens qui succèdent l_s). Nous utilisons du chapitre précédent la même identification des liens résumée dans le tableau 4.7.

Tableau 5.1 Caractérisation temporelle des trames intégrés

Étape d'intégration	Id i de trame	Période de production	$f_i.period$	$f_i.length$	lien virtuel
1	1	30	30	1	3-2
	2	30	30	1	3-8
	3	120	60	2	3-2
	4	120	60	4	3-8
	5	120	60	1	1-4
	6	120	60	3	7-4
	7	120	60	2	1-4
	8	120	60	1	7-4
2	9	30	30	2	3-8
	10	120	60	3	3-2
	11	120	60	7	3-8
	12	120	60	3	3-8
	13	30	30	4	7-2
	14	120	60	1	1-8
	15	120	60	2	7-4
	16	30	30	1	1-4
	17	30	30	2	7-4
	18	120	60	1	1-4
	19	120	60	1	7-4
3	20	60	60	1	9-4
	21	30	30	1	3-10
	100	60	60	11	11-{2,4,6,8,10}
	101	60	60	5	11-{2,4,6,8,10}
	102	60	60	2	11-{2,4,6,8,10}
	103	60	60	4	3-6
	104	60	60	1	5-4

5.4.2 Résultats d'intégration et observations

Nous expérimentons dans cette section trois aspects qui nous permettent d'évaluer notre approche. Comme première expérience, nous essayons d'intégrer les trames suivant trois stratégies qu'on détaille après. Nous utilisons pour ce fait l'outil de programmation par contraintes *MiniZinc* (Nethercote et al., 2007). En nous basant sur ces résultats, nous évaluons deux aspects. Nous investiguons en premier lieu s'il est judicieux de considérer une stratégie d'optimisation pour le problème d'intégration itérative. Pour le deuxième aspect, nous évaluerons la performance de notre approche d'intégration itérative. Comme l'outil MiniZinc présente plusieurs solveurs de contraintes, nous étudions les performances de ces derniers dans la résolution de notre problème.

Nous avons intégré les trames de notre exemple en trois étapes en fixant ces trois stratégies d'intégration.

1. Reconfiguration avec indifférence au coût : Le solveur a le libre choix dans la détermi-

nation des temps d'offset.Il ne considère pas les valeurs d'offsets pour les trames TT déjà intégrées.

2. Reconfiguration avec optimisation de coût : Le solveur doit trouver la solution qui optimise le coût associé à la reconfiguration des trames déjà intégrées
3. Reconfiguration non permise : Le solveur doit garder l'ordonnancement des trames TT déjà intégrées.

Les résultats d'intégration des stratégies sélectionnés sont rapportés dans l'ordre par le tableau 5.2.

Tableau 5.2 Résultats de l'intégration

	Stratégie 1	Stratégie 2	Stratégie 3
Étape 1	104 (msec)	104 (msec)	104 (msec)
Étape 2	173 (msec) 19 reconfig	231 (msec) 2 reconfig	148 (msec) Non possible
Étape 3 (Phase 1)	212 (msec) 51 reconfig	286(msec) [1] 0 reconfig	-
Étape 3 (Phase 2)	Insatisfiable 135 (msec)	Insatisfiable 135 (msec)	-

Intérêt de l'optimisation du coût de reconfiguration

Le choix de la première ou la troisième stratégie d'intégration coïncide avec un cas extrême d'intégration. Pour la première stratégie, nous n'accordons pas d'importance à l'aspect coût de reconfiguration. En comparant les résultats de la stratégie 2 qui optimise le coût de reconfiguration, nous remarquons que la stratégie 1 configure plus le réseau. Ce résultat est observé lors de la deuxième étape. Avec la stratégie 1, le solveur donne une solution avec 19 reconfigurations alors que la stratégie 2 configure seulement deux instances de trames TT déjà intégrées. Il semble que cette différence en termes de coût de reconfiguration est élargie lors que le réseau supporte plus de trafic. Ceci est visible quand nous observons le nombre de reconfigurations dans la troisième étape d'intégration. Nous notons 51 reconfigurations tandis que nous sommes capables d'intégrer les trames avec un coût nul. L'autre extrême donné par la stratégie 3 ne permet pas la reconfiguration. Cette stratégie échoue d'intégrer de nouvelles trames. Ce résultat est observé dans la deuxième étape d'intégration quand le nombre minimal de reconfigurations est 2. Dans ce cas, nous ne pouvons pas adopter la stratégie 3 pour intégrer les trames. Ceci se produit lors que la configuration actuelle est incompatible avec les nouvelles trames à intégrer. La reconfiguration est imminente dans ce cas.

Étude de performance

Dans la première étape d'intégration, comme il n'y a pas de trames qui sont déjà ordonnancées dans le réseau, la performance du solveur est la même pour les stratégies d'intégration. Nous avons obtenu les résultats dans les 104 millisecondes. Dans la deuxième phase, quand le solveur a la liberté de reconfigurer les trames TT, il produit de 19 reconfigurations dans les 173 millisecondes. Nous notons aussi que dans cette phase, tandis qu'on ordonnance que des trames TT, la performance de solveur est la même pour les 3 stratégies (Réponse approximative dans les 200 millisecondes). Ceci signifie qu'en visant un problème d'ordonnancement TT, le solveur est capable de produire un ordonnancement optimal avec les mêmes performances qu'un ordonnancement faisable. Dans la troisième étape d'intégration, les résultats deviennent plus intéressants. Nous rappelons qu'on intègre dans cette troisième étape les deux types de flux TT et RC. Ainsi, deux étapes sont exigées pour vérifier toutes les exigences. Dans cette étape d'intégration, nous adoptons notre heuristique dans la section 5.3.2 avec la stratégie d'optimisation de coût. Nous notons que les performances restent les mêmes (environ 200 millisecondes) pour trouver un ordonnancement TT. Cependant, en omettant cette heuristique, nous ne sommes pas capable de trouver une solution dans les 100 heures. En adoptant cette heuristique, le solveur donne les statistiques données par le tableau 5.3.

Tableau 5.3 Statistiques de performance de l'heuristique

Exécution	110 ms	Noeuds	156
Résolution	9 ms	Échecs	24
Variables	5379	Redémarrage	0
Propagateurs	5626	Profondeur maximale	149
Propagations	36760		

Malgré que ça prend en total 286 millisecondes pour retourner les résultats, la résolution est seulement en neuf millisecondes pour un temps d'exécution de 110 millisecondes. Le solveur met plus de temps à instancier le modèle que pour le résoudre. Le problème est assez large. Nous pouvons le remarquer par le nombre de variables 5379, le nombre de propagateurs (contraintes avec un algorithme de résolution spécifique) 5626, le nombre de contraintes propagées 36760 et la plus grande profondeur de l'arbre de recherche qui est 150. Malgré la grande taille du problème, le solveur échoue seulement 25 fois à instancier les variables. Ceci prouve l'efficience de notre heuristique pour guider le solveur vers la solution optimale.

Impact du choix du solveur

L'outil MiniZinc définie 4 solveurs G12 fd, Gecode, lazy et solveur MIP. Nous avons testé notre heuristique avec les 4 solveurs et nous avons obtenu les performances données par le tableau 5.4

Tableau 5.4 Performance de l'heuristique avec 4 différents solveurs

G12 fd	Gecode	lazy	MIP
1 s 297	286 msec	692 msec	19 h 44 min 32 s

Nous avons obtenu les meilleures performances de l'heuristique avec le solveur Gecode 286 ms. Il semble que l'heuristique marche bien aussi avec le solveur G12 fd et le solveur lazy (respectivement 692 ms et 1 s 297). Toutefois, avec le solveur MIP solver et en considérant la même heuristique, ça prend 19 heures 44 min 32 seconds pour résoudre le problème.

5.5 Conclusion

À la différence des approches analytiques conventionnelles pour la vérification des flux RC, nous avons présenté dans ce chapitre une nouvelle approche non pessimiste d'intégration itérative basée sur le formalisme de contraintes. Cette approche permet à la fois de vérifier les flux RC et de trouver un nouvel ordonnancement du système en minimisant le coût de reconfiguration.

Pour ce faire, nous avons adopté la méthodologie suivante. Nous avons d'abord formalisé à travers un ensemble de contraintes le principe de transfert de TTEthernet ainsi que les contraintes de latence. Nous avons défini par la suite notre approche à base de contraintes que nous l'avons enrichi avec une heuristique de recherche dédiée pour améliorer les performances de résolution. Nous avons montré également l'applicabilité de notre approche moyennant un exemple de réseau TTEthernet.

CHAPITRE 6 APPROCHE D'INGÉNIERIE DIRIGÉE PAR LES MODÈLES POUR LA GÉNÉRATION AUTOMATIQUE DES CONTRAINTES D'INTEGRATION ITÉRATIVE

L'architecture IMA repose sur le principe d'intégration de fonctions avioniques. Ces dernières sont conçues d'une manière indépendante et déployées par la suite sur cette architecture. Nous considérons une architecture IMA connectée à travers un réseau TTEthernet. Vu la complexité des systèmes considérés et la diversité des exigences temps réel dont ils font face, cette intégration s'avère une tâche d'ingénierie assez complexe.

Pour maitriser cette complexité de synthèse d'ordonnancement et de vérification des exigences, nous avons présenté dans les chapitres 4 et 5 des approches permettant l'intégration de multiples partitions IMA ainsi que des trames TTEthernet. Ces approches cherchent à trouver une nouvelle configuration du système (partitions et trames) qui peut nécessiter la reconfiguration des parties déjà intégrées. Ainsi, un coût supplémentaire de recertification sera attribué et notre objectif est de le minimiser. Nous considérons ce problème comme un problème d'optimisation de contraintes noté COP où nous satisfaisons non seulement un ensemble de contraintes, mais nous optimisons de plus le coût de reconfiguration

Afin d'utiliser une approche d'intégration itérative, les ingénieurs en avionique doivent spécifier à la fois le système existant ainsi que les nouvelles fonctionnalités avioniques en utilisant un ensemble de contraintes formelles. Ces contraintes peuvent être résolues d'une manière efficace en utilisant des techniques de programmation par contraintes. Le nombre et la complexité de ces contraintes augmente rapidement même pour des exemples de systèmes petits. Ainsi, l'utilisation d'une approche d'intégration itérative fait face à un problème réel de gestion de complexité de spécification des contraintes. Afin de surpasser ce problème, nous proposons dans ce chapitre une approche d'ingénierie dirigée par les modèles qui définit les abstractions nécessaires (c-à-d. méta-modèles) et définit les processus de transformation pour automatiser la génération des contraintes. Le contenu de ce chapitre fait l'objet de la publication (Beji et al., 2016).

Ce chapitre est organisé comme suit. La section 6.1 introduit notre approche d'ingénierie par les modèles. Nous consacrons les sections 6.2 et 6.3 pour présenter les méta-modèles de notre approche. Nous définissons dans la section 6.4 les processus de transformation qui génèrent le programme par contraintes pour un problème d'intégration spécifique. Cette partie est suivie dans la section 6.5 par un cas d'étude qui illustre notre approche.

6.1 Vue globale de l'approche

Nous présentons dans cette section notre approche d'ingénierie dirigée par les modèles pour générer le programme résolvant le problème d'une étape d'intégration. Comme le montre la figure 6.1, notre approche se base sur la définition de deux méta-modèles. Le premier, appelé méta-modèle de spécification d'une intégration, caractérise un problème d'intégration itérative dans un réseau TTEthernet. Le deuxième méta-modèle, appelé méta-modèle d'intégration par contraintes, définit la formalisation du programme par contraintes qui résout ce problème. Une instance du premier méta-modèle décrit un cas d'étude réel d'une étape d'intégration. Nous spécifions dans cette instance les trames configurées et comment elles sont déployées sur l'architecture réseau. Une instance du second méta-modèle modélise un programme qui résout une étape d'intégration.

Un outil de transformation qui se base sur les deux méta-modèles permet de transformer une instance du modèle de spécification d'intégration au modèle de programmation par contraintes correspondant. Ce dernier peut ainsi être transformé en code structuré suivant le langage de programmation par contraintes cible et résolu par un solveur CP pour trouver la nouvelle configuration optimale. Afin de tester notre approche, nous avons choisi le langage de MiniZinc (Group, 2016) comme langage cible. Nous spécifions dans la section suivante le méta-modèle de spécification d'intégration.

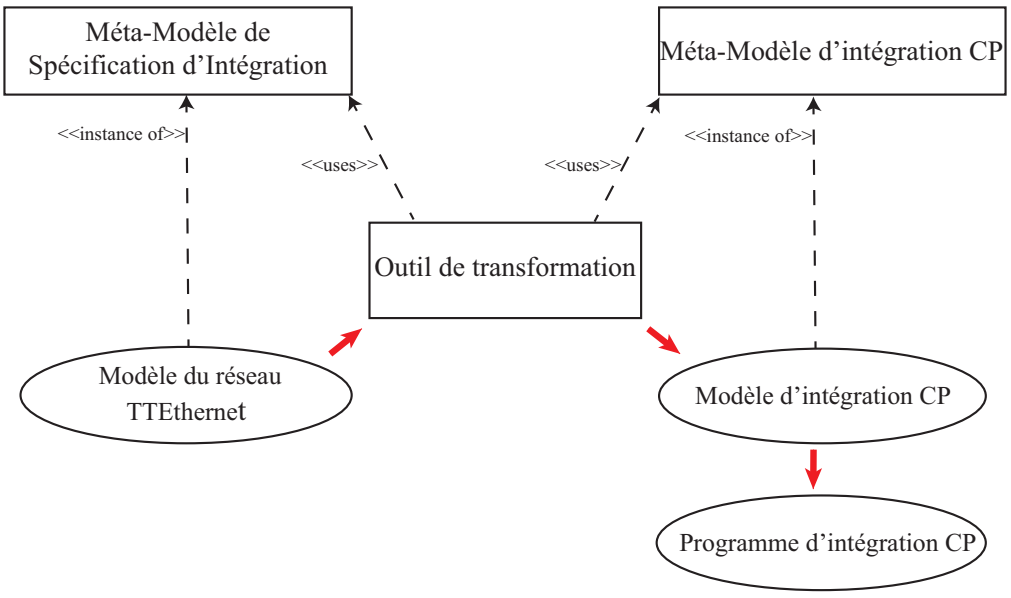


Figure 6.1 Vue d'ensemble de l'approche

6.2 Meta-Modèle de spécification d'intégration

Ce méta-modèle est spécifié par la figure 6.2. Un problème d'intégration est basé sur la définition d'étapes d'intégration qui sont définies par la méta-classe *IntegrationStep*. Cette méta-classe a un attribut *step* qui indique l'ordre de l'étape d'intégration. Elle a aussi les attributs *switchTreatmentDelay* et *HP* qui désignent respectivement le délai nécessaire pour un commutateur pour traiter une trame et l'hyper-période de toutes les périodes des trames. Une *IntegrationStep* est composée d'un ensemble de *FrameToConfigure* ou trames à configurer et un ensemble de *ConfiguredFrame* ou trames déjà configurés si elles existent. Une *ConfiguredFrame* diffère de *FrameToConfigure* par l'indication des *actualOffsets* ou Offsets actuels qui indiquent l'ordonnancement avant l'intégration de chaque instance de la trame considérée sur chaque lien de données. Une trame notée *Frame* est caractérisée par plusieurs attributs incluant la période *period*, la durée de transmission sur un lien *length* , le coût de reconfiguration *reconfigurationCost* et le nombre d'instances *nbInstances*, etc.

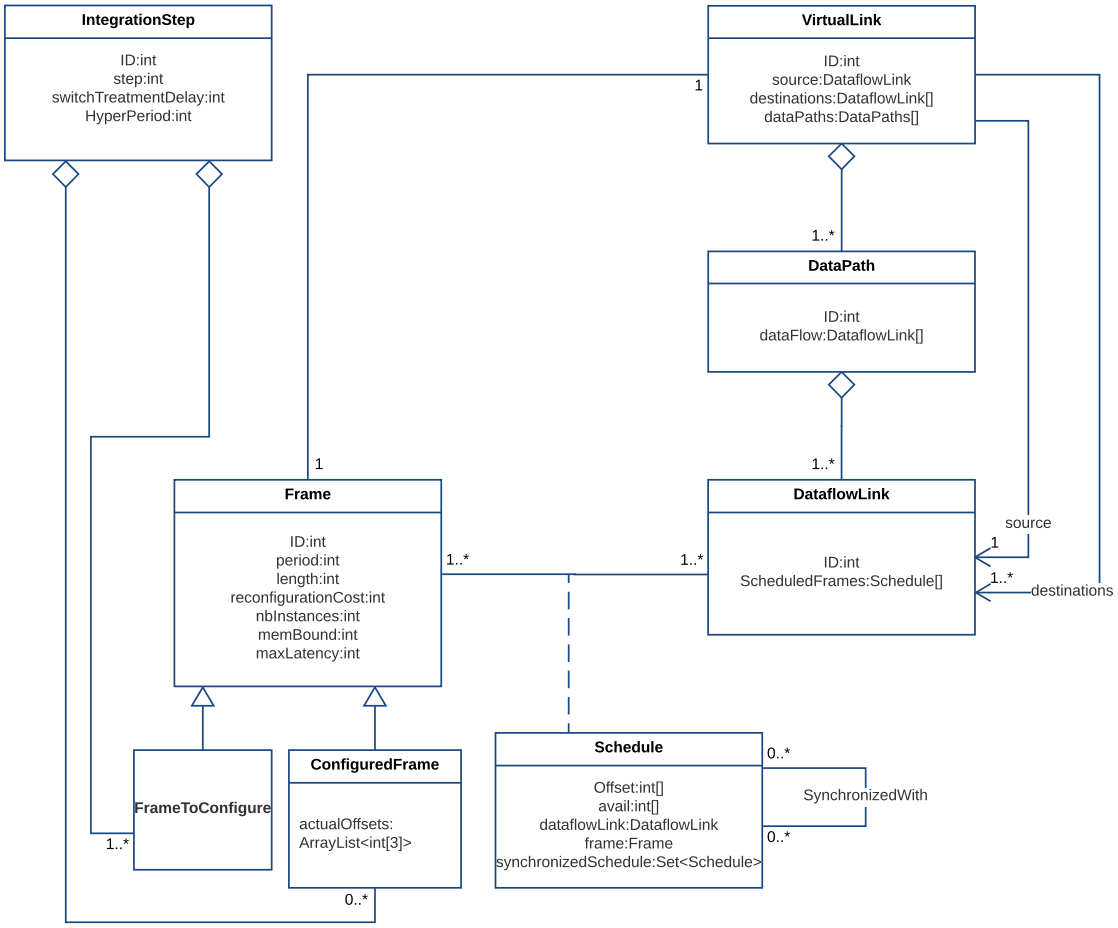


Figure 6.2 Le Méta-Modèle de Spécification d'Intégration

Un lien virtuel *VirtualLink* est associé avec chaque trame. Chaque *VirtualLink* peut être composé d'un nombre de chemins de données que nous notons chacun par *DataPath*. Chaque *DataPath* définit un chemin de la source vers une seule destination. Il est constitué par une séquence adjacente de liens de données. Un ordonnancement d'une trame (noté *Schedule*) caractérise une allocation des fenêtres de temps pour chaque trame sur chaque lien de données. Bien évidemment, plusieurs trames peuvent être ordonnancées sur un lien de données et une trame peut être transmise sur différents liens de données qui définissent le lien virtuel associé. Le *Schedule* d'une trame peut être synchronisé avec un autre de la même trame sur un autre lien de données. Ce cas se produit quand une trame doit être relayé simultanément sur différents lien de données.

6.3 Méta-modèle d'intégration CP

Notre méta-modèle d'intégration CP est illustré par la figure 6.3. Il est composé par un ensemble de méta-classes *VariableDeclaration*, un ensemble de contraintes que nous notons chacune par *Constraint* et une méta-classe *SolveItem*. Pour des raisons de lisibilité, nous n'incluons pas dans la figure 6.3 l'ensemble des relations entre ces méta-classes.

Le *SolveItem* modélise la directive pour le solveur en définissant deux attributs. Le premier attribut est *type* définit le type de problème qui peut être soit un problème de satisfaction ou de minimisation alors que le deuxième attribut *objectif* définit l'objectif à minimiser dans le cas d'un problème d'optimisation. Dans notre cas l'objectif à optimiser est le coût de reconfiguration.

6.3.1 Déclaration des variables

Comme le montre la figure 6.3, trois types de variables sont considérés pour résoudre un problème d'intégration. La méta-classe *FrameInstanceOffsets* spécifie les offsets des différentes instances d'une trame sur un lien de données. Elle a les deux attributs *name* et *type*. L'attribut *name* est de type *FrameInstanceName*. Il définit le nom de l'instance de la trame sur un lien de données. L'attribut *name* est ainsi identifié d'une manière unique par l'identifiant de la trame *IDFrame* et l'identifiant du lien *IDLink*. L'attribut *type* est de type *int[]*. La méta-classe *FrameInstanceReconfig* détecte si les offsets des instances de trame déjà configurés sont changés après l'intégration. Cette méta-classe a deux attributs *name* qui n'est que le nom de l'instance de trame sur un lien de données et l'attribut *type* de type *bool[]*. La méta-classe *ReconfigurationCost* spécifie le coût de l'ordonnancement après l'intégration.

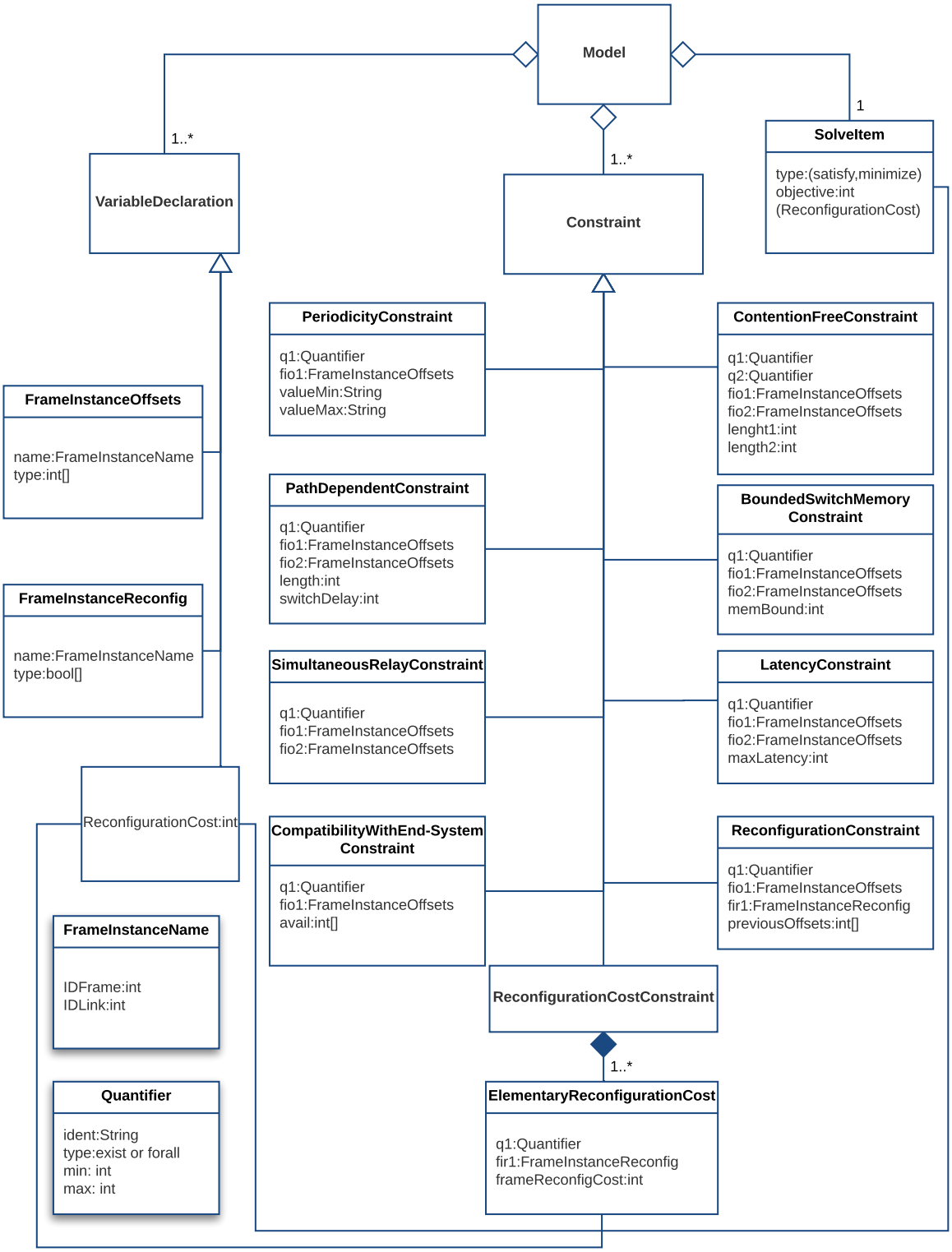


Figure 6.3 Méta-Modèle d'intégration CP

6.3.2 Contraintes

Afin de résoudre un problème d'intégration, nous considérons 9 types de contraintes illustrées par la figure 6.3. Nous introduisons pour la définition de ces contraintes un nouveau type *Quantifier* qui quantifie l'ordre de l'instance d'une trame. Ce type a quatre attributs : (1) *ident* qui identifie le nom du quantificateur, (2) *type* spécifie le type du quantificateur (ex. *exists* or *forall*), (3) *min* la valeur minimale du quantificateur et (4) *max* sa valeur maximale. Nous montrons dans cette section la représentation de quatre types de contraintes dans notre méta-modèle CP.

Contraintes de non-chevauchement

Nous rappelons qu'une contrainte de non-chevauchement exprime l'exclusion mutuelle de transmission sur un lien de données. Étant donné deux instances de trames TT, $f^l_{i,k}$ et $f^l_{j,k\prime}$ transmises sur un lien de données l, la fin de transmission de $f^l_{i,k}$ se produit avant le début de transmission $f^l_{j,k\prime}$ et vice versa. En considérant seulement les flux TT, la contrainte de non-chevauchement se réduit à cet ensemble de contraintes.

$$\begin{aligned} &\forall f^l_{i,k}, f^l_{j,k\prime} \in \mathcal{F}_{TT}, \\ &(i \neq j) \Rightarrow \quad \Big(\sigma(f^l_{i,k}) + f^l_{i,k}.C \leq \sigma(f^l_{j,k\prime})\Big) \vee \Big(\sigma(f^l_{j,k\prime}) + f^l_{j,k\prime}.C \leq \sigma(f^l_{i,k})\Big) \end{aligned} \tag{Non\_Chevauchement}$$

Dans le méta-modèle de programmation par contraintes, une contrainte de non-chevauchement se modélise par la méta-classe *ContentionFreeConstraint* qui a les attributs (1) *q1* et *q2* comme deux *Quantifier* sur les ordres d'instances respectifs k et $k^{'}$, (2) *fio1* et *fio2* les offsets respectifs de $f^l_{i,k}$ et $f^l_{j,k\prime}$ et (3) *length1* and *length2* pour désigner les durées de transmission de $f^l_{i,k}$ et $f^l_{j,k\prime}$.

Contraintes de dépendance des chemins

Nous introduisons ce type de contraintes pour exprimer la transmission séquentielle d'une trame f_i tout au long d'un chemin de données du lien virtuel vl_i. Formellement, cette

contrainte est définie comme suit :

$$\begin{aligned}
&\forall f_{i,k}^{l} \in \mathcal{F}_{TT}, \\
&\Big(\Big(f_{i,k}^{l} = root(f_{i,k}) \Rightarrow \Big(f_{i,k}.max_avail \leq \sigma(f_{i,k}^{l})\Big)\Big) \\
&\quad \wedge \Big(f_{i,k}^{l} \neq root(f_{i,k}) \Rightarrow \Big(\sigma(\text{pred}(f_{i,k}^{l})) + \text{pred}(f_{i,k}^{l}).C + tt_sw_delay\Big) \leq \sigma(f_{i,k}^{l})\Big)
\end{aligned} \tag{Dépendance_Chemin}$$

où, *tt_switch_delay* désigne le délai de traitement d'une trame TT par le commutateur. Une contrainte de dépendance de chemin est spécifiée dans le méta-modèle par la méta-classe *PathDependentConstraint* qui a les attributs suivants : (1) *q1* comme *Quantifier*sur l'ordre k de l'instance de f_i, (2) *fio1* and *fio2* les offsets respectifs de f_i^l et $f_i^{l'}$, (3) *length* le paramètre $f_{i,k}^{l}.C$ et (4) *switchDelay* le délai de traitement d'une trame TT par le commutateur réseau.

Contraintes de latence

Afin d'assurer qu'une trame respecte ses exigences de délai, nous définissons les contraintes de latence. Ces contraintes permettent de borner les délais de transmission d'une trame tout au long de ses chemins et sont formalisées comme suit.

$$\forall f_{i,k} \in \Gamma_{TT}, \forall f_{i,k}^{l} \in leaves(f_{i,k}), \Big(\sigma(f_{i,k}^{l}) + f_{i,k}^{l}.C - \sigma(root(f_{i,k}))\Big) \leq f_{i,k}.deadline \tag{Latence}$$

Nous représentons une contrainte de latence dans notre méta-modèle de programmation par contraintes par la méta-classe *LatencyConstraint* qui a les attributs (1) *q1* pour représenter l'ordre d'instance k (2) *fio1* pour désigner le temps d'offset de $root(f_{i,k})$, (3) *fio2* pour désigner le temps d'offset de $f_{i,k}^{l}$ et (4) *maxLatency* le seuil de latence maximal toléré.

Contraintes de reconfiguration

Les contraintes de reconfiguration détectent si les trames déjà configurées sont reconfigurées après l'intégration de nouvelles trames. Les contraintes de reconfiguration sont formalisées comme suit.

$$\forall f_{i,k}^{l} \in \mathcal{F}_{TT},$$
$$\Big(\sigma(f_{i,k}^{l}) \neq \sigma_{old}(f_{i,k}^{l})\Big) \Rightarrow \Big(\partial_{i,k}^{l}(\sigma, \sigma_{old}) = 0\Big) \qquad \text{(Reconfiguration)}$$

Dans le méta-modèle de programmation par contraintes, une contrainte de reconfiguration est modélisée par la méta-classe *ReconfigurationConstraint* qui a les attributs (1)*q1* pour représenter l'ordre d'instance k, (2) *fio1* pour désigner les offsets après l'intégration (3) *fir1* pour désigner les variables de reconfiguration $\partial_{i,k}^{l}(\sigma, \sigma_{old})$ et (4) *previousOffsets* qui contient les anciens offsets de f_i sur un lien de données l.

6.4 Processus de transformation de modèle

Nous détaillons dans cette section les règles de transformations implémentées dans notre outil de transformation pour générer automatiquement le modèle d'intégration CP. Nous notons par $ModelIn$ le modèle de spécification d'intégration et par $ModelOut$ le modèle d'intégration CP. Dans le reste de cette section, nous détaillons les règles de transformation correspondant pour chaque composant du modèle d'intégration CP.

6.4.1 Variables

Pour les variables du modèle de programmation par contraintes, on définit la règle de transformation donnée par l'algorithme 4 qui génère les instances *FrameInstanceOffset*. Dans *Ligne* 1, nous sélectionnons du modèle d'entrée une instance de la méta-classe *Schedule* notée par A. Nous créons dans la *Ligne* 2 une instance de la méta-classe *FrameInstanceOffsets* correspondante à A que nous dénotons par B. Dans les *Lignes* $3-5$, nous assignons les attributs.

ALGORITHM 4: Génération des instances de *FrameInstanceOffset*

forall $A = InstanceOf(ModelIn.\text{Schedule})$ **do**
  create B=***new*** $InstanceOf(ModelOut.FrameInstanceOffsets)$
  $B.type \leftarrow int[A.Frame.nbInstances]$
  $B.name.IDFrame \leftarrow A.frame.ID$
  $B.name.IDLink \leftarrow A.dataFlowLink.ID$
end

6.4.2 Contraintes

Pour les contraintes, nous présentons seulement la règle de transformation qui définit les instances de *ContentionFreeConstraint*. Cette règle est spécifiée par l'algorithme 5. Dans la *Ligne* $1-2$, nous sélectionnons du modèle d'entrée deux instances des méta-classes *Schedule*. Afin de définir correctement une contrainte de non-chevauchement, nous vérifions dans les *Lignes* $3-4$ que A et B sont deux instances qui définissent un ordonnancement de deux trames différentes sur le même lien de données. Pour assurer aussi l'unicité des contraintes, nous imposons que le *ID* d'une trame associée avec A soit inférieur à celui de B. Nous créons après dans la *Ligne* 4 une nouvelle instance C de la méta-classe *ContentionFreeConstraint*. Dans les *Lignes* $5-6$, nous définissions les deux quantificateurs de l'instance C crée. *q1* est réservé pour l'ordonnancement d'une trame associé avec l'instance A et *q2* pour celle de B. Dans les *Lignes* $7-11$, nous définissions les attributs *fio1* et *length1* qui correspondent respectivement à l'ordonnancement et au délai de transmission de la trame associée à A. D'une manière similaire, dans les *Lignes* $12-16$, nous définissions l'ordonnancement et le délai de transmission d'une trame associée avec B.

ALGORITHM 5: Génération des instances de *ContentionFreeConstraint*

```
forall A = InstanceOf(ModelIn.Schedule) do
    forall B = InstanceOf(ModelIn.Schedule) do
        if (A.frame.ID < B.frame.ID)and(A.dataflowLink = B.dataflowLink) then
            create C=new InstanceOf(ModelOut.ContentionFreeConstraint)
            C.q1 ← newQuantifier("i", forall, 1, A.Frame.nbInstances)
            C.q2 ← newQuantifier("j", forall, 1, B.Frame.nbInstances)
            forall D = InstanceOf(ModelOut.FrameInstanceOffsets) do
                if (D.name.IDFrame = A.frame.ID)and(D.name.IDLink =
                  A.dataflowLink.ID) then
                    C.fio1 ← D
                    C.length1 ← A.frame.length
                end
            end
            forall D = InstanceOf(ModelOut.FrameInstanceOffsets) do
                if (D.name.IDFrame = B.frame.ID)and(D.name.IDLink =
                  B.dataflowLink.ID) then
                    C.fio2 ← D
                    C.length2 ← B.frame.length
                end
            end
        end
    end
end
```

6.4.3 Directive de résolution SolveItem

Pour la directive de résolution *SolveItem*, on a une instance par modèle CP (c-à-d. un singleton). Nous générons cette instance en appliquant l'algorithme 6. Dans la *Ligne* 1, nous vérifions l'existence d'une trame déjà configurée. Ceci permet la définition de la nature du problème. S'il n'y a pas de trames déjà configurées qui est le cas des *Lignes* $2-3$, nous affectons la valeur *satisfy* à l'attribut *type* du problème. Dans le cas contraire, illustré dans les *Lignes* $5-6$, le problème est de type *minimize* et l'objectif désigné par l'attribut *objective* n'est que le *ReconfigurationCost* à minimiser.

ALGORITHM 6: Génération du *SolveItem*

if $(nbInstancesOf(ModelIn.IntegrationStep.ConfiguredFrame) = 0)$ **then**
  $ModelOut.SolveItem.type = satisfy$
  $ModelOut.SolveItem.objective = null$
else
  $ModelOut.SolveItem.type = minimize$
  $ModelOut.SolveItem.objective = ReconfigurationCost$
end

6.5 Cas d'étude

Dans cette section, nous illustrons notre approche dirigée par les modèles à travers l'intégration de la partie communication du système dont l'architecture physique est illustrée par la figure 6.4. Nous identifions dans cette figure les différents liens de données par des nombres et la direction de chaque flux par des lignes pointillées.

Les caractéristiques temporelles des trames sont données par le tableau 6.1. Comme le montre la première colonne de ce tableau, nous proposons d'intégrer l'ensemble des trames en trois étapes d'intégration. Les *ID*s des trames sont indiquées par la deuxième colonne. Les périodes de trames sont données par la troisième colonne. Nous réservons la quatrième colonne pour l'indication des dates de disponibilité au niveau des systèmes terminaux notés ES. La cinquième colonne indique les délais de transmission des trames sur un lien de données. Nous indiquons dans la dernière colonne la structure du lien virtuel. Nous adoptons la notation $l_s - \{l_{d_1}, ..., l_{d_n}\}$ pour indiquer que la trame associée est transmise en premier lieu sur le lien l_s et puis simultanément sur les liens l_{d_1} à l_{d_n}. Le coût de reconfiguration de chaque trame dans cet exemple est égal à 1.

En se basant sur ce cas d'étude, nous illustrons à travers deux exemples : (1) la spécification de l'intégration en entrée comme instance de *Méta-Modèle de spécification d'intégration*, (2)

Tableau 6.1 Caractéristiques temporelles des trames intégrés

Étape d'intégration	Id de trame	Période	Disponibilités	Durée sur un lien	Lien virtuel
1	3	30	[7,35,65,95]	1	3-2
	4	30	[5,33,63,93]	1	3-8
	40	60	[21,21]	2	3-2
	41	60	[15,15]	4	3-8
	42	60	[17,17]	1	1-4
	43	60	[17,17]	3	7-4
	1030	60	[9,9]	2	1-4
	1031	60	[9,9]	1	7-4
2	2	30	[9,41,67,97]	2	3-8
	10	60	[51,51]	3	3-2
	11	60	[31,31]	7	3-8
	12	60	[27,27]	3	3-8
	13	30	[25,61,75,107]	4	7-2
	14	60	[71,71]	1	1-8
	15	60	[89,89]	2	7-4
	1000	30	[17,53,89,105]	1	1-4
	1001	30	[19,55,69,101]	2	7-4
	1002	60	[63,63]	1	1-4
	1003	60	[7,7]	1	7-4
3	0	60	[35,95]	1	9-4
	1	30	[0,30,60,90]	1	3-10
	109	60	[22,82]	11	11-{2,4,6,8,10}
	110	60	[14,74]	5	11-{2,4,6,8,10}
	111	60	[7,67]	2	11-{2,4,6,8,10}
	140	60	[61,81]	4	3-6
	141	60	[44,104]	1	5-4

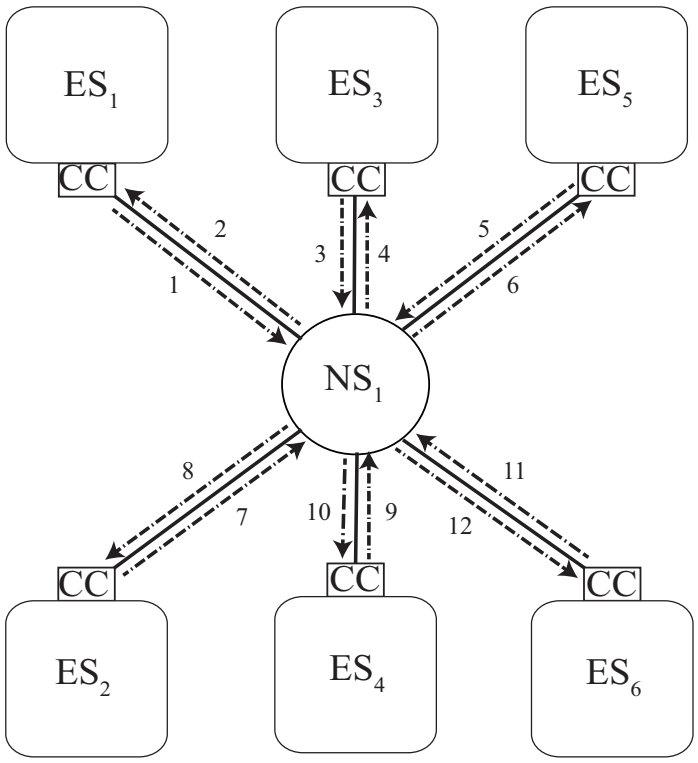


Figure 6.4 Architecture physique du système

une instance du méta-modèle d'intégration CP correspondant à la spécification en entrée et (3) le code *MiniZinc* relatif au modèle CP.

Bien que cet exemple illustre l'intégration de seulement 26 liens virtuels, nous notons que la résolution de chaque étape d'intégration nécessite environ mille lignes de code. Nous illustrons quelques aspects pertinents de notre approche à travers deux petits exemples.

Dans le premier exemple, nous fixons pour objectif de générer quelques variables CP et quelques contraintes de trames en choisissant l'exemple de contrainte de non-chevauchement. Dans le deuxième exemple, nous expliquons plutôt une contrainte qui exploite la structure du lien virtuel

6.5.1 Exemple 1

Considérant la première étape d'intégration et plus précisément l'intégration des trames f_3 et f_4, nous notons que f_3 est ordonnancé sur le lien de données avec les *ID*s 3 et 2. f_4 est ordonnancé sur le lien 3 et 8. Le modèle de spécification d'intégration dans cette partie est donné par la figure 6.5. Nous illustrons seulement les ordonnancements $s1$ et $s2$ des trames f_3 et f_4 sur le lien de données $l3$.

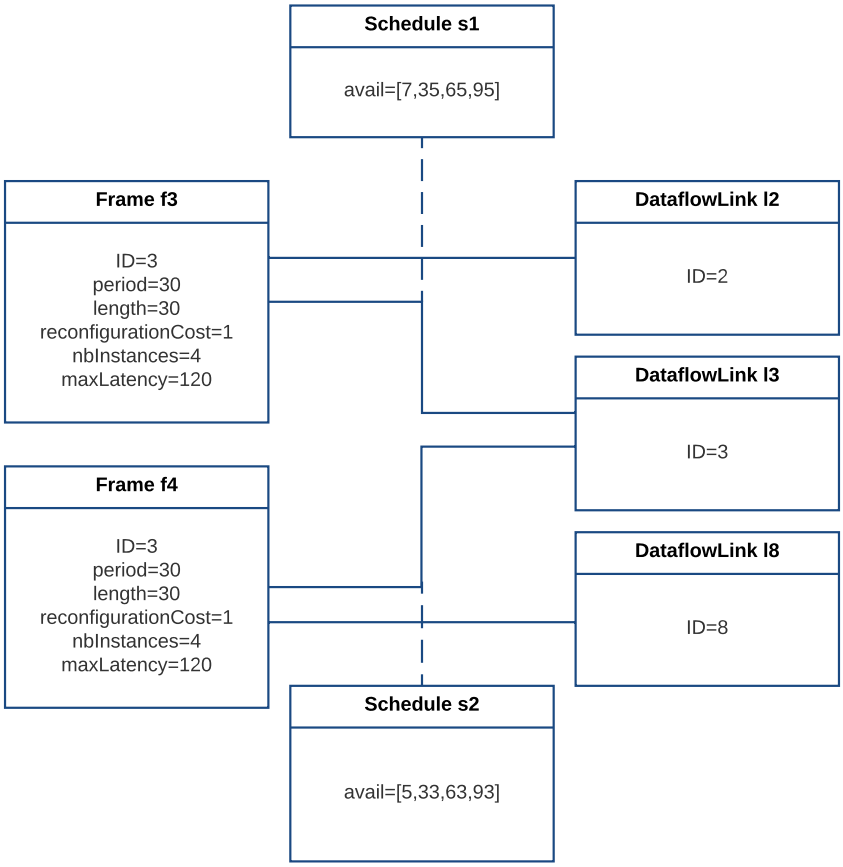


Figure 6.5 Modèle de spécification d'intégration de l'exemple 1

Comme le montre la figure 6.6, deux instances $fio1$ et $fio2$ de la méta-classe *FrameInstanceOffsets* sont définies dans notre modèle de programmation par contraintes pour représenter les ordonnancements $s1$ et $s2$. Comme le nombre des trames de f_3 considérés dans le modèle de spécification d'intégration est égal à 4, l'attribut type a comme valeur *int[4]*. Le code *MiniZinc* correspondant est donné par la figure 6.1. Nous notons que nous limitons nos valeurs d'offsets à l'intervalle [1..120] pour avoir un domaine fini assez large pour représenter les différentes valeurs possibles.

Listing 6.1 Déclaration des variables de l'exemple 1 dans *MiniZinc*

```
array[1..4] of var 0..120:Link3Offset3;
array[1..4] of var 0..120:Link3Offset4;
```

Maintenant que nous avons illustré les différentes variables du modèle d'intégration CP, nous présentons la contrainte de non-chevauchement de l'exemple 1. Cette contrainte est définie par l'instance $c1$ de la méta-classe *Contention FreeConstraint* comme le montre la figure 6.7.

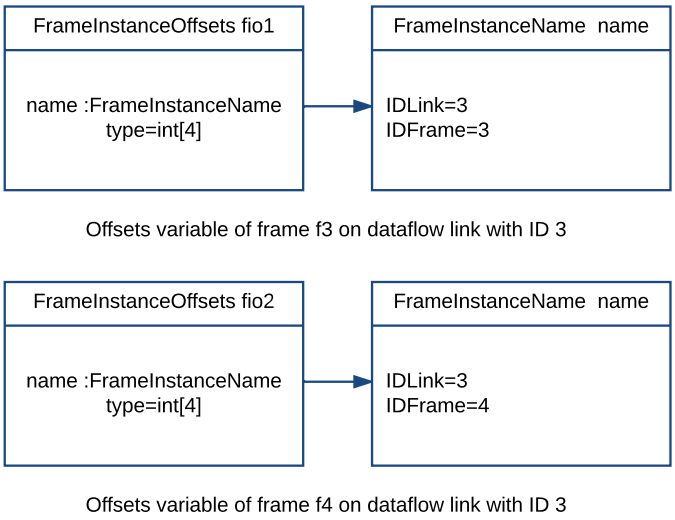


Figure 6.6 Variables CP considérées dans l'exemple 1

L'instance de $c1$ contient l'information requise pour générer le code *MiniZinc* correspondant qui est illustré par la figure 6.8

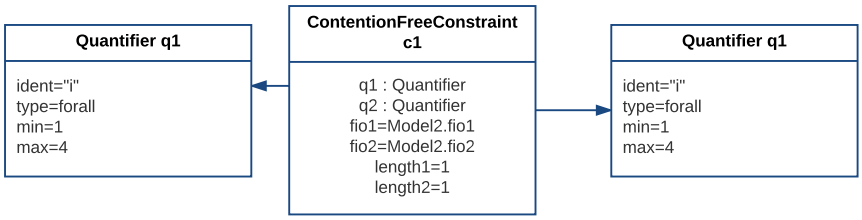


Figure 6.7 Contrainte de non-chevauchement considérée dans l'exemple 1

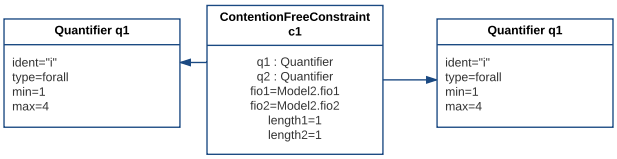


Figure 6.8 Example 1 :Code de contrainte de non-chevauchement en *MiniZinc*

6.5.2 Exemple 2

La trame f_3 est transmise sur le lien virtuel vl_3 qui est composé par le chemin $3-8$. La transmission de f_3 sur son chemin de données associé induit la définition de *Contrainte de*

dépendance de chemin qui a la structure présentée par la figure 6.9. Outre que les attributs *switchDelay* et *length*, pour définir cette contrainte, nous considérons de l'exemple 1 l'instance $fio1$ pour caractériser les offsets de f_3 sur le lien de données 2. Nous considérons aussi l'instance $fio3$ qui définit les offsets associés à la transmission de f_3 sur le lien de données 2. Le code *MiniZinc* correspondant à cette contrainte est donné par la figure 6.2

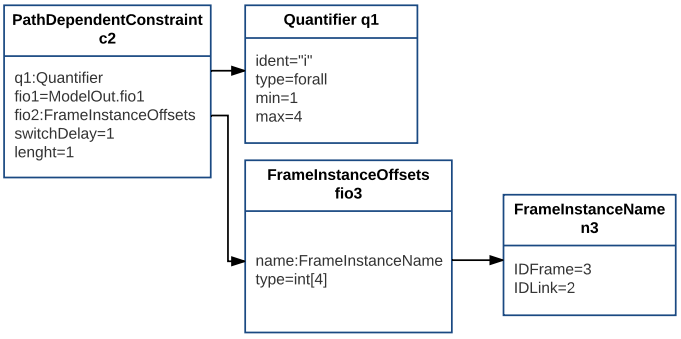


Figure 6.9 Modèle d'intégration CP de l'exemple 2

Listing 6.2 Example 2 : Code *MiniZinc* du contrainte de dépendance de chemin

```
constraint forall (i in 1..4)
(Link3Offset3[i] + 1 + 1 <= Link2Offset3[i]);
```

6.6 Conclusion

Nous avons proposé dans ce chapitre une approche d'ingénierie dirigée par les modèles pour supporter la synthèse automatique des programmes qui résolvent l'intégration des flux TT de TTEthernet. L'approche proposée se repose sur la définition de deux méta-modèles. Le premier spécifie un problème d'intégration itérative sur un réseau TTEthernet. Le deuxième décrit plutôt la structure d'un programme CP. Outre que les deux méta-modèles, cette approche est basée aussi sur la définition de processus de transformations qui automatisent la génération du modèle CP pour un problème d'intégration spécifique. Le modèle CP résultant est transformé en code CP permettant de chercher une nouvelle configuration optimale du réseau.

CHAPITRE 7 CONCLUSION

Nous avons exploré dans cette thèse l'intégration itérative des systèmes avioniques déployés sur des réseaux TTEthernet. Les systèmes avioniques modernes sont conçus principalement conformément au modèle d'architecture IMA. Cette architecture typique est basée sur le partage de ressources entre les applications critiques tout en assurant la non-interférence entre eux. Le partage de ressource permet, ainsi, d'intégrer davantage de fonctionnalités tout en gardant le poids des équipements dans les limites raisonnables. La méthodologie adoptée pour concevoir de tels systèmes est l'approche d'intégration itérative. Cette approche permet de maitriser la complexité et permet, en plus, de prendre en considération leur aspect évolutif.

Durant le processus d'intégration itérative des systèmes avioniques, les ingénieurs système peuvent reconfigurer les parties déjà intégrées afin d'assurer les exigences temps-réel dures. De telles reconfigurations impliquent un coût qui peut exprimer le coût de recertification et/ou les ressources déployées pour l'objectif de reconfiguration. Malheureusement, les approches de vérification actuelles ne prennent pas en considération l'aspect évolutif de tels systèmes et ne modélisent pas le coût. De plus, avec l'introduction de TTEthernet comme un nouveau standard, les fonctionnalités avioniques peuvent communiquer en mode synchrone et asynchrone. En mode synchrone, l'objectif est de synthétiser suivant un formalisme de contraintes un ordonnancement qui vérifie les exigences temps réel. En mode asynchrone, on fait recourt généralement aux méthodes analytiques tels le calcul réseau et l'approche pour trajectoire pour l'analyse de l'ordonnançabilité des flux.

7.1 Synthèse des travaux

Cette thèse explore le développement d'une approche d'intégration itérative qui permet d'intégrer les fonctionnalités avionique communiquant en mode synchrone et asynchrone. Cette approche prend en considération l'aspect évolutif, modélisera le coût d'intégration et enlèvera le pessimiste introduit par les approches analytiques. Cette problématique est adressée en fixant trois parties.

Dans la première partie, nous nous sommes intéressés à l'intégration de fonctionnalités avioniques déclenchées par le temps (synchrones). Nous avons développé un modèle formel des systèmes IMA et des flux déclenchés par le temps de TTEthernet. En nous basant sur ce modèle, nous avons développé un modèle formel de configuration avionique et nous avons formalisé le problème d'intégration itérative. Nous avons également défini une approche Max-Smt

partielle et à poids pondéré pour la reconfiguration optimale de fonctionnalités déclenchées par le temps. Cette approche définit deux types de contraintes SMT. Le premier type est appelé "contraintes dures". Elles assurent les exigences d'une architecture IMA déployée sur des réseaux TTEthernet. Le deuxième type de contraintes est intitulé "contraintes souples". Chaque contrainte souple modélise la reconfiguration d'une trame ou d'une partition. Ainsi, nous correspondons le coût de reconfiguration de chaque partie avec le coût de violation de la contrainte.

Dans la deuxième partie, nous nous intéressons à la considération des fonctionnalités déclenchées par les évènements (opèrent en mode asynchrone). La vérification de ces approches se fait par des méthodes analytiques qui sont adoptées par les avionneurs. Ces approches fournissent un bon compromis entre les performances et le pessimisme introduit dans l'analyse de l'ordonnacabilité. À cette fin, nous avons développé une approche à base de contraintes pour la reconfiguration optimale des flux synchrones et asynchrones. Dans ce contexte, nous avons aussi développé une heuristique de branchement dédié qui prend en considération l'ordre des variables et les différentes valeurs qui permettent d'améliorer les performances. Afin de tester cette heuristique, nous avons considéré l'intégration de deux fonctionnalités (Système du train d'atterrissage et système de contrôle de carburant). La considération d'une approche de programmation par contraintes pour la vérification des fonctionnalités asynchrones met en évidence deux contributions majeures dans le domaine. Premièrement, nous avons supprimé le pessimisme introduit par les approches analytiques. Deuxièmement, nous pouvons considérer simultanément la vérification des fonctionnalités synchrones et asynchrones. Les approches analytiques ne peuvent pas synthétiser un ordonnancement pour les fonctionnalités synchrones.

Avec l'aspect complexe des systèmes avioniques, le développement des programmes par contraintes pour l'intégration itérative est une tâche fastidieuse qui peut mener à plusieurs erreurs. Avec le contexte critique de tels systèmes, les erreurs ne sont pas tolérées. Pour cette fin, nous avons développé dans la troisième partie de cette thèse une approche d'ingénierie dirigée par les modèles pour l'automatisation de la génération des paramètres de reconfiguration directement de la spécification. Pour ce faire, nous avons automatisé la génération du programme par contraintes à partir la spécification du système.

En couvrant les trois parties de la thèse, nous avons pu développer une approche assez complète et générique qui (1) modélise l'aspect évolutif de ces systèmes (2) modélise le coût de reconfiguration (3) permet de générer un ordonnancement TT optimal du système et (4) vérifie l'ordonnançabilité des flux asynchrones d'une manière non pessimiste. Néanmoins, notre solution proposée présente certaines limites.

7.2 Limitations de la solution proposée

Le problème traité par nos approches fait partie d'un problème plus général. Ce dernier permet également de chercher aussi le plan d'allocation des partitions aux modules, le routage des liens virtuels ainsi que la fragmentation et la composition des trames tout en optimisant le coût de reconfiguration.

Les flux RC de TTEthernet, et plus précisément les flux AFDX, sont généralement ordonnancés suivant la politique FIFO (Zhang et al., 2011). C'est la politique que nous avons adopté dans notre approche. Néanmoins, le standard prévoit plus de flexibilité en spécifiant une politique d'ordonnancement avec priorité fixe.

Une autre limite que nous soulevons est en rapport avec notre approche d'ingénierie dirigée par les modèles. Notre outil de génération des contraintes couvre la génération des différentes variables et contraintes du problème d'intégration itérative. Néanmoins, notre approche se limite à la modélisation de la partie qui concerne l'intégration des flux TT. De plus, la transformation de la spécification du système aux contraintes d'intégration se fait par un algorithme ad hoc et n'utilise pas les techniques de transformation de modèles. Ceci nous mène à parler aux améliorations futures suivantes.

7.3 Améliorations futures

Une des améliorations que nous pourrons apporter à ce travail est de considérer le problème général que nous avons évoqué ci-dessus. Il s'agit dans ce cadre d'étendre notre travail sur trois axes : (1) la recherche d'un plan d'allocation des ressources (2) le routage des liens virtuels et (3) la fragmentation et composition des trames tout en optimisant le coût de reconfiguration.

Concernant la politique d'ordonnancement, nous pensons que nous pouvons évoluer notre approche vers une politique d'ordonnancement à priorité fixe. Nous pourrions dans ce cadre étendre notre modèle pour modéliser les priorités des trames et remplacer l'ensemble de contraintes FIFO par un autre ensemble de contraintes pour les priorités fixes. Il reste dans ce cadre à étudier si notre heuristique de branchement a de bonnes performances avec cette nouvelle politique et éventuellement réadapter notre heuristique pour prendre en considération les priorités. Nous pouvons étudier par la suite la possibilité de la recherche d'un plan d'assignation de priorités optimal en termes de coût.

Afin d'évoluer notre approche d'ingénierie par les modèles, nous pourrons en premier lieu étendre nos deux méta-modèles pour qu'ils couvrent l'intégration des systèmes avionique

communicants avec les deux types de flux. Par la suite, nous modéliserons les différents processus de transformation pour la partie ajoutée des deux méta-modèles. Nous pourrons explorer aussi la redéfinition de nos processus de transformations avec des langages spécialisés pour les transformations de modèles tels que l'ATL (Jouault et al., 2008) ou QVT (Kurtev, 2008). Cette amélioration nous permettra de valider les spécifications données en entrée et de détecter éventuellement les erreurs qui sont liées.

RÉFÉRENCES

M. Abuteir et R. Obermaisser, “Scheduling of rate-constrained and time-triggered traffic in multi-cluster ttethernet systems”, dans *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, Cambridge, United kingdom, 2015, pp. 239 – 245.

——, “Simulation environment for time-triggered ethernet”, dans *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*. IEEE, 2013, pp. 642–648.

A. Al Sheikh, “Resource allocation in hard real-time avionic systems : scheduling and routing problems”, Thèse de doctorat, Toulouse, INSA, 2011.

A. Al Sheikh, O. Brun, P.-E. Hladik, et B. J. Prabhu, “Strictly periodic scheduling in ima-based architectures”, *Real-Time Systems*, vol. 48, no. 4, pp. 359–386, 2012.

R. Alur et D. L. Dill, “A theory of timed automata”, *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

M. Amerion et M. Ektesabi, “A survey on scheduling and optimization techniques for static segment of flexray protocol”, dans *Proceedings of the World Congress on Engineering and Computer Science*, vol. 2, 2012.

B. Andrillon, “Contribution of integrated modular avionics of second generation for business aviation”, *SCARLETT Project ACP7-GA-2008-211439, Europe : FP7*, 2008.

ARINC, “ARINC 664, part 7 : Avionics full duplex switched ethernet (afdx)”, *AERONAUTICAL RADIO, INC.*, 2009.

——, “Integrated modular avionics (ima)”, *AERONAUTICAL RADIO, INC.*, ARINC 653, 2006.

C. Barrett, M. Deters, L. M. de Moura, A. Oliveras, et A. Stump, “6 years of SMT-COMP”, *J. Autom. Reasoning*, vol. 50, no. 3, pp. 243–277, 2013.

S. Baruah, V. Bonifaci, G. D’Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, et L. Stougie, “Scheduling real-time mixed-criticality jobs”, *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1140–1152, Aug 2012.

S. K. Baruah et S. Chakraborty, “Schedulability analysis of non-preemptive recurring real-time tasks”, dans *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th*

International. IEEE, 2006, pp. 8–pp.

H. Bauer, J.-L. Scharbarg, et C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of afdx avionics network", dans *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*. IEEE, 2009, pp. 1–8.

——, "Worst-case backlog evaluation of avionics switched ethernet networks with the trajectory approach", dans *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*. IEEE, 2012, pp. 78–87.

S. Beji, S. Hamadou, A. Gherbi, et J. Mullins, "Smt-based cost optimization approach for the integration of avionic functions in ima and ttethernet architectures", dans *Proceedings of the 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2014, pp. 165–174.

S. Beji, A. Gherbi, J. Mullins, et P.-E. Hladik, "Model-driven approach to the optimal configuration of time-triggered flows in a ttethernet network", dans *System Analysis and Modeling. Technology-Specific Aspects of Models*, J. Grabowski et S. Herbold, éds. Cham : Springer International Publishing, 2016, pp. 164–179.

S. Beji, S. Hamadou, J. Mullins, et A. Gherbi, "Iterative integration of ttethernet network flows", *International Journal of Critical Computer-Based Systems*, 2018.

——, "On the optimization of the integration of avionic functions", *Real-Time Systems*, 2018.

J. O. Berkey et P. Y. Wang, "Two-dimensional finite bin-packing algorithms", *Journal of the operational research society*, vol. 38, no. 5, pp. 423–429, 1987.

P. Bieber, F. Boniol, M. Boyer, E. Noulard, et C. Pagetti, "New challenges for future avionic architectures." *AerospaceLab*, no. 4, pp. p–1, 2012.

M. Bofill, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, et A. Rubio, "The barcelogic smt solver", dans *International Conference on Computer Aided Verification*. Springer, 2008, pp. 294–298.

F. Boniol et V. Wiels, "The Landing Gear System Case Study", dans *ABZ 2014 : The Landing Gear Case Study*, série Communications in Computer and Information Science, F. Boniol, V. Wiels, Y. Ait Ameur, et K.-D. Schewe, éds. Springer International Publishing, 2014, vol. 433, pp. 1–18.

M. Boyer et C. Fraboul, “Tightening end to end delay upper bound for afdx network calculus with rate latency fifo servers using network calculus”, dans *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*. IEEE, 2008, pp. 11–20.

X. Chang, “Network simulations with opnet”, dans *Proceedings of the 31st Conference on Winter Simulation : Simulation—a Bridge to the Future - Volume 1*, série WSC ’99. New York, NY, USA : ACM, 1999, pp. 307–314.

H. Charara et C. Fraboul, “Modelling and simulation of an avionics full duplex switched ethernet”, dans *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*. IEEE, 2005, pp. 207–212.

H. Charara, J.-L. Scharbarg, J. Ermont, et C. Fraboul, “Methods for bounding end-to-end delays on an afdx network”, dans *Real-Time Systems, 2006. 18th Euromicro Conference on*. IEEE, 2006, pp. 10–pp.

J. Chen et C. Du, “Schedulability analysis for independent partitions in integrated modular avionics systems”, dans *Progress in Informatics and Computing (PIC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 521–525.

J. Chen, C. Du, et P. Han, “Scheduling independent partitions in integrated modular avionics systems”, *PloS one*, vol. 11, no. 12, p. e0168064, 2016.

G. Chuyanov, N. Selvesyuk, et E. Zybin, “The role and importance of aircraft on-board equipment on the current aviation development stage”, Modern Avionics of Civil Aviation, 2014.

A. Cimatti, “Beyond boolean sat : Satisfiability modulo theories”, dans *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, May 2008, pp. 68–73.

F. Consortium *et al.*, “Flexray communications system-protocol specification”, *Version*, vol. 2, no. 1, pp. 198–207, 2005.

S. S. Craciunas et R. S. Oliver, “Smt-based task- and network-level static schedule generation for time-triggered networked systems”, dans *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*, série RTNS ’14. New York, NY, USA : ACM, 2014, pp. 45 :45–45 :54.

S. Craciunas et R. Oliver, “Combined task- and network-level scheduling for distributed time-triggered systems”, *Real-Time Systems*, vol. 52, no. 2, pp. 161 – 200, 2016.

R. Cruz, “A calculus for network delay 2. network analysis”, *Admissioncontrol*, vol. 14, pp. 132–141, 1991.

R. L. Cruz, “A calculus for network delay. i. network elements in isolation”, *Information Theory, IEEE Transactions on*, vol. 37, no. 1, pp. 114–131, 1991.

R. Cuninghame-Green, *Minimax algebra*. Springer-Verlag New York, 1979, vol. 166.

K. Czarnecki et S. Helsen, “Feature-based survey of model transformation approaches”, *IBM Systems Journal*, vol. 45, no. 3, pp. 621–645, 2006.

D. d. Niz, K. Lakshmanan, et R. Rajkumar, “On the scheduling of mixed-criticality real-time task sets”, dans *2009 30th IEEE Real-Time Systems Symposium*, Dec 2009, pp. 291–300.

L. M. de Moura et N. Bjørner, “Z3 : an efficient SMT solver”, dans *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, 2008, pp. 337–340.

E. Deroche, J.-L. Scharbarg, et C. Fraboul, “Mapping real-time communicating tasks on a distributed ima architecture”, dans *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, 2016, pp. 1–8.

S. Ding, N. Murakami, H. Tomiyama, et H. Takada, “A ga-based scheduling method for flexray systems”, dans *Proceedings of the 5th ACM international conference on Embedded software*. ACM, 2005, pp. 110–113.

S. Ding, X. Yin, H. Xu, et S. Zhang, “A hybrid ga-based scheduling method for static segment in flexray systems”, dans *2010 Chinese Control and Decision Conference*, May 2010, pp. 1548–1552.

B. Dutertre, “Yices manual version 2.4”, *SRI International, December*, 2015.

B. Dutertre et L. De Moura, “The yices smt solver”, *Tool paper at http ://yices. csl. sri. com/tool-paper. pdf*, vol. 2, p. 2, 2006.

C. Ekelin et J. Jonsson, “A lower-bound algorithm for minimizing network communication in real-time systems”, dans *Proceedings International Conference on Parallel Processing*, 2002, pp. 343–351.

M. Elshuber et R. Obermaisser, “Dependable and predictable time-triggered ethernet networks with cots components”, *Journal of Systems Architecture*, vol. 59, no. 9, pp. 679–690, 2013.

J. Ermont, J.-L. Scharbarg, et C. Fraboul, “Worst-case analysis of a mixed can/switched ethernet architecture”, dans *Proc. of the Real-Time and Network System Conference*, 2006.

C. Farcas, E. Farcas, I. H. Krueger, et M. Menarini, “Addressing the integration challenge for avionics and automotive systems 2014 ;from components to rich services”, *Proceedings of the IEEE*, vol. 98, no. 4, pp. 562–583, April 2010.

R. A. Fisher, “The logic of inductive inference”, *Journal of the Royal Statistical Society*, vol. 98, no. 1, pp. 39–82, 1935.

F. Frances, C. Fraboul, et J. Grieu, “Using network calculus to optimize the afdx network”, dans *ERTS 2006 : 3rd European Congress ERTS Embedded real-time software*, Toulouse, France, 2006, pp. pp. 1–8.

E. C. Gangl, “Evolution from analog to digital integration in aircraft avionics - a time of transition”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 3, pp. 1163–1170, July 2006.

R. Garside et F. J. Pighetti, “Integrating modular avionics : A new role emerges”, dans *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, Oct 2007, pp. 2.A.2–1–2.A.2–5.

D. Gaševic, D. Djuric, et V. Devedžic, *Model driven engineering and ontology development*. Springer Science & Business Media, 2009.

V. Gavrilu, D. Tma-Selicean, et P. Pop, “Fault-tolerant topology selection for ttethernet networks”, dans *Safety and Reliability of Complex Engineered Systems*, Zurich, Swaziland, 2015, pp. 4001 – 4009.

D. Goswami, M. Lukasiewycz, R. Schneider, et S. Chakraborty, “Time-triggered implementations of mixed-criticality automotive software”, dans *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 1227–1232.

J. Grieu, “Analyse et évaluation de techniques de commutation ethernet pour l’interconnexion des systèmes avioniques”, Thèse de doctorat, Institut National Polytechnique de Toulouse, 2004.

O. R. Group, “Minizinc 2.0”, http ://www.minizinc.org/ide/index.html, 02 2016.

Z. Guo et S. Baruah, “Mixed-criticality scheduling upon varying-speed multiprocessors”, dans *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*. IEEE, 2014, pp. 237–244.

J. Huang, J. O. Blech, A. Raabe, C. Buckl, et A. Knoll, “Static scheduling of a time-triggered network-on-chip based on smt solving”, dans *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2012, pp. 509–514.

K. Jeffay, D. F. Stanat, et C. U. Martel, “On non-preemptive scheduling of period and sporadic tasks”, dans *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*. IEEE, 1991, pp. 129–139.

Y. Jiang, “A basic stochastic network calculus”, *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 123–134, 2006.

F. Jouault, F. Allilaire, J. Bézivin, et I. Kurtev, “Atl : A model transformation tool”, *Science of Computer Programming*, vol. 72, no. 1, pp. 31 – 39, 2008.

G. Kemayo, F. Ridouard, H. Bauer, et P. Richard, “Optimistic problems in the trajectory approach in fifo context”, dans *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*. IEEE, 2013, pp. 1–8.

——, “Optimism due to serialization in the trajectory approach for switched ethernet networks”, dans *Proc. of Int. Conf. on Junior Researcher Workshop on Real-Time Computing (JRWRTC)*, 2013, pp. 13–16.

O. Kermia, “Timing analysis of ttethernet traffic”, *Journal of Circuits, Systems and Computers*, vol. 24, no. 9, pp. 1 550 140 (24 pp.) –, 10 2015.

Z. Kiziltan, A. Lodi, M. Milano, et F. Parisini, “Cp-based local branching”, dans *International Conference on Principles and Practice of Constraint Programming*. Springer, 2007, pp. 847–855.

H. Kopetz et G. Bauer, “The time-triggered architecture”, *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, Jan 2003.

H. Kopetz, “Event-triggered versus time-triggered real-time systems”, dans *Operating Systems of the 90s and Beyond*. Springer, 1991, pp. 86–101.

——, “The rationale for time-triggered ethernet”, dans *Real-Time Systems Symposium, 2008*. IEEE, 2008, pp. 3–11.

H. Kopetz, A. Ademaj, P. Grillinger, et K. Steinhammer, “The time-triggered ethernet (tte) design”, dans *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*. IEEE, 2005, pp. 22–33.

I. Kurtev, “State of the art of qvt : A model transformation language standard”, dans *Applications of Graph Transformations with Industrial Relevance*, A. Schürr, M. Nagl, et A. Zündorf, éds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, pp. 377–393.

M. Lauer, “Une méthode globale pour la vérification d’exigences temps réel : application à l’avionique modulaire intégrée”, Thèse de doctorat, Institut National Polytechnique de Toulouse-INPT, 2012.

M. Lauer, J. Mullins, et M. Yeddes, “Cost optimization strategy for iterative integration of multi-critical functions in ima and ttethernet architecture”, dans *Computer Software and Applications Conference Workshops (COMPSACW), 2013 IEEE 37th Annual*. IEEE, 2013, pp. 139–144.

J.-Y. Le Boudec et P. Thiran, “A short tutorial on network calculus. i. fundamental bounds in communication networks”, dans *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 4. IEEE, 2000, pp. 93–96.

Y. H. Lee, D. Kim, M. Younis, et J. Zhou, “Scheduling tool and algorithm for integrated modular avionics systems”, dans *19th DASC. 19th Digital Avionics Systems Conference. Proceedings (Cat. No.00CH37126)*, vol. 1, 2000, pp. 1–8.

Y.-H. Lee, D. Kim, M. Younis, J. Zhou, et J. McElroy, “Resource scheduling in dependable integrated modular avionics”, dans *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*. IEEE, 2000, pp. 14–23.

Y.-H. Lee, D. Kim, M. Younis, et J. Zhou, “Partition scheduling in apex runtime environment for embedded avionics software”, dans *Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236)*, Oct 1998, pp. 103–109.

H. Lhachemi, J. A. R. de Azua Ortega, D. Saussié, et G. Zhu, “Partition modeling and optimization of arinc 653 operating systems in the context of ima”, dans *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016, pp. 1–8.

X. Li, O. Cros, et L. George, “The trajectory approach for afdx fifo networks revisited and corrected”, dans *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*. IEEE, 2014, pp. 1–10.

X. Li et H. Xiong, “Modeling and analysis of integrated avionics processing systems”, dans *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*. IEEE, 2010, pp. 6–E.

C. Liu, T. Wang, C. Zhao, et H. Xiong, “Worst-case flow model of vl for worst-case delay analysis of afdx”, *Electronics letters*, vol. 48, no. 6, pp. 327–328, 2012.

M. Lukasiewycz, M. Glaß, J. Teich, et P. Milbredt, “Flexray schedule optimization of the static segment”, dans *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. ACM, 2009, pp. 363–372.

S. Luke, *Essentials of Metaheuristics*, 2e éd. Lulu, 2013.

A. K. Mackworth, “Consistency in networks of relations”, dans *Readings in Artificial Intelligence*. Elsevier, 1977, pp. 69–78.

K.-F. Man, K.-S. Tang, et S. Kwong, *Genetic algorithms : concepts and designs*. Springer Science & Business Media, 2012.

K. Marriott, P. J. Stuckey, L. Koninck, et H. Samulowitz, “A minizinc tutorial”, 2014.

S. Martin et P. Minet, “Schedulability analysis of flows scheduled with fifo : application to the expedited forwarding class”, dans *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, April 2006, pp. 8–15.

S. Martin, “Maîtrise de la dimension temporelle de la qualité de service dans les réseaux”, Thèse de doctorat, Université Paris XII Val de Marne, 2004.

A. Metzner, M. Fränzle, C. Herde, et I. Stierand, “Scheduling distributed real-time systems by satisfiability checking”, dans *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005), 17-19 August 2005, Hong Kong, China*. IEEE Computer Society, 2005, pp. 409–415.

J.-P. Moreaux, “Data transmission system for aircraft”, Août 2 2005.

A. Murshed, R. Obermaisser, H. Ahmadian, et A. Khalifeh, “Scheduling and allocation of time-triggered and event-triggered services for multi-core processors with networks-on-a-chip”, dans *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, July 2015, pp. 1424–1431.

M. Y. Nam, J. Lee, K. J. Park, L. Sha, et K. Kang, “Guaranteeing the end-to-end latency of an ima system with an increasing workload”, *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1460–1473, June 2014.

N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, et G. Tack, “Minizinc : Towards a standard cp modelling language”, dans *International Conference on Principles and Practice of Constraint Programming*. Springer, 2007, pp. 529–543.

T. Noll, “Safety, dependability and performance analysis of aerospace systems”, dans *International Workshop on Formal Techniques for Safety-Critical Systems*. Springer, 2014, pp. 17–31.

M. Paulitsch, H. Ruess, et M. Sorea, *Non-functional Avionics Requirements*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, pp. 369–384.

P. G. Peon, E. Uhlemann, W. Steiner, et M. Bjorkman, “A wireless mac method with support for heterogeneous data traffic”, dans *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Yokohama, Japan, 2015, pp. 3869 – 3874.

F. Pozo, G. Rodriguez-Navas, W. Steiner, et H. Hansson, “Period-aware segmented synthesis of schedules for multi-hop time-triggered networks”, dans *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug 2016, pp. 170–175.

F. Pozo, G. Rodriguez-Navas, H. Hansson, et W. Steiner, “Smt-based synthesis of ttethernet schedules : A performance study”, dans *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, Siegen, Germany, 2015, pp. 162 – 165.

F. Pozo, W. Steiner, G. Rodriguez-Navas, et H. Hansson, “A decomposition approach for smt-based schedule synthesis for time-triggered networks”, dans *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015, pp. 1–8.

F. M. Pozo Pérez, G. Rodriguez-Navas, H. Hansson, et W. Steiner, “Schedule synthesis for next generation time-triggered networks”, 2017.

F. Ren, T. Zhao, et H. Wang, “Formal specification and risk assessment approach of integrated complex system : A case study in ima domain”, dans *2015 First International Conference on Reliability Systems Engineering (ICRSE)*, Oct 2015, pp. 1–6.

F. Ridouard, J. Scharbarg, et C. Fraboul, “Stochastic network calculus for end-to-end delays distribution evaluation on an avionics switched ethernet”, dans *Industrial Informatics, 2007 5th IEEE International Conference on*, vol. 1. IEEE, 2007, pp. 559–564.

T. Robati, A. Gherbi, et J. Mullins, “A modeling and verification approach to the design of distributed {IMA} architectures using {TTEthernet}”, *Procedia Computer Science*, vol. 83, pp. 229 – 236, 2016.

F. Rossi, P. Van Beek, et T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.

A. Rutle, K. I. F. Simonsen, H. G. Schaathun, et R. Kirchhoff, “Model-driven software engineering in practice : A content analysis software for health reform agreements”, *Procedia Computer Science*, vol. 63, pp. 545 – 552, 2015.

SAE, “AS6802 : Time-triggered ethernet (ttethernet)”, *SAE Aerospace*, 2011.

L. Sagaspe, “Constraint-based design and allocation of shared avionics resources”, dans *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, Oct 2007, pp. 2.A.5–1–2.A.5–10.

F. Sagstetter, M. Lukasiewycz, et S. Chakraborty, “Schedule integration for time-triggered systems”, dans *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2013, pp. 53–58.

F. Sagstetter, P. Waszecki, S. Steinhorst, M. Lukasiewycz, et S. Chakraborty, “Multischedule synthesis for variant management in automotive time-triggered systems”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 637–650, April 2016.

K. Schmidt et E. G. Schmidt, “Optimal message scheduling for the static segment of flexray”, dans *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*. IEEE, 2010, pp. 1–5.

A. Schrijver, *Theory of Linear and Integer Programming*. New York, NY, USA : John Wiley & Sons, Inc., 1986.

L. Sha, R. Rajkumar, et J. P. Lehoczky, “Priority inheritance protocols : an approach to real-time synchronization”, *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, Sep 1990.

C. Si, S. Wang, et B. Liu, “A model driven multi-constraint safety analysis method for integrated modular avionics systems on time domain”, dans *2015 Prognostics and System Health Management Conference (PHM)*, Oct 2015, pp. 1–6.

F. Smirnov, M. Glass, F. Reimann, et J. Teich, “Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks”, dans *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2017, pp. 1–6.

W. Steiner, “An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks”, dans *2010 31st IEEE Real-Time Systems Symposium*, Nov 2010, pp. 375–384.

——, “Synthesis of static communication schedules for mixed-criticality systems”, dans *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, March 2011, pp. 11–18.

W. Steiner et B. Dutertre, “Smt-based formal verification of a ttethernet synchronization function”, dans *Formal Methods for Industrial Critical Systems*. Springer, 2010, pp. 148–163.

——, “Automated formal verification of the ttethernet synchronization quality”, dans *NASA Formal Methods*. Springer, 2011, pp. 375–390.

W. Steiner, G. Bauer, B. Hall, M. Paulitsch, et S. Varadarajan, “Ttethernet dataflow concept”, dans *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*. IEEE, 2009, pp. 319–322.

K. Steinhammer, P. Grillinger, A. Ademaj, et H. Kopetz, “A time-triggered ethernet (tte) switch”, dans *Proceedings of the conference on Design, automation and test in Europe : Proceedings*. European Design and Automation Association, 2006, pp. 794–799.

E. Suethanuwong, “Performance evaluation and optimization of standard ethernet traffic in ttethernet systems”, Thèse de doctorat, Vienna University of Technology, 2011.

E. Suethanuwong et S. Hirunmutrapom, “Searching of best-effort messages in ttethemet switches during the timely blocking intervals”, dans *2016 IEEE RIVF International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, Piscataway, NJ, USA, 2016, pp. 102 – 107.

E. Suethanuwong, “Scheduling time-triggered traffic in ttethernet systems”, dans *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE, 2012, pp. 1–4.

Z. Sun, H. Li, M. Yao, et N. Li, “Scheduling optimization techniques for flexray using constraint-programming”, dans *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 931–936.

D. Tamas-Selicean et P. Pop, “Optimization of ttethernet networks to support best-effort traffic”, dans *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–4.

D. Tamas-Selicean, S. Marinescu, et P. Pop, “Analysis and optimization of mixed-criticality applications on partitioned distributed architectures”, dans *7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012*. IET, 2012.

D. Tamas-Selicean, P. Pop, et W. Steiner, “Synthesis of communication schedules for ttethernet-based mixed-criticality systems”, dans *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2012, pp. 473–482.

D. Tămaş-Selicean, P. Pop, et W. Steiner, “Design optimization of ttethernet-based distributed real-time systems”, *Real-Time Systems*, vol. 51, no. 1, pp. 1–35, 2015.

D. TamasSelicean, P. Pop, et W. Steiner, “Timing analysis of rate constrained traffic for the ttethernet communication protocol”, dans *Real-Time Distributed Computing (ISORC), 2015 IEEE 18th International Symposium on*. IEEE, 2015, pp. 119–126.

——, “Timing analysis of rate constrained traffic for the ttethernet communication protocol”, dans *Real-Time Distributed Computing (ISORC), 2015 IEEE 18th International Symposium on*. IEEE, 2015, pp. 119–126.

X. Tao, Y. Zhu, Y. Mao, H. Song, M. Liu, X. Liu, W. Sheng, et W. Shi, “Designing ARINC653 Partition Constrained Scheduling for Secure Real Time Embedded Avionics”, dans *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE. IEEE, 2015, Proceedings Paper, pp. 213–217.

K. Tindell et J. Clark, “Holistic schedulability analysis for distributed hard real-time systems”, *Microprocessing and microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

A. Trindade et L. C. Cordeiro, “Applying smt-based verification to hardware/software partitioning in embedded systems”, *Design Autom. for Emb. Sys.*, vol. 20, no. 1, pp. 1–19, 2016.

S. Vestal, “Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance”, dans *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, Dec 2007, pp. 239–243.

M. T. B. Waez, A. Wąsowski, J. Dingel, et K. Rudie, “Synthesis of a reconfiguration service for mixed-criticality multi-core systems : An experience report”, dans *International Workshop on Formal Aspects of Component Software*. Springer, 2014, pp. 162–180.

F. R. Wagner, F. A. Nascimento, et M. F. Oliveira, “Model-driven engineering of complex embedded systems : concepts and tools”, *Porto Alegre, RS : Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS)/Cooperative Computing and Communication Laboratory (C-LAB), University of Paderborn, Paderborn, Germany*, 2011.

W. Wajs et W. P., “Time-triggered and event-triggered real-time computing for distributed control systems”, *IFAC Proceedings Volumes*, vol. 36, no. 1, pp. 109–112, feb 2003.

G. Wang, “Integration technology for avionics system”, dans *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*. IEEE, 2012, pp. 7C6–1–7C6–9.

C. B. Watkins et R. Walter, “Transitioning from federated avionics architectures to integrated modular avionics”, dans *Digital Avionics Systems Conference, 2007. DASC’07. IEEE/AIAA 26th*. IEEE, 2007, pp. 2.A.1–1–2.A.1–10.

Z. Wu, T. Lv, X. Wang, et N. Huang, “The buffer size assignment of afdx based on network calculus”, dans *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on*. IEEE, 2011, pp. 1319–1323.

H. Zeng, M. D. Natale, A. Ghosal, et A. Sangiovanni-Vincentelli, “Schedule optimization of time-triggered systems communicating over the flexray static segment”, *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, pp. 1–17, Feb 2011.

X. Zhang, X. Chen, L. Zhang, G. Xin, et T. Xu, “End-to-end delay analysis of avionics full duplex switched ethernet with different flow scheduling scheme”, dans *Proceedings of 2011*

International Conference on Computer Science and Network Technology, vol. 4, Dec 2011, pp. 2252–2258.

X. Zhang et Y. Wang, "Research of afdx network delay based on modified network calculus", dans *Network Infrastructure and Digital Content (IC-NIDC), 2012 3rd IEEE International Conference on*. IEEE, 2012, pp. 178–181.

L. Zhao, H. Xiong, Z. Zheng, et Q. Li, "Improving worst-case latency analysis for rate-constrained traffic in the time-triggered ethernet network", *Communications Letters, IEEE*, vol. 18, no. 11, pp. 1927–1930, 2014.

L. Zhao, P. Pop, Q. Li, J. Chen, et H. Xiong, "Timing analysis of rate-constrained traffic in ttethernet using network calculus", *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, Mar 2017.

L. Zheng, X. Bao, et T. Zhao, "A safety analysis research of resource process integration for ima system", dans *2015 Prognostics and System Health Management Conference (PHM)*, Oct 2015, pp. 1–6.

R. Zurawski, "Time-triggered ethernet", dans *The industrial communication technology handbook*. CRC Press, 2005, ch. Time-Triggered Ethernet, pp. 181–221.

ANNEXE A EXEMPLE ILLUSTRATIF DE CODE SMT POUR L'INTÉGRATION DES FONCTIONNALITÉS AVIONIQUE COMMUNIQUANT EN MODE TT

Nous illustrons dans cette partie l'intégration itérative des fonctionnalités avioniques communiquant en mode TT en se basant sur le formalisme SMT. Nous prenons comme langage de référence celui de l'outil *YICES* (Dutertre and De Moura, 2006; Dutertre, 2015). Un Code *YICES* est structuré en 3 parties : (1) une première partie pour la déclaration des différentes variables et paramètres du modèle, (2) une deuxième partie qui modélise les contraintes du problème en se basant sur les paramètres et les variables définies dans la première partie et (3) une dernière partie pour l'indication des directives de résolution au solveur. Pour notre problème d'intégration itérative, nous illustrons les trois parties à travers des exemples pris du système du train d'atterrissage et système de contrôle de carburant présenté dans la section 4.5.2.

A.1 Déclaration des variables

Notre modèle contient deux catégories de variables. La première catégorie représente les offsets des différentes instances des trames. Un exemple serait le suivant.

```
;; Offsets de f3 sur le lien 3
(define Link3Offset3_0::int )
(define Link3Offset3_1::int )
(define Link3Offset3_2::int )
(define Link3Offset3_3::int )
```

Cette partie de code définit quatre variables d'offsets de la trame f_3 sur le lien d'id 3. Un indice à la fin de chaque nom de variable (0 à 3) identifie dans l'ordre les différentes instances de f_3 sur le lien d'id 3. Le deuxième type de variables identifie les offsets des instances des partitions. À titre d'exemple, les différentes variables d'offset pour la partition *FDE* seraient le suivant.

```
;; Offsets de FDE
(define oFDE_0::int)
(define oFDE_1::int)
(define oFDE_2::int)
(define oFDE_3::int)
```

Nous illustrons dans la prochaine section quelques exemples pour la déclaration des contraintes d'intégration du système du train d'atterrissage et système de contrôle de carburant.

A.2 Declaration des contraintes

Les contraintes dures dans le langage de *YICES* sont déclarées par la mention *assert*. Les instances de la partition *FDE* sont exécutées 4 fois durant la *MAF* avec une période de 30. Les contraintes de périodicité de la partition *FDE* peuvent ainsi être exprimées comme suit.

```
;; Contraintes de periodicite des partitions
(assert (<= 0 oFDE_0))
(assert (<= oFDE_0 30))
(assert (<= 30 oFDE_1))
(assert (<= oFDE_1 60))
(assert (<= 60 oFDE_2))
(assert (<= oFDE_2 90))
(assert (<= 90 oFDE_3))
(assert (<= oFDE_3 120))
```

L'exemple suivant illustre le non-chevauchement d'exécution de la partition *FDE* avec la première instance de la partition *LGFAE*. Les durées respectives d'exécution de ces partitions sont 5 et 10. Ils sont codés directement dans cet ensemble de contraintes.

```
;; Contraintes de non-chevauchement des partitions
(assert (or (<= (+ oFDE_0 5) oLGFAE_0)(<= (+ oLGFAE_0 10) oFDE_0)))
(assert (or (<= (+ oFDE_1 5) oLGFAE_0)(<= (+ oLGFAE_0 10) oFDE_1)))
(assert (or (<= (+ oFDE_2 5) oLGFAE_0)(<= (+ oLGFAE_0 10) oFDE_2)))
(assert (or (<= (+ oFDE_3 5) oLGFAE_0)(<= (+ oLGFAE_0 10) oFDE_3)))
```

La trame f_3 transporte le message de la partition *FDE*. Le lien d'id 3 est le premier lien du lien virtuel vl_3. La compatibilité des ordonnancements IMA et TTEthernet pour cette communication peut être traduite par le code suivant.

```
;; Contraintes de compatibilite IMA TTEthernet
(assert (<= (+ oFDE_0 5) Link3Offset3_0))
(assert (<=  Link3Offset3_0 (+ oFDE_1 5)))
(assert (<= (+ oFDE_1 5) Link3Offset3_1))
(assert (<=  Link3Offset3_1 (+ oFDE_2 5)))
```

Le non-chevauchement de la première instance de f_3 avec les différentes instances de la trame f_4 sur le lien d'id 3 peuvent être codées de la façon suivante.

```
;; Contraintes de non chevauchement des trames
(assert ( or ( <= (+ Link3Offset3_0 1)Link3Offset4_0 )
( <= (+ Link3Offset4_0 1)Link3Offset3_0 )))
(assert ( or ( <= (+ Link3Offset3_0 1)Link3Offset4_1 )
( <= (+ Link3Offset4_1 1)Link3Offset3_0 )))
(assert ( or ( <= (+ Link3Offset3_0 1)Link3Offset4_2 )
( <= (+ Link3Offset4_2 1)Link3Offset3_0 )))
(assert ( or ( <= (+ Link3Offset3_0 1)Link3Offset4_3 )
( <= (+ Link3Offset4_3 1)Link3Offset3_0 )))
```

La trame f_3 est transmise sur le lien d'id 3 puis sur le lien d'id 2. En supposant une durée de traitement d'un commutateur TTEthernet égale à 1 et en sachant que la durée de transmission de f_3 sur un lien de données est égale à 1. Les contraintes relatives à cette dépendance de chemin peuvent s'écrire comme suit.

```
;; Contraintes de dependance de chemin
(assert (<= (+ (+ Link3Offset3_0 1) 1) Link2Offset3_0))
(assert (<= (+ (+ Link3Offset3_1 1) 1) Link2Offset3_1))
(assert (<= (+ (+ Link3Offset3_2 1) 1) Link2Offset3_2))
(assert (<= (+ (+ Link3Offset3_3 1) 1) Link2Offset3_3))
```

Le délai entre le début de transmission de f_3 sur le lien d'id 3 et le lien suivant d'id 2 doit être borné à 30 pour exprimer la limite de la capacité de mémoire relayant f_3 du lien 3 au lien 2. Ainsi la limite de mémoire du commutateur pour le relai de la trame f_3 peut être exprimée comme suit.

```
;; Contraintes de limite de memoire
(assert (<= (- Link3Offset3_0 Link2Offset3_0) 30 ))
(assert (<= (- Link3Offset3_1 Link2Offset3_1) 30 ))
(assert (<= (- Link3Offset3_2 Link2Offset3_2) 30 ))
(assert (<= (- Link3Offset3_3 Link2Offset3_3) 30 ))
```

La partition *FDE* communique avec la partition NLG à travers la trame f_3. La trame f_3 est transmise sur le lien f_3 puis sur le lien 2 avant d'arriver à sa destination. Une durée limite de 90 est fixée du moment du début d'exécution de la partition source *FDE* jusqu'à la fin d'exécution de la partition destination *NLG*. La contrainte de latence concernant les deux premières instances de ces deux partitions peut s'exprimer comme suit.

```
;; Contraintes de latence de bout-en-bout
(assert (=> (and
(<= (+ oFDE_1 0 5)(+ Link3Offset3_1 0))
(<= (+ Link3Offset3_1 0)(+ oFDE_2 0 5))
```

```
(<= (+ oNLG_-1 0 )(+ Link2Offset3_1 0 1))
(<= (+ Link2Offset3_1 0 1)(+ oNLG_0 0 ))
)
(<= (- (+ oNLG_0 0 10)(+ oFDE_1 0 )) 90 )))
```

Les contraintes souples sont identifiées dans le langage de *YICES* par la mention de *assert+*, l'expression booléenne de la contrainte puis le poids de cette contrainte. Les offsets des différentes instances de la partition *FDE* sont déjà configurées respectivement à 0, 30, 62 et 92. En considérant que le coût de reconfiguration de chaque instance de *FDE* est estimée à 7, les contraintes souples de reconfiguration des instances de la partition *FDE* peuvent ainsi être exprimées comme suit.

```
;; Contraintes Souples pour partitions
(assert+ (= oFDE_0 0) 7)
(assert+ (= oFDE_1 30) 7)
(assert+ (= oFDE_2 62) 7)
(assert+ (= oFDE_3 92) 7)
```

Les offsets des différentes instances de la trame f_3 sont déjà configurées respectivement à 5, 36, 68 et 97. En considérant un coût de reconfiguration de chaque instance de trame sur un lien de données est égale à 1, les contraintes de reconfiguration des instances de f_3 sur le lien d'id 3 peuvent être exprimées comme suit.

```
;; Contraintes Souples pour trames
(assert+ (= Link3Offset3_0 5) 1)
(assert+ (= Link3Offset3_1 36) 1)
(assert+ (= Link3Offset3_2 68) 1)
(assert+ (= Link3Offset3_3 97) 1)
```

A.3 Directive de résolution

La résolution par le solveur *SMT* en général est exprimé par le moyen d'une directive de résolution. Pour le solveur *YICES*, on initie la résolution de type *WPMS* (c-à-d. minimistion de la somme des poids des contraintes souples violées) en indiquant la directive suivante.

```
(max-sat)
```

ANNEXE B EXEMPLE ILLUSTRATIF DE CODE DE PROGRAMMATION PAR CONTRAINTES POUR L'INTÉGRATION ITÉRATIVE DES FLUX DÉCLENCHÉS PAR LE TEMPS ET À QUOTAS LIMITÉS

Nous illustrons dans cette partie l'intégration itérative des flux TT et RC en se basant sur le formalisme de programmation par contraintes. Nous prenons comme langage de référence celui de l'outil *MiniZinc* (Nethercote et al., 2007; Marriott et al., 2014).

D'une manière similaire à un programme SMT, un programme par contraintes et structuré en trois parties : (1) déclaration des variables et des paramètres (2) déclaration des contraintes et (3) déclaration des directives de résolution. Une autre partie s'ajoute par rapport à un programme SMT et qui permet de définir la fonction objectif en considérant les variables déclarées. Pour illustrer ses différentes parties, nous nous basons dans cette annexe sur des exemples pris du cas d'étude de la section 5.4. Nous commençons par illustrer des exemples de notre partie de déclaration des variables.

B.1 Déclaration des variables

L'hyper période des trames est de 60. Afin de considérerun domaine fini et assez large pour considérer toutes les solutions possibles, nous nous limitons au domaine $[0..2 \times HP]$ pour toutes les variables du domaine. Nous présentons en premier les variables d'offsets.

Le premier type de variables que nous déclarons modélise les différents offsets des trames TT. On regroupe les différentes instances d'une trame sur un lien dans un tableau. La mention de *array* et *var* indiquent respectivement qu'il s'agit de tableau et de déclaration des variables. Un exemple de déclarations des variables d'offsets des différentes instances de la trame f_3 sur le lien d'id 3 serait le suivant.

```
% Variables des offsets
array[1..4]  of var  0..120:Link3Offset3;
```

Afin de détecter les reconfigurations des différentes instances de trames TT, nous définissons les variables de reconfiguration. Les variables des différentes instances sur un même lien sont regroupées ensemble et ont la forme suivante *IsLink<idLien>Offset<idTrame>Reconfig*. L'exemple suivant illustre les variables de reconfiguration pour les instances de la trame f_3 sur le lien 3 et ils sont de type booléen.

```
% Variables de reconfiguration
array[1..4] of var bool:IsLink3Offset3Reconfig;
```

Nous modélisons les dates de disponibilité des informations transmises par les instances de trames RC sur un lien de données par les variables de la forme *Link<idLien>Disp<idTrame>*. L'exemple suivant illustre les dates de disponibilité des informations transmises par la f_{100} sur le premier lien d'id 11.

```
% Variables de disponibilites RC
array[1..2] of var 0..120:Link11Disp100;
```

Pour modéliser les dates de régulation du trafic, nous définissons les variables de la forme *Link<idLien>First<idTrame>*. L'exemple suivant illustre les dates des régulation pour les différentes transmissions de la trame f_{100} pour un envoi sur le premier lien du lien virtuel associé d'id 11.

```
% Variables de regualtion RC
array[1..2] of var 0..120:Link11First100;
```

Nous modélisons aussi les dates de transmission des différentes instances des trames sur chaque lien du lien virtuel associé par les variables de la forme *Link<idLien>Start<idTrame>*. L'exemple suivant illustre la déclaration des variables de début de transmission de la trame f_{100} sur le lien d'id 11.

```
% Variables de debut de transmission RC
array[1..2] of var 0..120:Link11Start100;
```

D'une manière similaire aux variables de début de transmission des trames RC sur chaque lien, nous déclarons les variables de fin de transmission des différentes instances sur chaque lien. La déclaration des variables associées aux fins de transmissions des trames sur chaque lien constitue une redondance dans notre modèle. En effet, nous pouvons obtenir directement ces dates en ajoutant les durées de transmission aux dates de début de transmission. Ce choix a été fait pour faciliter la propagation des contraintes et essentiellement celles de type *FIFO* qui reposent sur les dates d'arrivée des trames au niveau de chaque commutateur réseau. L'exemple suivant illustre la déclaration des variables de fin de transmission de la trame f_{100} sur le lien d'id 11.

```
% Variables de fin de transmission RC
array[1..2] of var 0..180:Link11End100;
```

Nous déclarons aussi des variables qui permettent de déterminer les durées de transmission des différentes trames sur le réseau. Pour ce faire, nous définissons un tableau de variables de la forme *Vl<idTrame>ToDestLink<idLink>Vect*. Chaque tableau détermine pour les différentes transmissions de la trame la durée de transmission dans le chemin de données ayant comme lien destination le lien d'id *idTrame*. Nous déclarons aussi une autre variable de type entière pour déterminer parmi les différentes transmissions sur ce chemin de données celle qui a la plus longue durée. L'exemple suivant montre la déclaration de ces variables pour le chemin de la trame f_3 vers le lien destination d'id 2.

```
% Variables des durees de transmission
array[1..4]  of var int:Vl3ToDestLink2Vect;
var int:Vl3ToDestLink2;
```

La déclaration de ce type de variables permet de propager plus aisément les contraintes de latence. Nous montrons dans la section suivante l'utilisation de l'ensemble des variables qu'on a déclaré pour la définition des contraintes du problème d'intégration des flux.

B.2 Déclaration des contraintes

Le premier type de contraintes fixe les dates de disponibilités des informations transmises par les différentes instances des trames. Pour la déclaration de ce type de contraintes, nous fixons les intervalles de disponibilités de l'information. L'exemple suivant configure les dates de disponibilité de l'information transmise par les instances de la trame f_{100}.

```
% Contraintes de disponibilite RC
constraint
(Link11Disp100[1]>=22);
constraint
(Link11Disp100[2]>=82);
constraint
(Link11Disp100[1]<=31);
constraint
(Link11Disp100[2]<=91);
```

Afin de lier les évènements de disponibilité de l'information aux évènements de régulation de trafic (Contraintes de régulation de trafic), nous déclarons les contraintes suivantes. Ce type de contraintes prend deux formes. Pour les premières instances de trames les dates de régulation de trafic coïncident exactement avec les dates de disponibilités de l'information. Pour la deuxième forme, nous utilisons la contrainte globale *maximum*. Cette contrainte est qualifiée comme globale vu qu'elle est une contrainte prédéfinie et dont la résolution est bien

optimisée. Elle permet de prendre le maximum entre la date de disponibilité de l'information et la date de régulation de la l'instance précédente plus la BAG associé. Pour la trame f_{100}, cette contrainte peut être déclarée par ces deux contraintes. La première correspond avec la première forme. La deuxième correspond à la deuxième forme et détermine la date de régulation pour la deuxième instance de la trame.

```
% Contraintes de regulation de trafic RC
constraint
(Link11Disp100[1]==Link11First100[1]);
constraint
maximum(Link11First100[2],[Link11First100[1]+5]++[Link11Disp100[2]]);
```

Les débuts de transmission des trames RC occurrent après la régulation de trafic. Afin de valider cet ordre d'évènement (c-à-d. régulation de trafic puis transmission) nous modélisons des contraintes liant les variables de régulation aux variables des débuts de transmission. Prenant l'exemple de la trame f_{100}, cette contrainte peut être codée de la façon suivante.

```
% Contraintes de debut de transmission
constraint  forall(i in 1..2)
(Link11First100[i]<= Link11Start100[i]);
```

Nous définissons également un ensemble de contraintes pour lier les variables de début de transmission aux variables de fin de transmission des trames de RC. Pour la trame f_{100}, la liaison entre le début et la fin de transmission sur le lien d'id 11 peut être traduite comme suit.

```
% Contrantes de delai de transmission
constraint forall( i in 1..2)
(Link11End100[i]-Link11Start100[i]==11);
```

Nous exprimons la dépendance des chemins RC en liant les variables des fins de transmission aux variables de début de transmission sur les liens suivants. Considérant un délai de traitement d'une trame par le commutateur réseau égal à une unité de temps, la dépendance de chemin de f_{100} sur les liens consécutifs d'id 11 et 2 peuvent s'exprimer comme suit

```
% Contraintes de chemin RC
constraint forall (i in 1..2)
(Link11End100[i] + 1 <= Link2Start100[i]);
```

Pour la dépendance des chemins TT, nous lions plutôt les offsets des différentes instances des trames sur deux liens consécutifs sur le réseau. Prenant l'exemple de la trame f_1 sur les deux liens consécutifs, la contrainte de dépendance de chemin peut s'écrire comme suit.

```
% Contraintes de chemin TT
constraint forall (i in 1..4)
(Link3Offset1[i] + 1 + 1 <= Link2Offset1[i]);
```

D'une manière similaire, nous exprimons la contrainte de limite de mémoire pour la transmission TT en bornant le délai entre deux instances de trames sur deux liens consécutifs. Pour l'exemple de la trame f_1, la contrainte de dépendance de chemin associée peut s'écrire comme suit.

```
% Contraintes de limite de memoire
constraint forall (i in 1..4)
((Link2Offset1[i] - Link3Offset1[i]) <= 30);
```

Les contraintes *FIFO* expriment l'ordre entre les dates de régulation de deux instances de différentes trames et les dates de transmission de ces deux instances sur le premier lien. Ils expriment aussi l'ordre des fins de transmission de deux instances de trames différentes et les débuts de transmission de ces instances de trames sur le prochain lien. Prenant l'exemple des trames f_{100} et f_{101} et la transmission sur le premier lien d'id 11 et le prochain lien d'id 2, les deux formes de la contrainte *FIFO* peuvent s'écrire de la façon suivante.

```
% Constraintes FIFO
constraint forall(i in 1..2 ,j in 1..2)
(((Link11First100[i] <= Link11First101[j]))<->
(Link11Start109[i] <= Link11Start110[j] ));
constraint forall(i in 1..2 ,j in 1..2)
(((Link11End100[i] <= Link11End101[j]))<->
((Link2Start100[i] < Link2Start101[j] ));
```

Nous notons ici qu'on exprime ces deux formes de contraintes par des équivalences entre les ordres des évènements pour permettre la propagation des contraintes dans les deux sens de l'équivalence.

Afin d'exprimer la contrainte d'envoi le plus tôt possible, nous vérifions que nous ne pouvons pas avancer la transmission de la trame même d'une unité de temps. Nous considérons pour cela les deux cas ou l'envoi est sur le premier lien du lien virtuel ou l'autre cas du relai par un commutateur réseau. Prenant le cas de la transmission de la trame f_{100} sur son premier lien d'id 11. Si une instance de cette trame n'est pas transmise juste au moment de la régulation, ceci signifie que le lien d'id 11 est occupé par la transmission de la trame f_{101} ou la trame f_{102}. Ceci peut être codé de la façon suivante.

```
% Contrainte Envoi le plus tot possible (Premier lien)
```

```
constraint forall (i in 1..2)
(( (Link11First100[i]) < (Link11Start100[i]) ) ->(
(exists(j in 1..2)
(((Link11Start101[j])<= (Link11Start101[i] - 1)) /\
((Link11Start100[i] - 1)<= (Link11Start101[j] +5))))\/
(exists(j in 1..2)
(((Link11Start102[j])<= (Link11Start100[i] - 1)) /\
((Link11Start100[i] - 1)<= (Link11Start102[j] +2))))
));
```

Nous prenons maintenant le cas d'un envoi le plus tôt possible de la même trame f_{100} sur son deuxième lien d'id 2. Si cette trame n'est pas relayée dès son arrivée au niveau du commutateur alors, le lien d'id 2 est occupé par la transmission de l'une des trames suivantes f_1, f_3, f_{10}, f_{13} , f_{101} ou f_{102}. Nous codons cette contrainte de la façon suivante.

```
% Contrainte Envoi le plus tot possible (Lien Intermediaire)
constraint forall (i in 1..2)
((( Link11End100[i]+1) < (Link2Start100[i] )) ->(
(exists(j in 1..4)
(((Link2Offset1[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<=( Link2Offset1[j] +1))))\/
(exists(j in 1..2)
(((Link2Offset3[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<=( Link2Offset3[j] +2))))\/
(exists(j in 1..2)
(((Link2Offset10[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<=( Link2Offset10[j] +3))))\/
(exists(j in 1..4)
(((Link2Offset13[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<=( Link2Offset13[j] +4))))\/
(exists(j in 1..2)
(((Link2Start101[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<= (Link2Start101[j] +5))))\/
(exists(j in 1..2)
(((Link2Start102[j])<= (Link2Start100[i] - 1) )/\
((Link2Start100[i] - 1)<= (Link2Start102[j] +2))))
));
```

Nous exprimons également un ensemble de contraintes pour exiger l'ordre des instances d'une seule trame sur un lien. À titre d'exemple, la première instance de la trame f_{100} sur le lien 11 est avant la deuxième instance de la même trame sur le même lien. Ceci peut se traduire comme suit en code *MiniZinc*.

```
% Contraintes d'anti-symetrie
constraint forall (i in 1..1)
( Link11Start100[i] <Link11Start100[i+1] );
```

Pour assurer le non-chevauchement des transmissions des trames on utilise la contrainte globale appelée *disjunctive*. Cette contrainte prend comme premier paramètre les dates de début de transmission des différentes instances des trames sur le lien considéré et comme deuxième paramètre leurs différentes durées de transmissions dans l'ordre. Moyennant ces deux paramètres, elle assure la disjonction des fenêtres de temps associées aux transmissions de ces instances sur le lien considéré. Pour le lien d'id 2, les trames transmises sur ce lien sont f_1, f_3, f_{10}, f_{13}, f_{100}, f_{101} et f_{102} la contrainte globale de non-chevauchement peut être ainsi codée comme suit.

```
% Contraintes de non-chevauchement
constraint
disjunctive(
[ Link2Offset1[i] | i in 1..4 ]++[ Link2Offset3[i] | i in 1..2 ]++
[ Link2Offset10[i] | i in 1..2 ]++[ Link2Offset13[i] | i in 1..4 ]++
[ Link2Start100[i] | i in 1..2 ]++[ Link2Start101[i] | i in 1..2 ]++
[ Link2Start102[i] | i in 1..2 ],
[ 1, 1, 1, 1]++[ 2, 2]++[ 3, 3]++[ 4, 4, 4, 4]++
[ 11, 11]++[ 5, 5]++[ 2, 2]);
```

Nous définissons le lien entre les variables de reconfiguration et les différents offsets des trames TT par les contraintes de reconfiguration. Pour la première instance de la trame f_1 sur lien d'id 3, en connaissant que son ancien offset est égal à 17, la contrainte de reconfiguration peut s'écrire comme suit.

```
% Contrainte de reconfiguration
constraint
((Link3Offset1[1]=17) <-> (IsLink3Offset1Reconfig[1]=0));
```

Pour exprimer la contrainte de latence d'une trame sur un de ces chemins, nous définissons trois contraintes. La première lie les variables des durées de transmission aux différentes dates de transmission des trames. La deuxième détermine entre les différentes transmissions de cette trame sur le chemin la durée la plus longue. La troisième borne ce délai de transmission. Pour la trame f_1 en fixant un délai maximal de 100 unités de temps sur son chemin vers le lien destination d'id 2, ces contraintes peuvent être codées dans l'ordre comme suit.

```
% Contraintes de latence TT RC
constraint forall (i in 1..4)
```

```
(( Link2Offset1[i]+1-Link3Offset1[i])=Vl1ToDestLink2Vect[i]);
constraint
(maximum( Vl1ToDestLink2,[Vl1ToDestLink2Vect[i]|i in 1..4]));
constraint
( Vl1ToDestLink2 <= 100);
```

Nous montrons dans la section suivante la déclaration de la fonction objectif.

B.3 Déclaration de la fonction objectif

Pour définir la fonction objectif, nous déclarons une variable entière *obj*. La valeur de cette variable est définie par une contrainte qui prend en compte les valeurs des variables de reconfiguration et les couts de reconfiguration. Le code suivant fournit la déclaration de cette variable et une partie de la définition de cette variable en considérant seulement les deux premières instances de la trame f_1 sur le lien d'id 3.

```
% Fonction objectif
var int:obj;
constraint
(obj=IsLink3Offset1Reconfig[1]*1
+IsLink3Offset1Reconfig[2]*1
..
);
```

Nous définissons dans la prochaine section la directive de résolution relative au problème d'intégration itérative des deux types de flux.

B.4 Directive de résolution

La directive de résolution permet de définir le type du problème, la fonction objectif à optimiser et éventuellement l'heuristique de branchement à considérer. Le code suivant illustre dans la deuxième ligne qu'il s'agit de construire un arbre de recherche booléen. Ceci coïncide parfaitement avec notre contexte vu que les variables de branchement sont les variables booléennes de reconfiguration. Dans la troisième ligne du code, nous déclarons nos variables de reconfiguration en suivant l'ordre des liens dans les liens virtuels. Ce code indique au solveur de résoudre la reconfiguration pour la trame f_1 sur le lien d'id 3 avant le lien d'id 2. L'indication explicite de l'ordre de ces variables est donnée par la mention de *input_order* comme paramètre du choix des variables. Nous indiquons aussi pour le solveur qu'il faut commencer par les cas de non-reconfiguration (valeurs booléennes égales à 0). Ceci revient à commencer

par les plus petites valeurs d'où la mention *indomain_min* au solveur. La dernière ligne de ce code illustre la nature du problème (un problème de minimisation) via la directive *minimize* et la fonction objectif à optimiser. Dans notre cas, il s'agit de la variable *obj*. Nous notons aussi que pour la phase 2 de notre problème nous choisissons la directive *satisfy* pour indiquer qu'il s'agit d'un problème de satisfaction.

```
% Directive de resolution
solve
::bool_search(
[IsLink3Offset1Reconfig[i]|i in 1..4]++
[IsLink2Offset1Reconfig[i]|i in 1..4]++
..
,input_order,indomain_min, complete )
minimize obj;
```