Projecte de Fi de Carrera

Control Actiu en AFDX
i
Modelització d'ARINC 653

Joan Adrià RUIZ DE AZÚA ORTEGA

Responsables

Ph.D **Guchuan Zhu**	-	École Polytechnique de Montréal
Ph.D **Fabrice Francès**	-	ISAE DISC
Ph.D **José Radzik**	-	ISAE DEOS
Ph.D **Oscar Esparza**	-	ETSETB - UPC

Table des matières

1 Prélude **4**

2 Contrôle actif de la bande passante et du jitter dans AFDX **6**
2.1 Introduction . . . 6
2.2 Formulation du problème . . . 8
2.2.1 Base du Network Calculus . . . 8
2.2.2 Modelage du système . . . 10
2.3 Bandwidth Efficiency Control Loop . . . 11
2.3.1 Limites de la stabilité . . . 11
2.3.2 Stratégie de contrôle . . . 14
2.4 Jitter Stabilization Control Loop . . . 15
2.4.1 Limites de la stabilité . . . 15
2.4.2 Stratégie de contrôle . . . 19
2.4.3 Prédiction du jitter $\widehat{\Delta t}$ . . . 21
2.5 Contrôle de la composition . . . 22
2.6 Étude de cas . . . 23
2.7 Conclusion . . . 24
2.8 Références . . . 24

3 Modélisation de la planification de partitions pour différents niveaux d'états critiques **28**
3.1 Introduction . . . 28
3.2 Partage temporel pour des systèmes avec plusieurs niveaux critiques . . . 30
3.2.1 Déclaration du problème . . . 30
3.2.2 Caractérisation des temps d'exécution et de commutation d'une instance 33
3.2.3 Caractérisation du temps d'exécution de la tâche avec préemption . . 35
3.2.4 Résummé des variables du problème . . . 38
3.3 Formulation des contraintes du problème . . . 38
3.3.1 Distribution des tâches entre partitions . . . 40
3.3.2 Contraintes des partitions . . . 41
3.3.3 Contraintes d'ordonnancement . . . 44
3.4 Formulation des objectifs pour l'optimisation . . . 46
3.4.1 Minimisation du temps alloué aux partitions . . . 46
3.4.2 Minimisation du temps de réponse . . . 48
3.4.3 Fonctions objectives quand $\mu = 0$ . . . 49
3.4.4 Complexité du problème . . . 50
3.5 Étude de cas . . . 50
3.6 Conclusion . . . 53
3.7 Références . . . 53

4 Reconnaissances **56**

Table des figures

2.1 Une configuration avec $T_{app} = 96$ ms et $BAG = 64$ ms. 7
2.2 Représentation modulaire d'un End-System avec 3 VLs 11
2.3 Représentation du modelage de la BECL pour le VL_j 12
2.4 Représentation du modèle JSCL pour le VL de référence VL_m et $\forall i \in \mathcal{I}$, VL_i 15
2.5 Représentation du nouveau processus de point limite 20
2.6 Longueur de la frame et jitter après le SM : bleu-croix pour le VL_1 ; rouge-étoile pour le VL_2 ; vert-étoile pour le VL_3 ; bleu ciel-carré pour le VL_4 ; noire-carré pour les frames du système de contrôle. 25

3.1 Définition d'une tâche 31
3.2 Définition de la k-ème instance de la partition P_j 32
3.3 Problème d'intégration 33
3.4 Résultats de l'optimisation 52

Liste des tableaux

2.1 Paramètres des VLs 23
2.2 Les valeurs théoriques 23

3.1 Variables du problème 39
3.2 Paramètres des tâches 51

Chapitre 1

Prélude

Ce rapport vise à exprimer les détails sur le travail et l'expérience réalisés pendant le stage de recherche à l'**École Polytechnique de Montréal** dans le département Génie Électrique. La recherche a été focalisée sur le domaine des systèmes embarqués. Concrètement, elle a été divisée sur deux sujets basés sur deux standards : ARINC 664 et ARINC 653.

Le premier sujet basé sur le standard ARINC 664 Part 7, connu comme Avionic Full Duplex Switched Ethernet (AFDX), a été focalisé sur l'amélioration de l'utilisation des ressources, comme la bande passante, avec l'application d'un système actif de contrôle.

AFDX est une technologie qui permet de garantir des performances prévisibles, ce qui est convenable pour les applications avioniques en temps-réel. Pour renforcer la qualité du service, le standard définit le concept de Virtual Link (VL) qui détermine une connexion du type point-à-point logique et unidirectionnelle. La transmission du VL est contrainte par le Bandwidth Allocation Gap (BAG), c'est-à-dire, deux frames consécutives doivent respecter au moins un période BAG entre ses transmissions. En considérant que quelques valeurs du BAG sont autorisées au même temps que la période de l'application est arbitraire, l'utilisation de la bande passante peut être petite. En plus, comme AFDX est un protocole asynchrone, le jitter dégrade le déterminisme du réseau (limitation pour les applications critiques).

Pour cette raison, la recherche réalisée a été focalisée sur l'utilisation d'un système de *feedback* de contrôle actif pour éviter ces problèmes. Concrètement, deux schémas de contrôle ont été définis. Le premier, appelé Bandwidth Efficiency Control Loop (BECL), vise à maximiser l'utilisation de la bande passante sur un VL. Le deuxième, appelé Jitter Stabilization Control Loop (JSCL), est utilisé pour stabiliser le jitter sur une valeur prédéfinie.

L'explication des résultats sur cette recherche est détaillée dans le Chapitre 2.

Le deuxième sujet, ce qui se base sur le standard ARINC 653, permet d'apporter un modèle d'optimisation pour l'allocation des partitions ainsi que l'ordonnancement des processus (appelés *tâches*) qui partagent les même ressources dans un seul système.

L'utilisation de l'architecture Intégrée Modulaire Avionique (IMA) est une tendance technologique dans l'industrie avionique, grâce à sa capacité de partager le temps et l'espace entre les différents processus. Néanmoins, combiner l'allocation et l'ordonnancement des applications qui partagent des ressources dans la même plateforme devient un problème d'une énorme complexité. Pour adresser ce défi, un modèle qui permet un ajustement automatique selon un critère d'optimisation a été développé.

Dans le modèle présenté toutes les variables sont entières et toutes les contraintes sont formulées à travers d'inégalités et égalités linéaires. Pour cette raison, ce problème peut être résolu d'une façon efficiente en utilisant différent algorithmes des programmation linéaire. Le Chapitre 3 détaille les contraintes et les objectifs d'optimisation utilisés pour la modélisation

du système.

Cette expérience m'a permis d'approfondir sur le domaine des systèmes embarqués, surtout le réseau AFDX. De plus, j'ai pu appliquer des connaissances obtenues pendant mes études sur des situations réelles. Les résultats de cette expérience ont été la production de deux articles :

— l'article *Active Bandwidth Efficiency and Jitter Control for AFDX : A Network Calculus Approach* a été soumis comme un draft sur la conférence *IEEE INFOCOM 2016*.
— l'article *A Modeling and Optimization of Simultaneous Partition Allocation and Schedule Design for Mixed-Criticality Systems* a été soumis comme un draft sur le journal *European Journal of Operational Research*.

Chapitre 2

Contrôle actif de la bande passante et du jitter dans AFDX

2.1 Introduction

Avionics Full-Duplex Switched Ethernet (AFDX) est une technologie émergente développée par l'industrie avionique pour supporter les exigences des systèmes. Concrètement, l'incrément de la demande sur la bande passante et la fiabilité des systèmes de communications avioniques sont les défis que cette technologie, standardisée dans *ARINC 664 Part 7* [1], essaie de réussir. Due à ses prestations, elle est déployée sur un grand nombre des avions commerciaux actuels, comme par exemple Airbus A380, A350 et Boeing B787.

Le réseau AFDX est typiquement composé de trois éléments : End Systems (ESs), Switchers (SWs) et de liaisons physiques. L'ES produit ou consomme des services pour transférer ou recevoir des données générées par les systèmes avioniques. Chaque ES est connecté à un SW à travers une liaison physique redondante, c'est-à-dire, d'une duplication de la connexion pour augmenter la robustesse de la transmission. De plus, pour assurer une transmission sans collisions, et donc en garantissant un accès au médium sans pertes, une connexion du type full duplex est adoptée par le standard.

Le réseau AFDX est installé sur la technologie *Ethernet*, qui est caractérisée par une communication aléatoire qui n'offre aucune garantie de service pour les applications avioniques. Pour cette raison, pour atteindre une communication déterministe qui permet d'accorder des garanties de performance prévisibles, AFDX introduit le concept de *Virtual Link* (VL). Comme le standard [1] indique, un VL est un chemin logique unidirectionnel qui connecte l'ES source avec un ou plusieurs ESs finaux.

Pour garantir un service fiable, une bande passante maximale est allouée à chaque connexion. Ce limite est respecté grâce à que le trafic est modelé à l'entrée du réseau (end-systems) en suivant le *Bandwidth Allocation Gap* (BAG) T_{BAG}. Ce paramètre définit l'intervalle minimal de temps entre deux frames successifs d'un VL. En plus, le *Maximum Frame Length* (MFL) $l^{\max}$ détermine la taille maximale des frames qui peuvent être transmises dans le VL. D'autres mécanismes sont appliqués dans les SWs pour offrir une garantie de la performance, comme par exemple la régulation et le filtrage des frames.

Dans la littérature actuelle sur AFDX, la plupart du travail est dédié à renforcer la performance du réseau, notamment à la réduction et/ou contraction du retard et l'estimation

du jitter [2, 3, 4, 5, 6, 7, 8, 9, 10]. Le travail présenté dans ce chapitre adresse le problème de performance sur une autre perspective, soit l'amélioration de l'efficience de la bande passante et la stabilisation du jitter. Ce dernier objectif a une grande importance due au fait qu'il a un gros impact sur le déterminisme du réseau AFDX.

Spécifiquement, il doit être remarqué que dans la pratique l'utilisation de la bande passante devient petite en comparaison avec celle qui est préservée par l'utilisation du BAG et MFL. Une des raisons est le fait que le BAG doit être défini à partir d'un petit ensemble de valeurs ($\{2^n\}_{n=0}^{7}$ ms), tandis que la période du flux de données de l'application est spécifique et donc elle peut être arbitraire. On considère l'exemple illustré dans la Fig. 2.1 avec un BAG de 64 ms pour transmettre l'information d'une application avec la période de 96 ms. On note que un tiers de la bande passante allouée est gaspillée.

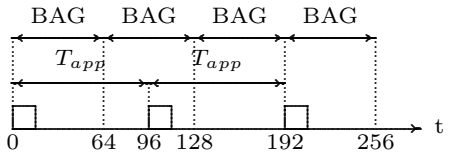


FIGURE 2.1: Une configuration avec $T_{app} = 96$ ms et $BAG = 64$ ms.

Un aspect plus important à considérer pour ce type de configuration est le fait que le déterminisme de la communication devient dégradé. Ce phénomène est dû à que AFDX est un protocole asynchrone et le standard ne force pas au VL la transmission des frames s'il n'y a pas de données à transmettre, même si le VL est disponible. Pour cette raison, l'ES final est incapable de déterminer si un créneau BAG vide est dû à la perte de la frame ou à la non-transmission de l'ES source.

Pour adapter la transmission afin d'éviter cette dégradation, l'article [11] propose l'insertion des frames qui remplissent tous les créneaux BAG vides. Néanmoins, l'incrément sur le nombre de frames dans les fluxes des VLs peut introduire des jitters supplémentaires due au multiplexage. De plus, une insertion statique des frames introduit des entêtes supplémentaires, ce qui réduit la performance.

Pour répondre à cette problématique, le travail présenté dans ce chapitre utilise l'approche du *contrôle actif* dans le cadre du *Network Calculus* (NC) [12]. Ce concept se base sur un système de bouclage qui est capable d'analyser le comportement d'un élément pour prendre une décision. Cette décision est réalisée à éviter ou renforcer un état de l'élément analysé. Pour cette raison, une loi de contrôle, nommée *feedback*, est produite en concordance avec la variation de l'état de l'élément analysé.

Dans notre cas d'étude, cette technique permet d'augmenter ou de limiter le flux d'entrée du VL en fonction de l'utilisation de la bande passante et du jitter. Pour cette raison, grâce à ce contrôle actif sur le flux d'entrée, l'ES est capable d'optimiser les ressources du VL au même temps qu'il stabilise le jitter.

Différent types de systèmes de contrôle basés sur le NC ont été développés récemment. Il est proposé dans [13, 14, 15] un système capable de dynamiquement analyser le retard de la communication dans un réseau TCP pour contrôler le flux d'entrée, et donc obtenir l'utilisation maximale de la bande passante sans congestionner le réseau. Un algorithme basé sur

NC pour contrôler l'admission sur un réseau sans fils maillé est développé par [16, 17], qui permet d'améliorer la performance du système en comparaison avec le traditionnel TDMA. Une approche similaire est proposée par [18] pour le contrôle de flux d'un réseau de capteurs distribué (*Distributed Sensor Network*). Ce type de réseau est intéressant parce que son modèle de queue est similaire au concept de VL dans le réseau AFDX.

La recherche réalisée introduit deux boucles de contrôle : *Bandwidth Efficiency Control Loop* (BECL) et *Jitter Stabilization Control Loop* (JSCL). La BECL vise à maximiser l'utilisation de la bande passante du VL, pendant que la JSCL vise à stabiliser le jitter des VLs, donc le déterminisme du réseau est amélioré. Comme dans les prochaines sections détaillent, ces deux boucles de contrôle peuvent être utilisées simultanément ou séparément dans un VL. Cependant, la JSCL est limité à être applicable sur un unique VL, celui qui dispose de la valeur maximale du jitter la plus petite possible. Les limites du *feedback* pour garantir la stabilité de chaque boucle fermée sont définis à partir de la théorie de Network Calculus. De cette façon, une simple (mais effective) lois de contrôle est développée.

Ce chapitre est organisé de la façon suivante. La Section 2.2 présente les notations du Network Calculus et la formulation utilisée pour définir les boucles de contrôle. Les Sections 2.3 et 2.4 sont dédiées au développement de la BECL et la JSCL, respectivement. La composition de la boucle qui inclut les deux boucles de contrôle est débattue dans la Section 2.5. Un étude de cas est présenté dans la Section 3.5, suivit par des remarques concluants dans la Section 3.6.

2.2 Formulation du problème

2.2.1 Base du Network Calculus

Cette section introduit quelques notations basiques et propriétés de Network Calculus (NC) requises pour modéliser le système. Pour une présentation plus détaillée de NC, référez-vous à [19].

On considère $\mathbb{R}^+$ comme l'ensemble de nombres réels non-négatifs et $\bar{\mathbb{R}}^+ \equiv \mathbb{R}^+ \cup \{+\infty\}$. Deux ensembles de fonctions croissantes, nommées *processus*, sont définies de la façon suivante :

$$\Gamma = \{M : \mathbb{R}^+ \mapsto \bar{\mathbb{R}}^+ : t \mapsto M(t)\},$$

$$\tilde{\Gamma} = \{\tilde{N} : \mathbb{R}^+ \times \mathbb{R}^+ \mapsto \bar{\mathbb{R}}^+ : (s,t) \mapsto \tilde{N}(s,t)\}.$$

On note que $\tilde{\Gamma}$ est une généralisation de Γ. Concrètement, pour chaque $M \in \Gamma$, il existe un $\tilde{M} \in \tilde{\Gamma}$ tel que

$$\forall s,t \geq 0, \quad \tilde{M}(s,t) = \begin{cases} M(t-s), & \text{si } s \in [0,t]; \\ \infty, & \text{autrement.} \end{cases}$$

De plus, Γ_0 dénote le sous-ensemble de Γ tel que $M(0) = 0$. Dans le même sens, $\tilde{\Gamma}_0$ dénote le sous-ensemble de $\tilde{\Gamma}$ tel que $\tilde{M}(t,t) = 0$.

Définition 2.1 (***Min-plus convolution***). *On considère $M \in \Gamma$ et $\tilde{N} \in \tilde{\Gamma}$. Les min-plus convolutions entre M et $\tilde{N}$ sont les processus spécifiés $\forall s, t \geq 0$*

$$(M \otimes M)(t) = \inf_{s \in [0,t]} \{M(t-s) + M(s)\}, \tag{2.1a}$$

$$(M \otimes \tilde{N})(t) = \inf_{s \in [0,t]} \{M(s) + \tilde{N}(s,t)\}, \tag{2.1b}$$

$$(\tilde{N} \otimes \tilde{N})(s,t) = \inf_{\tau \in [s,t]} \{\tilde{N}(s,\tau) + \tilde{N}(\tau,t)\}. \tag{2.1c}$$

Définition 2.2 (***Closure sous-additive***). *On considère $M \in \Gamma$ (ou $\in \tilde{\Gamma}$) et dénote $M^{(i)}$ comme le processus obtenu en répétant la convolution avec lui-même $(i-1)$ fois :*

$$M^{(i)} = \underbrace{M \otimes M \otimes M \otimes \cdots \otimes M}_{i-1 \ fois}$$

Ensuite, la closure sous-additive de M est définie par

$$M^* = \inf_{n \geq 0} \{M^{(n)}\}$$

Définition 2.3 (***Réponse impulsionnelle***). *On considère M, $N \in \Gamma$ comme, respectivement, les processus d'entrée et de sortie d'un système linéaire (causal). Ensuite, le processus $\tilde{H} \in \Gamma$ est nommée comme la réponse impulsionnelle du système s'il respecte*

$$N = M \otimes \tilde{H}.$$

Définition 2.4 (***Sous-additivité***). *Une fonction $\tilde{N} \in \tilde{\Gamma}$ est mentionnée sous-additive si*

$$\forall s, t \geq 0, \forall \tau \in [s,t], \quad \tilde{N}(s,t) \leq \tilde{N}(s,\tau) + \tilde{N}(\tau,t).$$

Théorème 2.1 (***Closure et sous-additivité*** **[19]**). *Si $\tilde{N} \in \tilde{\Gamma}$ est sous-additive*

$$\tilde{N}^* = \tilde{N}.$$

Démonstration. En considérant la propriété sous-additive

$$\forall s, t \geq 0, \forall \tau \in [s,t], \quad \tilde{N}(s,t) \leq \tilde{N}(s,\tau) + \tilde{N}(\tau,t),$$

la relation suivante est établie $\forall s, t \geq 0$

$$(\tilde{N} \otimes \tilde{N})(s,t) = \inf_{\tau \in [s,t]} \{\tilde{N}(s,\tau) + \tilde{N}(\tau,t)\} = \tilde{N}(s,t).$$

Alors, la closure du processus devient le même processus $\tilde{N}^* = \tilde{N}$. □

Définition 2.5 (***fonction $\tilde{\Lambda}$-contraint***). *La fonction $M \in \Gamma$ est nommée $\tilde{\Lambda}$-contraint si*

$$\forall t \geq 0, \quad M(t) \leq (M \otimes \tilde{\Lambda})(t).$$

Théorème 2.2 (***Backlog bound*** **[18]**). *On considère $M \in \Gamma$ comme une entrée $\tilde{\Lambda}$-contraint de l'élément du réseau $\tilde{S} \in \tilde{\Gamma}$ et $N = M \otimes \tilde{S}$ comme la sortie de cet élément. Dans ce cas, le backlog, défini comme $M - N$, est limité par le processus $Q \in \Gamma$ si*

$$\forall s, t \geq 0, \quad \tilde{\Lambda}(s,t) \leq \tilde{S}(s,t) + Q(t).$$

Démonstration. Si le processus M est $\tilde{\Lambda}$-contraint, la relation suivante est établie (Définition 2.5) :

$$\forall t \geq 0, \quad M(t) \leq (M \otimes \tilde{\Lambda})(t).$$

De plus, si on considère la relation $\tilde{\Lambda}(s,t) \leq \tilde{S}(s,t) + Q(t)$, c'est possible de montrer que $\forall t \geq 0$

$$M(t) \leq \inf_{s\in[0,t]}\{M(s) + \tilde{\Lambda}(s,t)\} \leq \inf_{s\in[0,t]}\{M(s) + \tilde{S}(s,t) + Q(t)\}$$

En suivant la définition de réponse impulsionnelle (Définition 2.3)

$$M(t) \leq \inf_{s\in[0,t]}\{M(s) + \tilde{S}(s,t)\} + Q(t) = N(t) + Q(t)$$

Cela démontre que le backlog est limité par le processus Q. □

Théorème 2.3 (***propriété de*** $\tilde{\Lambda}^*$***-contraint*** **[18]**)**.** *Si le processus $M \in \Gamma$ est $\tilde{\Lambda}^*$-contraint, il est $\tilde{\Lambda}$-contraint aussi.*

Démonstration. En considérant que le processus M est $\tilde{\Lambda}^*$-contraint (Définition 2.5)

$$\forall t \geq 0, \quad M(t) \leq (M \otimes \tilde{\Lambda}^*)(t) = \inf_{s\in[0,t]}\{M(s) + \tilde{\Lambda}^*(s,t)\},$$

et que $\tilde{\Lambda}^* \leq \tilde{\Lambda}$, donc

$$\forall t \geq 0, \quad M(t) \leq \inf_{s\in[0,t]}\{M(s) + \tilde{\Lambda}^*(s,t)\} \leq \inf_{s\in[0,t]}\{M(s) + \tilde{\Lambda}(s,t)\}$$

Cela montre que le processus M devient $\tilde{\Lambda}$-contraint aussi. □

Théorème 2.4 (***Restriction du processus de départ*** **[18]**)**.** *Le processus de sortie d'un élément linéaire avec une réponse impulsionnelle $\tilde{\Lambda}^* \in \tilde{\Gamma}_0$ est $\tilde{\Lambda}$-contraint*

Démonstration. Les processus $M, N \in \Gamma$ dénotent l'entrée et la sortie de l'élément du réseau avec une réponse impulsionnelle $\tilde{\Lambda}^* \in \tilde{\Gamma}$. Ensuite, si le processus de sortie est $\tilde{\Lambda}^*$-contraint cela implique que

$$(N \otimes \tilde{\Lambda}^*)(t) = (M \otimes \tilde{\Lambda}^* \otimes \tilde{\Lambda}^*)(t) = (M \otimes \tilde{\Lambda}^*)(t) = N(t)$$

Cela montre que le processus de sortie N devient $\tilde{\Lambda}^*$-contraint. En appliquant le théorème 2.3 c'est possible de montrer que le processus N est aussi $\tilde{\Lambda}$-contraint. □

2.2.2 Modelage du système

Comme la Fig. 2.2 montre, un ES du réseau AFDX est composé d' un *Module Régulateur* (RM) et un *Module Planificateur* (SM). Le RM est utilisé pour adapter le flux d'entrée de chaque VL, caractérisé par une période T_{app}, pour qu'il respecte la spécification du BAG. Le SM est le composant qui multiplexe les frames de tous les VL, qui seront transmises à travers de la liaison physique. Les boucles de contrôle, détaillés dans les prochaines sections, sont appliquées par VL.

Les notations suivantes sont utilisées tout au long de ce chapitre pour spécifier les variables et les paramètres du modelage du système de contrôle :

— $\mathcal{N} = \{1, 2, \cdots, n\}$ est l'ensemble d'indices qui représente tous les VLs de l'ES, avec n comme le nombre total de VLs.

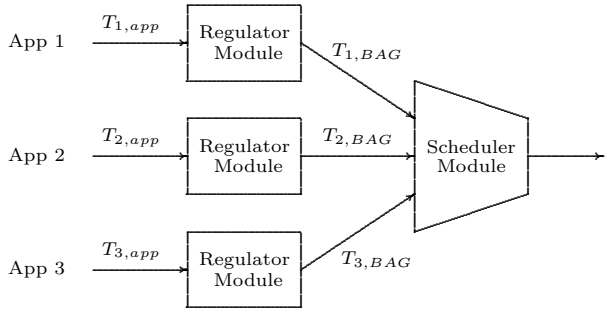


FIGURE 2.2: Représentation modulaire d'un End-System avec 3 VLs

— $\mathcal{I}$ est l'ensemble d'indices qui représente les VLs de l'ES en excluant celui sur lequel la JSCL est appliquée.
— $m \in \mathcal{N} \backslash \mathcal{I}$ est l'indice qui représente le seul VL sur lequel la JSCL est appliquée.
— L'indice k représente le temps discret avec une période $T_{j,BAG}$, $j \in \mathcal{N}$. Il caractérise l'instant $t = kT_{j,BAG}$ pour le VL_j.
— Le maximum et le minimum de la variable x sont dénotes par $x^{\max}$ et $x^{\min}$, respectivement.
— Le minimum entre les processus X et Y est dénoté par $\min\{X, Y\} = X \wedge Y$.
— La fonction projection $\max\{\cdot, 0\}$ est dénoté comme $(\cdot)^+$.
— Un processus X impliqué dans la BECL est dénoté par X^B.
— Un processus X impliqué dans la JSCL est dénoté par X^J.

2.3 Bandwidth Efficiency Control Loop

La BECL vise à améliorer l'efficience de la bande passante du VL sur lequel elle est appliquée. Ce système de contrôle actif peut être appliqué sur tous les VLs de l'ES ($\forall j \in \mathcal{N}$).

Comme la Fig. 2.3 montre, la BECL du VL_j est composée d'un buffer virtuel, qui est représenté par la réponse impulsionnelle $\tilde{S}^B_{B_j}$ et positionné après le RM. Ce buffer permet d'interpréter le flux régulé F_j pour détecter si le maximum de la bande passante du VL est atteint. Pour contrôler l'efficience de la bande passante du VL, le processus de la commande C^B_j est appliqué et le processus du feedback résultant R^B_j est propagé pour limiter le flux d'entrée du VL.

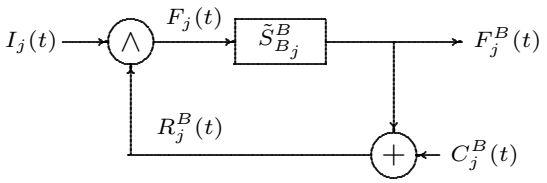


FIGURE 2.3: Représentation du modelage de la BECL pour le VL_j

2.3.1 Limites de la stabilité

Cette sous-section établit les limites du processus de feedback sous lesquelles le système en boucle fermée reste stable. Les relations entre les différents processus peuvent être dérivées à partir de la Fig. 2.3, qui sont exprimées par $\forall j \in \mathcal{N}$:

$$F_j = R_j^B \wedge I_j, \tag{2.2}$$

$$F_j^B = F_j \otimes \tilde{S}_B^B. \tag{2.3}$$

Ces fonctions sont caractérisées par la réponse impulsionnelle du buffer virtuel $\tilde{S}_{B_j}^B$ et de la commande $\tilde{H}_{C_j}^B$.

Lemme 2.5 (Lemme 1 dans [18])**.** *On considère $B \in \Gamma_0$ comme la fonction qui représente le niveau du buffer virtuel à travers du temps, c'est-à-dire, le processus de la réduction. Dénotés par M, $N \in \Gamma$ les processus d'entrée et de sortie, respectivement. Ensuite, $N = M \otimes \tilde{S}_B \in \Gamma_0$ avec*

$$\forall s, t \geq 0, \quad \tilde{S}_B(s,t) = B(t) - B(s). \tag{2.4}$$

Lemme 2.6. *On considère C comme le processus de la commande et M, $N \in \Gamma$ comme les processus d'entrée et de sortie. Ensuite, $N = M + C = M \otimes \tilde{H}_C$ avec*

$$\forall s, t \geq 0, \quad \tilde{H}(s,t) = \begin{cases} C(t), & \text{si } s = t; \\ \infty, & \text{autrement.} \end{cases}$$

En utilisant le Lemme 2.6, il est possible de représenter le processus de feedback R_j^B de la façon suivante :

$$\forall j \in \mathcal{N}, \quad R_j^B = F_j^B + C_j^B = F_j^B \otimes \tilde{H}_{C_j}^B. \tag{2.5}$$

Théorème 2.7. *On considère $\tilde{S}_{B_j}^B \in \tilde{\Gamma}_0$ comme la réponse impulsionnelle du buffer virtuel et $\tilde{H}_{C_j}^B \in \tilde{\Gamma}$ comme celle de la commande pour le VL_j. Ensuite, la réponse impulsionnelle de la boucle de feedback avec $R_j^B = F_j \otimes \tilde{S}_{L_j}^B$ est exprimée par*

$$\forall j \in \mathcal{N}, \quad \tilde{S}_{L_j}^B = \tilde{S}_{B_j}^B \otimes \tilde{H}_{C_j}^B. \tag{2.6}$$

Démonstration. En utilisant le Lemme 2.6, $\forall j \in \mathcal{N}$

$$R_j^B = F_j^B \otimes \tilde{H}_{C_j}^B = F_j \otimes \tilde{S}_{B_j}^B \otimes \tilde{H}_{C_j}^B = F_j \otimes \tilde{S}_{L_j}^B.$$

Ensuite par définition, la démonstration devient directe. □

A partir du Théorème 2.7, on peut obtenir :

$$\forall j \in \mathcal{N}, \quad F_j = R_j^B \wedge I_j = (F_j \otimes \tilde{S}_{L_j}^B) \wedge I_j(t). \tag{2.7}$$

Théorème 2.8. *Si*

$$\forall j \in \mathcal{N}, \forall t \geq 0, \quad C_j^B(t) > 0, \tag{2.8}$$

$F_j = I_j \otimes (\tilde{S}_{L_j}^B)^*$ est la solution unique de (2.7).

Démonstration. En se basant sur le Lemme 8 de [20], si

$$\forall j \in \mathcal{N},\, \forall t \geq 0, \forall s \in [0,t] \quad \inf_{s \leq u \leq v \leq t} \tilde{S}^B_{L_j}(u,v) > 0, \tag{2.9}$$

$I_j \otimes (\tilde{S}^B_{L_j})^*$ est la solution unique de (2.7). Comme cette condition est satisfaite $\forall t \geq 0$, $\forall s \in [0,t]$ et $\forall j \in \mathcal{N}$:

$$(\tilde{S}^B_{B_j} * \tilde{H}^B_{C_j})(s,t) = B^B_j(t) - B^B_j(s) + C^B_j(t) > 0,$$

on obtient donc $\forall j \in \mathcal{N}$, $\forall t \geq 0$ and $\forall s \in [0,t]$

$$C^B_j(t) > B^B_j(s) - B^B_j(t).$$

En considérant que le processus $B^B_j \in \Gamma_0$ est non-décroissant, le pire cas apparait quand $s = t$, c'est-à-dire

$$\forall j \in \mathcal{N}, \forall t \geq 0, \quad C^B_j(t) > B^B_j(t) - B^B_j(t) = 0.$$

□

Théorème 2.9. *Si*

$$\forall j \in \mathcal{N},\, \forall t \geq 0, \quad C^B_j(t) \geq 0, \tag{2.10}$$

$\tilde{S}^B_{L_j}$ *est sous-additive.*

Démonstration. Grâce à la Définition 2.4, il peut être montré que $\forall s,t \geq 0$, $\forall \tau \in [s,t]$ et $\forall j \in \mathcal{N}$:

$$\begin{aligned} &\tilde{S}^B_{L_j}(s,\tau) + \tilde{S}^B_{L_j}(\tau,t) - \tilde{S}^B_{L_j}(s,t) \geq 0, \\ &C^B_j(\tau) \geq 0. \end{aligned}$$

La démonstration devient en spécifiant $\tau = t$. □

Quand $\tilde{S}^B_{L_j}$ est sous-additive, la relation suivante reste (Théorème 2.1) :

$$\forall j \in \mathcal{N}, \quad \tilde{S}^B_{L_j} = (\tilde{S}^B_{L_j})^*.$$

Pour cette raison, la solution unique de (2.7) devient

$$\forall j \in \mathcal{N}, \quad F_j = I_j \otimes \tilde{S}^B_{L_j}. \tag{2.11}$$

Théorème 2.10. *On considère F^B_j comme le processus du flux régulé et k comme l'indice du temps discret pour le VL_j. Si*

$$\forall j \in \mathcal{N}, \quad R^B_j[k] > F^B_j[k-1], \tag{2.12}$$

Les Théorèmes 2.9 et 2.8 sont satisfaits.

Démonstration. En utilisant (2.5) dans le Théorème 2.8, la condition devient :

$$\forall j \in \mathcal{N}, \forall t \geq 0, \quad C^B_j(t) = R^B_j(t) - F^B_j(t) > 0.$$

Comme le feedback doit être réaliser à chaque période BAG $T_{j,BAG}$ du VL_j, cette condition doit être respectée dans le domaine discret. En plus, le processus du feedback à l'instant k dépend que de l'état de l'instant $k-1$. □

Théorème 2.11. *On considère F_j^B comme le processus du flux régulé, $l_j^{\max}$ comme la MFL et k comme l'indice du temps discret pour le VL_j. On suppose que les Théorèmes 2.9 et 2.8 sont satisfaits. Si*

$$\forall j \in \mathcal{N}, \quad R_j^B[k] \leq F_j^B[k-1] + l_j^{\max}, \tag{2.13}$$

le niveau du buffer virtuel reste sous le point Q_j^B.

Démonstration. Quand les Théorèmes 2.9 et 2.8 sont satisfaits, le Théorème 2.3 indique que le processus F_j est $\tilde{S}_{L_j}^B$-contraint. Cela veut dire que $\forall j \in \mathcal{N}$ et $\forall s, t \geq 0$:

$$\begin{aligned}
&\tilde{S}_{B_j}^B(s,t) + C_j^B(t) \leq \tilde{S}_{B_j}^B(s,t) + Q_j^B(t), \\
&C_j^B(t) \leq Q_j^B(t).
\end{aligned}$$

A partir de (2.5), on obtient que :

$$\forall j \in \mathcal{N}, \forall t \geq 0, \quad R_j^B(t) \leq F_j^B(t) + Q_j^B(t).$$

On considère le processus $Q_j^B \in \Gamma$ comme le niveau maximal du buffer virtuel, c'est-à-dire, sa taux binaire est la valeur maximale permise par le VL :

$$\forall j \in \mathcal{N}, \quad q_j^B = \frac{l_j^{\max}}{T_{j,BAG}}.$$

Dans ce cas, cette valeur est constante et correspond à une frame du type MFL après une période BAG :

$$\forall j \in \mathcal{N}, \quad Q_j^B[k] = Q_j^B = l_j^{\max},$$

qui doit être respectée dans le domaine discret. □

2.3.2 Stratégie de contrôle

Pour que la boucle fermée BECL soit stable dans chaque VL_j, le processus du feedback R_j^B doit rester dans ces limites $\forall j \in \mathcal{N}$:

$$F_j^B[k-1] < R_j^B[k] \leq F_j^B[k-1] + l_j^{\max}. \tag{2.14}$$

Pour atteindre la meilleure performance, ce processus est défini de la façon suivante :

$$\forall j \in \mathcal{N}, \quad R_j^B[k] = F_j^B[k-1] + l_j^{\max}. \tag{2.15}$$

2.4 Jitter Stabilization Control Loop

La JSCL vise à stabiliser le jitter de la transmission du VL sous une valeur maximale ΔT. Due à une condition particulière, exprimée dans les sous-sections suivantes, ce type de système de contrôle actif peut être appliqué sur un unique VL, appelé le *VL de référence*. Il est représenté par l'indice $m \in \mathcal{N} \backslash \mathcal{I}$.

Comme la Fig. 2.4 montre, la JSCL est composée d'un buffer virtuel positionné après le SM. Ce buffer permet d'interpréter le flux planifié D pour analyser la variation du jitter de chaque VL. En utilisant le processus de la commande C_m^J, le processus du feedback R_m^J

contrôle le flux d'entrée I_m du VL_m.

Dans ce modèle, le SM est représenté comme un composant de l'ES qui produit un jitter aléatoire Δt sur chaque flux des VLs. Pour que les spécifications de AFDX soient satisfaites, la valeur maximale fixée du jitter doit être défini au-dessous de la valeur maximale du jitter, c'est-à-dire, $\forall j \in \mathcal{N},\ \Delta T_j < \Delta t_j^{\max}$.

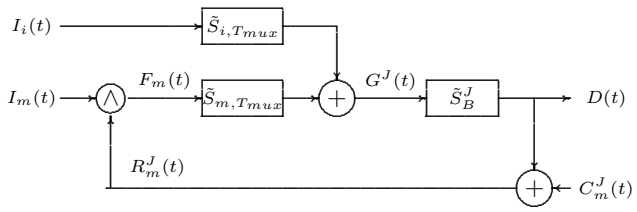


FIGURE 2.4: Représentation du modèle JSCL pour le VL de référence VL_m et $\forall i \in \mathcal{I},\ VL_i$

2.4.1 Limites de la stabilité

Pour maintenir la boucle fermée stable, le processus du feedback doit être contraint par des limites. Pour résoudre cet aspect, on dérive de la Fig. 2.4 les relations entre les différents processus :

$$F_m = R_m^J \wedge I_m, \tag{2.16}$$

$$G^J = F_m \otimes \tilde{S}_{m,T_{mux}} + \sum_{i \in \mathcal{I}} I_i \otimes \tilde{S}_{i,T_{mux}}. \tag{2.17}$$

Les réponses impulsionnelles du buffer virtuel $\tilde{S}_B^J$ et de la commande $\tilde{H}_{C_m}^J$ suivent la même définition que dans les Lemmes 2.5 et 2.6, respectivement.

Théorème 2.12. *On considère Δt_j comme le jitter et M_j, $N_j \in \Gamma$ comme les processus d'entrée et de sortie de SM pour le VL_j. Ensuite, le processus de sortie peut être exprimé par $N_j = M_j \otimes \tilde{S}_{T_{j,mux}} \in \Gamma_0$, avec la réponse impulsionnelle de SM déterminée $\forall j \in \mathcal{N}$, $\forall s, t \geq 0$*

$$\tilde{S}_{T_{j,mux}}(s,t) = \begin{cases} 0, & \textit{si } t - s = \Delta t_j(t); \\ \infty, & \textit{autrement.} \end{cases} \tag{2.18}$$

Démonstration. En considérant le jitter comme un retard de transmission introduit par le SM, la démonstration est la même que dans la Section A.1 de [18]. □

Définition 2.6. *Un flux est qualifié comme orthogonal quand il est possible de le différentier avec les autres flux qui se trouvent dans la même liaison.*

L'utilisation des flux orthogonaux peut simplifier la complexité du modelage. Concrètement, en travaillant seulement avec le VL de référence. En considérant que les frames des VLs appartient à un groupe des flux orthogonaux, grâce à l'identificateur du VL, chaque frame est différentiée à l'entrée du buffer virtuel :

$$G^J = \sum_{j \in \mathcal{N}} G_j^J, \tag{2.19}$$

avec

$$\forall j \in \mathcal{N}, \quad G_j^J = F_j \otimes \tilde{S}_{j,T_{mux}}. \tag{2.20}$$

Néanmoins, la valeur maximale du buffer virtuel Q_m^J du VL_m doit être au courant des états des autres VLs, en particulier du jitter.

En utilisant la Définition 2.6 et l'équation (2.20), on obtient :

$$D_m = G_m^J \otimes \tilde{S}_B^J, \tag{2.21}$$

$$R_m^J = D_m + C_m^J. \tag{2.22}$$

Théorème 2.13. *On considère $\tilde{S}_B^J \in \tilde{\Gamma}_0$ comme la réponse impulsionnelle du buffer virtuel, $\tilde{H}_{C_m}^J \in \tilde{\Gamma}$ comme la réponse impulsionnelle de la commande et $\tilde{S}_{T_{m,mux}}$ comme la réponse impulsionnelle du SM pour le VL_m. La réponse impulsionnelle de la boucle du feedback avec $R_m^J = F_m \otimes \tilde{S}_{L_m}^J$ est déterminée par*

$$\tilde{S}_{L_m}^J = \tilde{S}_{T_{m,mux}} \otimes \tilde{S}_B^J \otimes \tilde{H}_{C_m}^J. \tag{2.23}$$

Démonstration. En appliquant le Lemme 2.6 et le Théorème 2.12 à (2.22) :

$$R_m^J = G_m^J \otimes \tilde{S}_B^J \otimes \tilde{H}_{C_m}^J = F_m \otimes \tilde{S}_{T_{m,mux}} \otimes \tilde{S}_B^J \otimes \tilde{H}_{C_m}^J.$$

La démonstration devient automatique. □

A partir du Théorème 2.13, on dérive :

$$F_m = (F_m \otimes \tilde{S}_{L_m}^J) \wedge I_m. \tag{2.24}$$

Théorème 2.14. *On considère $B^J \in \Gamma_0$ comme le processus de réduction du buffer virtuel, C_m^J comme le processus de la commande et $\Delta t_m^{\max}$ comme la valeur maximale du jitter pour le VL_m. Si*

$$\forall t \geq 0, \quad C_m^J(t) > B^J(t + \Delta t_m^{\max}) - B^J(t), \tag{2.25}$$

$F_m = I_m \otimes (\tilde{S}_{L_m}^J)^*$ est la solution unique de (2.24).

Démonstration. La démonstration suit les mêmes étapes que dans la démonstration du Théorème 2.8. Néanmoins, dans ce cas, $\tilde{S}_{L_m}^J$ est défini de la façon suivante :

$$\tilde{S}_{L_m}^J = \tilde{S}_{T_{m,mux}} \otimes \tilde{S}_B^J \otimes \tilde{H}_{C_m}^J$$

avec $\forall s, t \geq 0$

$$(\tilde{S}_{T_{m,mux}} \otimes \tilde{S}_B^J)(s,t) = B^J(t) - B^J(s + \Delta t_m(f(s))),$$

Il faut mentionner que $f(s)$ est la solution maximale de $t - \Delta t_m(t) = s$ pour une spécifique valeur de $s \in [0,t]$. Pour cette raison, (2.9) doit être satisfaite et

$$\forall t \geq 0, \forall s \in [0,t], \quad C_m^J(t) > B^J(s + \Delta t_m(f(s))) - B^J(t).$$

Dans le pire des cas $s = t$

$$\forall t \geq 0, \quad C_m^J(t) > B^J(t + \Delta t_m(f(s))) - B^J(t).$$

Comme c'est possible de limiter le jitter à sa valeur maximale, le limite du pire des cas est exprimé par (2.25). □

Théorème 2.15. *On considère $B^J \in \Gamma_0$ comme le processus de réduction du buffer virtuel, C_m^J comme le processus de la commande et $\Delta t_m^{\max}$ comme la valeur maximale du jitter pour le VL_m. Si*

$$\forall t \geq 0, \quad C_m^J(t) \geq B^J(t + \Delta t_m^{\max}) - B^J(t), \tag{2.26}$$

$\tilde{S}_{L_m}^J$ est sous-additive.

Démonstration. En utilisant la Définition 2.4, il est possible de montrer que $\forall s, t \geq 0$ et $\forall \tau \in [s, t]$:

$$\begin{aligned} &\tilde{S}_{L_m}^J(s,\tau) + \tilde{S}_{L_m}^J(\tau,t) - \tilde{S}_{L_m}^J(s,t) \geq 0; \\ &C_m^J(\tau) \geq B^J(\tau + \Delta t_m(f(s))) - B^J(\tau). \end{aligned}$$

La démonstration continue en spécifiant $\tau = t$ et la valeur maximale du jitter. □

Grâce au Théorème 2.1, la solution unique de (2.24) devient :

$$F_m = I_m \otimes \tilde{S}_{L_m}^J.$$

Théorème 2.16. *On considère D_m comme le processus de sortie de l'ES avec les frames du VL de référence (VL_m), $\Delta t_m^{\max}$ comme la valeur maximale du jitter, $T_{m,BAG}$ comme la période BAG et k comme l'indice du temps discret pour le VL_m. Si*

$$R_m^J[k] > D_m\left[k - 1 + \left\lceil \frac{\Delta t_m^{\max}}{T_{m,BAG}} \right\rceil\right], \tag{2.27}$$

les Théorèmes 2.15 et 2.14 sont satisfaits.

Démonstration. En utilisant (2.22) dans le Théorème 2.14, la condition devient de la façon suivante :

$$\forall t \geq 0, \quad R_m^J(t) - D_m(t) > B^J(t + \Delta t_m^{\max}) - B^J(t).$$

En considérant que le buffer virtuel n'a pas de pertes, son processus de réduction peut être défini comme $B^J = D_m$. Pour cette raison :

$$\forall t \geq 0, \quad R_m^J(t) > D_m(t + \Delta t_m^{\max}).$$

Comme dans le cas de la BECL, cette condition doit être respectée dans le domaine du temps discret du VL de référence, c'est-à-dire, à chaque période $T_{m,mux}$ du VL_m. En considérant le domaine discret, le pire cas est la valeur supérieure entre la valeur maximale du jitter et la période BAG. □

Théorème 2.17. *On considère D_m comme le processus de sortie de l'ES avec des frames du VL de référence, Bw comme la capacité de la liaison, ΔT_i comme la valeur maximale fixée du jitter, Δt_i comme le jitter pour le VL_i et k comme l'indice du temps discret pour le VL_m. On suppose que les des Théorèmes 2.15 et 2.14 sont satisfaites. Si*

$$R_m[k] \leq D_m[k-1] + Bw \min_{i \in \mathcal{I}} \left\{ (\Delta T_i - \Delta t_i[k])^+ \right\}, \tag{2.28}$$

le niveau du buffer virtuel reste sous le point Q_m^J.

Démonstration. Quand les conditions des Théorèmes 2.15 et 2.14 sont satisfaites, le processus d'entrée du buffer virtuel est $\tilde{\Lambda}$-contraint :

$$\tilde{\Lambda} = \tilde{S}^J_{L_m} \otimes \tilde{S}_{T_{m,mux}} = \tilde{S}_{T_{m,mux}} \otimes \tilde{S}_l + C_l,$$

avec $\forall t \geq 0$ et $\forall s \in [0, t]$:

$$\tilde{S}_l(s,t) = (\tilde{S}^J_B \otimes \tilde{S}_{T_{m,mux}})(s,t) = B^J(t - \Delta t_m(t)) - B^J(s),$$
$$C_l(t) = (C^J_m \otimes \tilde{S}_{T_{m,mux}})(s,t) = C^J_m(t - \Delta t_m(t)).$$

En considérant $\tilde{S}_{T_{m,mux}} \otimes \tilde{S}_l \leq \tilde{S}_l$:

$$\forall t \geq 0, \forall s \in [0,t] \quad \tilde{S}_l(s,t) + C_l(t) \leq \tilde{S}^J_B(s,t) + Q^J_m(t).$$

Alors,

$$C^J_m(t - \Delta t_m(t)) \leq B^J(t) - B^J(t - \Delta t_m(t)) + Q^J_m(t).$$

En spécifiant $t - \Delta t_m(t) \to t$

$$C^J_m(t) \leq B^J(t + \Delta t_m(t)) - B^J(t) + Q^J_m(t + \Delta t_m(t)).$$

Cette condition doit être satisfaite quand $B(t + \Delta t_m(t)) - B(t)$ est minimale, c'est-à-dire, quand $\Delta t_m(t) = \Delta t^{\min}_m$. Comme le jitter est bien délimité par $\Delta t^{\max}_m \geq \Delta t_m(t) \geq 0$, on obtient

$$\forall t \geq 0, \quad C^J_m(t) \leq Q^J_m(t).$$

À partir de (2.22), ce possible de montrer la relation suivante :

$$\forall t \geq 0, \quad R^J_m(t) \leq D_m(t) + Q^J_m(t).$$

$Q^J_m \in \Gamma$ représente la valeur maximale du niveau du buffer virtuel que son taux de frames correspond à la quantité maximale de bits $bits^{\max}_m$ qui ne font pas dépasser la valeur maximale fixée du jitter ΔT_i, c'est-à-dire :

$$q^J_m(t) = \frac{bits^{\max}_m(t)}{T_{m,BAG}}.$$

$bits^{\max}_m$ dépend des jitters des VL_i, $\forall i \in \mathcal{I}$. Pour cette raison, pour assurer que les exigences de tous les VLs sont satisfaites, il doit avoir le taux de transmission plus petit :

$$bits^{\max}_m(t) = \min_{i \in \mathcal{I}} \{ bits^{\max}_m(t)|_i \},$$

avec, $\forall i \in \mathcal{I}$

$$bits^{\max}_m(t)|_i = bits^{\max}_T|_i - \sum_{q \in \mathcal{I};\, q \neq i} bits_q(t), \tag{2.29}$$

$$bits^{\max}_T|_i = Bw\ \Delta T_i. \tag{2.30}$$

En effet, il est la combinaison entre les bits permis pour atteindre la valeur maximale fixée du jitter $bits^{\max}_T|_i$ et les bits envoyés depuis les autres VLs $bits_q(t)$. Cela est traduit selon l'expression suivante :

$$Q^J_m[k] = Bw\ \min_{i \in \mathcal{I}} \{ \Delta T_i - \Delta t_i[k] \}, \tag{2.31}$$

$$\Delta t_i[k] = \sum_{q \in \mathcal{I};\, q \neq i} \frac{bits_q[k]}{Bw}. \tag{2.32}$$

Comme $\forall j \in \mathcal{N},\ \Delta T_j < \Delta t_j^{\max}$, $Q_m^J[k]$ (2.31) peut devenir une valeur négative. Pour cette raison, on le contraint à une valeur positive :

$$Q_m^J[k] = Bw \min_{i \in \mathcal{I}} \{(\Delta T_i - \Delta t_i[k])^+\}. \tag{2.33}$$

La condition peut être respectée dans le domaine discret, ce qui complète la démonstration. □

2.4.2 Stratégie de contrôle

Pour que la boucle fermée JSCL soit stable dans le VL de référence (VL_m), le processus du feedback $R_m^J[k]$ doit rester dans ces limites :

$$\begin{aligned} R_m[k] &> D_m \left[k - 1 + \left\lceil \frac{\Delta t_m^{\max}}{T_{m,BAG}} \right\rceil \right], \\ R_m[k] &\leq D_m[k-1] + Bw \min_{i \in \mathcal{I}} \{(\Delta T_i - \Delta t_i[k])^+\}. \end{aligned}$$

On note que la relation suivante n'est pas toujours établie :

$$\begin{aligned} D_m \left[k - 1 + \left\lceil \frac{\Delta t_m^{\max}}{T_{m,BAG}} \right\rceil \right] &< D_m[k-1] + Bw\ \rho[k], \\ \rho[k] &= \min_{i \in \mathcal{I}} \{(\Delta T_i - \Delta t_i[k])^+\}. \end{aligned}$$

Pour cette raison, une adaptation du point limite Q_{new}^J est introduite pour que toujours ces limites soient satisfaites.

Théorème 2.18. *On considère D_m comme le processus de sortie de l'ES avec des frames du VL de référence, Bw comme la capacité de la liaison, ΔT_i comme la valeur maximale fixée du jitter, Δt_i comme le jitter pour le VL_i, $\Delta t_m^{\max}$ comme la valeur maximale du jitter, ϵ comme la période de bit et k comme l'indice du temps discret pour le VL_m. On suppose que les conditions des Théorèmes 2.15 et 2.14 sont satisfaites. Si*

$$\begin{aligned} R_m[k] &\leq D_m[k-1] + Bw\ \max\{\rho[k],\ \gamma\}, \\ \gamma &= \Delta t_m^{\max} + \epsilon, \\ \rho[k] &= \min_{i \in \mathcal{I}} \{(\Delta T_i - \Delta t_i[k])^+\}, \end{aligned} \tag{2.34}$$

Le niveau du buffer virtuel reste sous la valeur Q_{new}^J définie de la façon suivante

$$Q_{new}^J(t) = Bw\ \max\{\rho(t),\ \gamma\} \tag{2.35}$$

avec

$$\rho(t) = \min_{i \in \mathcal{I}} \{(\Delta T_i - \Delta t_i(t))^+\}.$$

On note que cette stratégie est valide parce que $\Delta T_i < \Delta t_i^{\max}$.

Démonstration. La démonstration suit la même idée que celle utilisée dans la démonstration du Théorème 2.17.

Comme la Fig. 2.5 montre, Q_{new}^J doit être capable d'offrir un feedback en fonction du jitter au même temps qu'il satisfait (2.34). Pour cette raison, quand $Q_{new}^J(t) = Bw\ \gamma$:

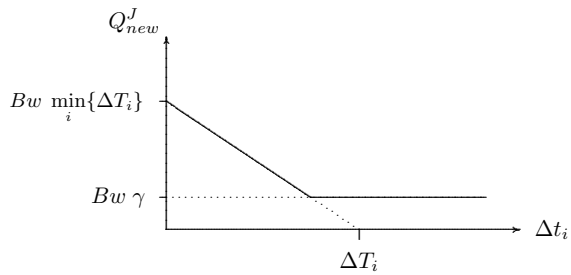


FIGURE 2.5: Représentation du nouveau processus de point limite

$$\forall t \geq 0, \quad D_m(t + \Delta t_m^{\max}) < D_m(t) + Bw\ \gamma.$$

Alors,

$$\forall t \geq 0, \quad D_m(t + \Delta t_m^{\max}) - D_m(t) < Bw\ \gamma.$$

$D_m(t + \Delta t_m^{\max}) - D_m(t)$ peut avoir différentes valeurs selon la valeur maximale du jitter. Néanmoins, seulement la situation dans laquelle $\Delta t_m^{\max}$ n'est pas suffisante pour envoyer une frame de longueur $l^{\max}$ devient acceptable par les réseaux AFDX :

$$\Delta t_m^{\max} < \frac{l^{\max}}{Bw} \rightarrow \frac{D_m(t + \Delta t_m^{\max}) - D_m(t)}{Bw} \leq \Delta t_m^{\max}. \tag{2.36}$$

Ensuite, γ doit satisfaire la condition suivante :

$$\gamma > \Delta t_m^{\max},$$

Pour cette raison, la définition de γ devient :

$$\gamma = \Delta t_m^{\max} + \epsilon = \Delta t_m^{\max} + \frac{1}{Bw}.$$

La stabilité des limites est respectée dans le domaine discret. □

Pour obtenir la propriété montrée dans la Fig. 2.5, la prochaine contrainte doit être respectée pendant le design :

$$\gamma = \Delta t_m^{\max} + \epsilon < \min_{i \in \mathcal{I}}\{\Delta T_i\}. \tag{2.37}$$

En effet, cette contrainte est forte, ce qui implique que le VL de référence est celui qui a la plus petite valeur maximale du jitter. Si, en revanche, la condition (2.36) n'est pas satisfaite, (2.14) ne peut pas être assurée.

Avec ce nouveau limite, le système sera stable tant que le processus de feedback $R_m^J[k]$ reste dans ces limites :

$$\begin{aligned} R_m[k] &> D_m\left[k - 1 + \left\lceil \frac{\Delta t_m^{\max}}{T_{m,BAG}} \right\rceil\right], \\ R_m[k] &\leq D_m[k-1] + Bw\ \max\{\rho[k],\ \gamma\}. \end{aligned} \tag{2.38}$$

Pour cette raison, la définition de ce processus devient la suivante :

$$\begin{aligned} R_m[k] &= D_m[k-1] + Bw \ \max\{\rho[k],\ \gamma\}, \\ \gamma &= \Delta t_m^{\max} + \epsilon, \\ \rho[k] &= \min_{i\in\mathcal{I}}\{(\Delta T_i - \Delta t_i[k])^+. \end{aligned} \tag{2.39}$$

2.4.3 Prédiction du jitter $\widehat{\Delta t}$

Évidemment, l'implémentation du processus de feedback exprimée par (2.39) a besoin de la connaissance du jitter $\Delta t_i[k]$, ce qui n'est pas disponible à l'instant k. Pour cette raison, un schéma basé sur la prédiction du jitter est considéré ($\widehat{\Delta t}$) :

$$\hat{\rho}[k] = \min_{i\in\mathcal{I}}\{(\Delta T_i - \widehat{\Delta t}_i[k])^+\}. \tag{2.40}$$

La prédiction du jitter peut être réalisée au niveau du SM, parce qu'il est conscient des périodes BAG de tous les VLs.

Théorème 2.19. *On considère $T_{i,BAG}$ comme la période BAG, $l_i^{\max}$ comme le MFL pour le VL_i, Bw comme la capacité de la liaison et k comme l'indice du temps discret pour le VL_m. En considérant le prochain sous-ensemble d'indices de $\mathcal{N}$:*

$$\mathcal{J}_k = \{p \in \mathcal{I} \mid kT_{m,BAG} \mod T_{p,BAG} = 0\} \subset \mathcal{N}.$$

Ensuite, pour tous les $i \in \mathcal{I}$ le jitter prédit est exprimé de la façon suivante :

$$\begin{aligned} \widehat{\Delta t}_i[k] &= \min\left\{\left\lfloor \frac{T_{i,BAG} - \beta_i[k]}{T_{i,BAG}} \right\rfloor, 1\right\} \sum_{p\in\mathcal{J}_k;\, p\neq i} \frac{l_p^{\max}}{Bw}, \\ \beta_i[k] &= kT_{m,BAG} \mod T_{i,BAG}. \end{aligned} \tag{2.41}$$

Démonstration. La prédiction du jitter est réalisée à l'instant $k-1$. Quand $kT_{m,BAG} \mod T_{i,BAG} \neq 0$, la frame qui appartient au VL_i ne va pas se trouver dans le SM à l'instant k, c'est-à-dire, $\widehat{\Delta t}_i[k] = 0$. Quand $kT_{m,BAG} \mod T_{i,BAG} = 0$, la frame qui appartient au VL_i va se trouver dans le SM à l'instant k. Dans ce cas, le jitter est introduit par les frames qui appartient à $\mathcal{J}_k$. Comme la taille de la frame ne peut pas être prédite, le pire cas doit être contraint par :

$$\widehat{\Delta t}_i[k] = \sum_{p\in\mathcal{J}_k;\, p\neq i} \frac{l_p^{\max}}{Bw}.$$

□

Finalement, en utilisant la définition de la période BAG, le jitter prédit devient périodique avec une période exprimée par

$$\varrho = \frac{\max_{j\in\mathcal{N}}\{T_{j,BAG}\}}{T_{m,BAG}}.$$

2.5 Contrôle de la composition

Le contrôle de la composition entre la BECL et la JSCL peut être atteint en utilisant l'opérateur min :

$$R_m[k] = \min\{R_m^B[k],\ R_m^J[k]\}. \tag{2.42}$$

Ce contrôle peut être appliqué que sur un VL unique, tandis que les autres peuvent utiliser la BECL. On note que même si le contrôle peut assurer la stabilité de la boucle fermée, il peut se produire un scénario non désiré. Plus précisément, quand $R_m^J[k]$ est toujours plus grand que $l_m^{\max}$ le feedback devient inutile. Pour éviter cette situation, la prochaine contrainte est établie :

$$(R_m^J[k])^{\min} = Bw\ \gamma + D_m[k-1] < D_m[k-1] + l_m^{\max}.$$

En utilisant les définitions de γ et $\Delta t_m^{\max}$, cette contrainte devient de la façon suivante :

$$1 + \sum_{i \in \mathcal{I}} l_i^{\max} < l_m^{\max}. \tag{2.43}$$

En conclusion, pour maximiser le débit en même temps que la stabilité est assurée, les conditions (2.36), (2.37) et (2.43) doivent être satisfaites. Ce schéma de contrôle peut atteindre une haute utilisation de la bande passante en même temps qu'il maintient le jitter sous une valeur fixée. Toutefois, due à (2.37), il peut exister une petite variance dans la limite supérieure du jitter, qui est définie de la façon suivante :

$$\begin{aligned} \Delta jit[k] &= (\gamma - \hat{\rho}[k])^+, \\ \hat{\rho}[k] &= \min_{i \in \mathcal{I}}\{(\Delta T_i - \widehat{\Delta t}_i[k])^+\}. \end{aligned} \tag{2.44}$$

Due au fait que $\widehat{\Delta t}_i[k]$ est périodique, ce système de contrôle actif peut assurer une valeur limite du jitter :

$$\forall j \in \mathcal{N}, \quad \Delta t_j \leq \min_{i \in \mathcal{I}}\{\Delta T_i\} + \max_{k \leq \varrho}\{\Delta jit[k]\}. \tag{2.45}$$

2.6 Étude de cas

Pour valider le développement des schémas de contrôle présentés, on considère un ES partagé par 4 VLs dans un réseau avec une capacité de $Bw = 100$ Mbps. Les paramètres de ces VLs sont montrés dans la Table 2.1. Dans ce cas, le système de contrôle actif (BECL et JSCL) est appliqué que sur le 4eme VL, VL_4. Les outils `TrueTime 1.5` et `Matlab` ont été utilisés pour implémenter le simulateur.

Les valeurs théoriques du jitter prédit $\widehat{\Delta t}$, $Bw\,\rho$ et $Bw\,\gamma$ sont représentées sur la Table 2.2. Les résultats des simulations sont présentés dans la Fig. 2.6.

La Figure 2.6b illustre le flux de sortie du SM qui correspond à une transmission normale. La représentation de la longueur des frames de chaque VL surligne le fait que le système de contrôle actif ajoute des frames avec une taille variable, en fonction du processus de feedback R_4, au même temps que le VL_4 transmet de l'information. On note aussi un incrément de la bande passante instantanée du VL_4 sans interférer sur la data originale. Particulièrement,

Table 2.1: Paramètres des VLs

	VL_1	VL_2	VL_3	VL_4
T_{app}[ms]	150	255	100	90
T_{BAG}[ms]	128	128	64	64
$l^{\max}$[bytes]	450	450	450	1500
ΔT[ms]	0.15	0.19	0.15	0.1
$\Delta t^{\max}$[ms]	0.192	0.192	0.192	0.108

Table 2.2: Les valeurs théoriques

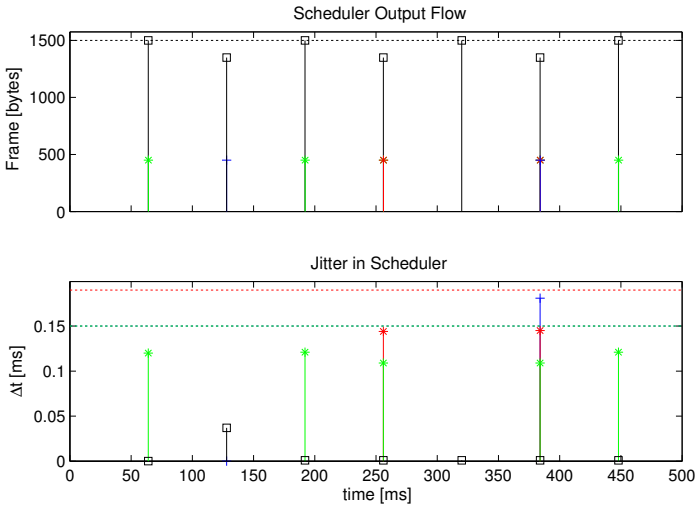
	$k=1$	$k=2$
$\widehat{\Delta t}_1$ [ms]	0	0.072
$\widehat{\Delta t}_2$ [ms]	0	0.072
$\widehat{\Delta t}_3$ [ms]	0	0.072
$Bw\,\rho$ [bytes]	1875	975
$Bw\,\gamma$ [bytes]	1350	1350
Δjit [ms]	0	0.03

la quantité de bits transmises du VL_4 pendant les premiers $450\,ms$ a augmenté un 78% en utilisant le feedback du contrôle actif.

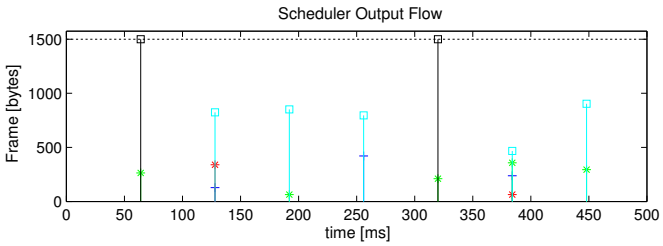
Pour bien analyser le comportement du système de contrôle dans le pire cas, une nouvelle simulation a été exécutée avec la caractéristique de forcer la transmission des frames avec une taille de MFL à chaque VL. De plus, le flux d'entrée à VL_4 a été enlevé pour mieux illustrer l'effet du système de contrôle. La Figure 2.6a montre que, dans ce cas, les jitters des VLs sont plus petits que sa valeur maximale fixée ΔT avec l'exception à l'instant $t = 384$ ms. Dans ce moment, le pire scénario est apparu : l'accumulation de tous les frames des VL dans le SM. Dans ce cas, le feedback pour respecter la valeur maximale fixée du jitter serait 975 bytes. Cependant, comme la condition (2.37) doit être satisfaite, le feedback exige $Bw\gamma = 1350$ bytes. Ce situation introduit une variation sur le jitter du VL_1 montrée dans la figure ($\Delta t_1 = 0.18$ ms).

Pour mieux comprendre cette situation, une autre simulation a été lancé, dans laquelle chaque flux d'entrée des VLs transmet des frames à chaque période BAG. De plus, le processus de feedback n'est plus limité par la contrainte du Théorème 2.17, c'est-à-dire, $\gamma = 0$. Dans la Fig. 2.6c, il est surligné que les jitters deviennent toujours plus petits ou égales à la valeur maximale fixée du jitter ΔT. Même si ce résultat est prometteur, la stabilité de la boucle fermée ne peut pas être évaluée en se basant sur l'analyse de NC. Pour cette raison, une intéressante recherche pourrait être l'utilisation d'un outil mathématique moins pessimiste pour l'analyse du système, comme par exemple *Stochastic Network Calculus*.

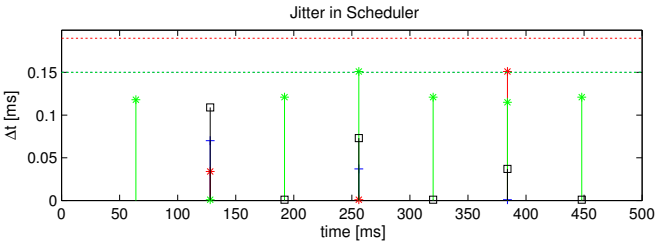
Bien que cette simulation a été lancée avec une seule boucle fermée sur le VL_4, le système de contrôle actif peut être implémenté avec une BECL sur chaque VL et une JSCL sur le VL avec la valeur maximale du jitter plus petite. Ce scénario serait plus proche du pire cas, comme la Fig. 2.6a. Il est possible de voir que le système devient stable dans ce scénario et maintient le jitter sous la valeur maximale fixée.



(a) Longueur de la frame et jitter quand le VL_4 n'envoie jamais des frames



(b) Longueur de la frame dans une transmission normale du VL_4



(c) Jitter quand les applications envoient à chaque période BAG et $\gamma = 0$

FIGURE 2.6: Longueur de la frame et jitter après le SM : bleu-croix pour le VL_1 ; rouge-étoile pour le VL_2 ; vert-étoile pour le VL_3 ; bleu ciel-carré pour le VL_4 ; noire-carré pour les frames du système de contrôle.

2.7 Conclusion

Ce chapitre développe des nouveaux schémas pour le contrôle du flux de VL dans un réseau AFDX, qui permettent d'améliorer l'efficience de la bande passante au même temps qu'ils stabilisent le jitter. Les conditions pour garantir la stabilité de la boucle fermée sont dérivées en utilisant la théorie de Network Calculus, en se basant sur celles qui ont été présentés dans d'autres articles.

Il est démontré que le contrôle actif peut améliorer le déterminisme du réseau AFDX, qui est une considération importante pour le support des applications critiques. Néanmoins, on note que les conditions (2.36), (2.37) et (2.43) sont un peu restrictives. Ces conditions peuvent limiter l'application de ces schémas de contrôles. Cependant, l'expérimentation a montré que ces conditions pourraient être relaxées. Pour cette raison, il serait intéressant de déterminer de conditions moins restrictives.

Finalement, ce serait aussi intéressant d'identifier quelle type de données peuvent être transportées à partir du système de contrôle actif dans un scénario des applications de temps réel.

2.8 Références

[1] ARINC 664. *Aircraft Data Network Part 7 Avionics Full-Duplex Switched Ethernet Network*. AERONAUTICAL RADIO INC., 2009.

[2] M. Boyer and C. Fraboul. Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus. In *Proc. of IEEE WFCS*, pages 11–20, May 2008.

[3] H. Bauer, J. Scharbarg, and C. Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *Industrial Informatics, IEEE Transactions on*, 6(4) :521–533, Nov 2010.

[4] Melhem Tawk, Guchuan Zhu, Yvon Savaria, Xue Liu, Jian Li, and Fei Hu. A tight end-to-end delay bound and scheduling optimization of an avionics AFDX network. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 7B3–1. IEEE, 2011.

[5] Xuesong Zhang, Xin Chen, Lei Zhang, Guofeng Xin, and Tong Xu. End-to-end delay analysis of avionics full duplex switched ethernet with different flow scheduling scheme. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 4, pages 2252–2258, dec. 2011.

[6] Yu Hua and Xue Liu. Scheduling heterogeneous flows with delay-aware deduplication for avionics applications. *Parallel and Distributed Systems, IEEE Transactions on*, 23(9) :1790–1802, Sep. 2012.

[7] Jian Li, Haibing Guan, Jianguo Yao, Guchuan Zhu, and Xue Liu. Performance enhancement and optimized analysis of the worst case end-to-end delay for AFDX networks. In *Proc. of IEEE GreenCom*, pages 301 –310, Nov. 2012.

[8] M Boyer, N Navet, and M Fumey. Experimental assessment of timing verification techniques for AFDX. In *Proc. of ERTS, Toulouse, France*, 2012.

[9] Xiaoting Li, J.-L. Scharbarg, and C. Fraboul. Improving end-to-end delay upper bounds on an afdx network by integrating offsets in worst-case analysis. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–8, Sep. 2010.

[10] J.-L. Scharbarg, F. Ridouard, and C. Fraboul. A probabilistic analysis of end-to-end delays on an AFDX avionic network. *Industrial Informatics, IEEE Transactions on*, 5(1) :38–49, Feb. 2009.

[11] Meng Li, Martin Lauer, Guchuan Zhu, and Yvon Savaria. Determinism Enhancement of AFDX Networks via Frame Insertion and Sub-Virtual Link Aggregation. *Industrial Informatics, IEEE Transactions on*, 10(3) :1684–1695, 2014.

[12] Simon Dobson. An adaptive systems perspective on network calculus, with applications to autonomic control. *International Journal of Autonomous and Adaptive Communications Systems*, 1(3) :332–341, 2008.

[13] Mingyu Chen, Jinsong Zhang, Manohar N Murthi, and Kamal Premaratne. TCP congestion avoidance : A network calculus interpretation and performance improvements. In *Proc of the INFOCOM 2005*, volume 2, pages 914–925, 2005.

[14] Mingyu Chen, Manohar N Murthi, Kamal Premaratne, and Xingzhe Fan. A Congestion Control Method Jointly Utilizing Delay and Marking/Loss Feedback. In *GLOBECOM*. Citeseer, 2006.

[15] Mingyu Chen, Jinsong Zhang, Manohar N Murthi, and Kamal Premaratne. Delay-based TCP congestion avoidance : A network calculus interpretation and performance improvements. *Computer Networks*, 53(9) :1319–1340, 2009.

[16] Paola Cappanera, Luciano Lenzini, Alessandro Lori, Giovanni Stea, and Gigliola Vaglini. Efficient link scheduling for online admission control of real-time traffic in wireless mesh networks. *Computer Communications*, 34(8) :922–934, 2011.

[17] Lianming Zhang, Jianping Yu, and Xiaoheng Deng. Modelling the guaranteed QoS for wireless sensor networks : a network calculus approach. *EURASIP Journal on Wireless Communications and Networking*, 2011(1) :1–14, 2011.

[18] J Zhang, K Premaratne, and Peter H Bauer. Resource allocation and congestion control in distributed sensor networks-a network calculus approach. In *Proceedings of Fifteenth International Symposium on Mathematical Theory of Networks and Systems*, 2002.

[19] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus : a theory of deterministic queuing systems for the internet*, volume 2050. Springer Science & Business Media, 2001.

[20] Rajeev Agrawal, François Baccelli, and Rajendran Rajan. An algebra for queueing networks with time varying service and its application to the analysis of integrated service networks. *Mathematics of Operations Research*, 29(3) :559–591, 2004.

Chapitre 3

Modélisation de la planification de partitions pour différents niveaux d'états critiques

3.1 Introduction

Pour satisfaire l'augmentation de la demande sur la puissance de calcul et de bande passante au même temps qu'il est offert une réduction de la taille, poids et consommation de puissance (SWaP), l'industrie de l'aviation a adopté l'architecture Intégrée Modulaire Avionique (IMA) [1, 2] pour les systèmes avioniques. Contrairement à l'architecture traditionnelle fédérée, l'architecture IMA permet aux fonctions de différents sous-systèmes de partager la même plateforme de computation, en améliorant de cette façon l'efficience d'utilisation des ressources.

Pour éviter des situations dans lesquelles la faillite d'une fonction peut se propager jusqu'à affecter les autres, concrètement celles qui ont un niveau d'état critique plus petit à celles qui l'ont plus grand, les standards ARINC 651 pour le IMA [1] et ARINC 653 pour les spécifications du système d'exploitation (OS) sous-adjacent [3] imposent le paradigme du *partitioning*. Ce partage en espace et temps permet la création d'un mécanisme de confinement sans faillites pour le hardware et le software. Évidemment, partager les mêmes ressources entre fonctions de différents sous-systèmes qui ont différents niveaux critiques fait que le design et la performance du système sont plus complexes. Pour cette raison, le développement des outils qui automatisent le processus de design du système (d'une façon optimale) sont de plus en plus intéressant.

Le travail présenté dans ce chapitre est orienté sur le partage temporel dans les plateformes d'ARINC-653. Dans ce contexte, le partage vise à déterminer un ensemble de ressources du type software qui assurent une isolation temporelle robuste pour les processus, nommés *tâches*. Le temps d'exécution est alloué à chaque *partition*, ainsi qu'aux tâches qui le composent, à travers de la spécification de frames d'exécution, nommées *instances*. Il faut indiquer que pour assurer une stricte sécurité pour les systèmes critiques, les paramètres du partage temporel sont configurés statiquement dans l'IMA, c'est-à-dire, ils ne peuvent pas être changés pendant l'exécution. Pour cette raison, le problème considéré peut être modélisé dans le cadre de la programmation mathématique.

Le design des systèmes ARINC-653 vise à déterminer l'allocation des partitions temporels et l'ordonnancement des tâches dans chaque partition. Dans ce contexte, un modèle et un

design, en utilisant une procédure du type *ad hoc*, sont proposés dans [4] pour optimiser le débit du système au même temps qui préserve les niveaux critiques des tâches.

Une proposition plus sophistiquée en utilisant un ordonnancement basé sur deux niveaux hiérarchiques (priority-driven algorithm) est introduite dans [5] pour le design d'une classe générique de système en temps-réel. Même si cette procédure de design fonctionne bien pour ce type de tâches, elle ne considère pas la préemption des tâches, c'est-à-dire, l'interruption de l'exécution des tâches due à l'émergence d'autres processus qui ont une priorité plus haute.

D'autres propositions basées sur un ordonnancement hiérarchique ont été aussi développées, comme par exemple le travail reporté dans [6, 7, 8, 9, 10]. Il est noté que dans le travail susmentionné le design est réalisé d'une façon récursive et quelques paramètres (l'ordre d'exécution, la distribution des tâches, etc.) doivent être imposés *a priori* pour atteindre un problème solvable. De plus, la capacité que le système complet soit planifié doit être aussi évaluée *a posteriori*. Même si une procédure de design peut être réalisée successivement ou récursivement entre les deux niveaux, il existe un manque d'automatisme pour le réglage. Pour cette raison, il est difficile d'atteindre une optimisation globale en termes d'utilisation du processeur et de performance du système complet.

Une structure analytique pour formuler la configuration des partitions avec multiples niveaux critiques est proposée dans [11]. Ce schéma est basé sur la théorie des graphes qui permet de représenter l'interaction entre les tâches. Même si cette technique simplifie la représentation d'un modèle non-linéaire, le comportement du système complet n'est pas défini et la préemption n'est pas considérée.

Un modèle qui capture le comportement essentiel d'un système ARINC-653 est présenté dans [12, 13], ce qui permet de créer un design automatique pour l'allocation des partitions et l'ordonnancement des tâches. Néanmoins, le modèle utilisé dans le design et l'optimisation est essentiellement non-linéaire, ce qui demande de grandes ressources de calcule. De plus, l'utilisation des heuristiques pour résoudre ce type de problèmes (non-linéaires) n'offre aucune garantie en termes d'un optimum global.

Le travail présenté dans ce chapitre introduit un modèle entièrement linéaire pour l'allocation des partitions et l'ordonnancement des tâches simultanées avec la préemption des tâches. Ce modèle constitue la base pour calculer des solutions réalisables dans le problème d'ordonnancement. En plus, il est suffisamment générique pour spécifier une large gamme des considérations du design, comme par exemple le temps de réponse global, le temps d'allocation de tous les partitions ou/et le temps d'utilisation des processus.

Le modèle accepte différents spécifications des tâches, comme par exemple le temps d'exécution dans le pire cas (WCET), la périodicité, la date limite, la priorité et l'ordre relatif entre les exécutions. Par conséquent, une configuration [1] caractérisée par une distribution des tâches entre partitions et le temps de début et final des partitions, peut être atteint en même temps qu'il y a une garantie de l'ordonnancement du système global.

Il doit être mentionné que ce modèle est composé de variables entières et que toutes les contraintes sont formulées à travers des égalités et inégalités linéaires, ce qui maintient le problème de computation tractable. Pour cette raison, ce problème peut être résolu par beaucoup des algorithmes existants du type entière-linéaire [14, 15]. Le logiciel `IBM ILOG`

1. Si cette configuration existe pour le problème considéré.

`CPLEX Optimization Studio` a été utilisé pendant le travail de modélisation.

Il est noté que même si ce modèle est formulé dans le contexte des applications ARINC-653, le modèle proposé est applicable sur une large gamme de problèmes sur la communication du réseau [16, 17, 18], de l'informatique [19, 20, 21, 22, 23] et du management de la production [24, 25, 26, 27, 28]. De plus, ce modèle est flexible, c'est-à-dire, il permet d'ajouter des contraintes supplémentaires pour résoudre des problèmes plus spécifiques à une application donnée.

Le chapitre est organisé de la façon suivante. La Section 3.2 introduit les contraintes et les variables du problème utilisées pour la modélisation. La Section 3.3 détaille les contraintes impliquées dans la formulation du modèle. Les critères de performance et les objectifs d'optimisation sont définis dans la Section 3.4. La Section 3.5 présente une étude de cas et les résultats obtenus. Finalement, quelques remarques de conclusion sont apportées dans la Section 3.6.

3.2 Partage temporel pour des systèmes avec plusieurs niveaux critiques

Cette section introduit les notations utilisées pour la formulation du problème. Le temps est vu comme une variable discrète, en prenant des valeurs dans l'ensemble $\mathbb{R}_+$ avec un pas de temps $\Delta t \in \mathbb{R}_+^*$. Il est adopté que toutes les constantes et variables temporelles sont normalisées par $\Delta t = 1$.

Pour simplifier les notations, on introduit :

$$\forall a \leq b \in \mathbb{Z},\ \mathcal{I}_{a,b} \triangleq \{a, a+1, \ldots, b\},\ \mathcal{I}_a \triangleq \mathcal{I}_{1,a},\ \mathcal{J}_a \triangleq \mathcal{I}_{0,a}.$$

3.2.1 Déclaration du problème

Indices

Les indices suivants sont utilisés dans la modélisation du problème :

— $i \in \mathbb{N}^*$: indice qui représente les tâches ;
— $j \in \mathbb{N}^*$: indice qui représente les partitions ;
— $k \in \mathbb{N}^*$: indice qui représente les instances des partitions ;
— $l \in \mathbb{N}$: indice qui représente le temps d'exécution.

On note que l'indice du temps l caractérise l'instant $l\Delta t$. Avec la normalisation des constantes et variables temporelles, l fait référence à l'indice du temps et le temps d'exécution.

Tâches

On considère $n \in \mathbb{N}^*$ comme le nombre de tâches intégrées τ_i ($i \in \mathcal{I}_n$) et on dénote par $\mathcal{T} = \{\tau_i,\ i \in \mathcal{I}_n\}$ l'ensemble composé de toutes les tâches. Comme la Figure 3.1 illustre, une tâche est définie par le 4-tuple $\tau_i = (C_i, D_i, T_i, \rho_i)$ avec $C_i \in \mathbb{N}^*$ comme le pire temps

d'exécution (normalisé), $D_i \in \mathbb{N}^*$ comme la date limite (normalisée) [2], $T_i \in \mathbb{N}^*$ la période (normalisée) et $\rho_i \in \mathbb{N}^*$ la priorité.

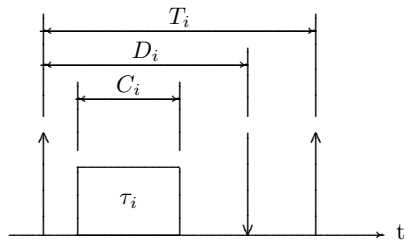


FIGURE 3.1: Définition d'une tâche

On considère Θ comme le plus petit commun multiple (PPCM) des périodes des tâches :

$$\Theta = \operatorname*{lcm}_{i \in \mathcal{I}_n} T_i.$$

Le problème de la modélisation peut être limité par le espace temporel $l \in \mathcal{J}_\Theta$, c'est-à-dire, le temps devient une succession des périodes avec l'amplitude Θ. Grâce à cette simplification, les variables sont déduites au cours du temps avec une translation temporelle.

Finalement, on considère ρ_M comme la priorité plus grande :

$$\rho_M = \max_{i \in \mathcal{I}_n} \rho_i.$$

Partitions

On considère $m \in \mathbb{N}^*$ comme le nombre de partitions P_j ($j \in \mathcal{I}_m$) et on dénote par $\mathcal{P} = \{P_j,\ j \in \mathcal{I}_m\}$ l'ensemble composé de toutes les partitions. Une partition P_j est un ensemble non-vide composé de n_j tâches τ_i. Formellement :

$$\forall j \in \mathcal{I}_m,\ \exists n_j \in \mathcal{I}_n,\ \exists i_1, \ldots, i_{n_j} \in \mathcal{I}_n \text{ tel que } P_j = \{\tau_{i_1}, \ldots, \tau_{i_{n_j}}\}.$$

On note que la condition $P_j \neq \emptyset$ pour tous les $j \in \mathcal{I}_m$ implique $n \geq m$, c'est-à-dire, le nombre de tâches qui sont intégrées ne doit pas être plus petit que le nombre de partitions. On utilisera cette hypothèse lors de cette étude.

De plus, une tâche τ_i est distribuée à une seule partition. Pour cette raison, l'ensemble $\mathcal{P}$ est un sous-ensemble de $\mathcal{T}$, c'est-à-dire :

— $\mathcal{T} = \bigcup_{j \in \mathcal{I}_m} P_j$;
— $j_1 \neq j_2 \Longrightarrow P_{j_1} \cap P_{j_2} = \emptyset$.

2. La date limite d'une tâche τ_i doit être inférieure ou égale à sa période, c'est-à-dire, $D_i \leq T_i$.

On considère $\alpha_{i,j} \in \{0,1\}$ comme une variable binaire qui représente la distribution des tâches entre les partitions. Plus exactement :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \alpha_{i,j} = \begin{cases} 1, & \text{si } \tau_i \in P_j; \\ 0, & \text{si } \tau_i \notin P_j. \end{cases}$$

Instances

Une instance est définie comme la frame temporelle qui est allouée à l'exécution d'une partition donnée P_j. Comme une partition peut être exécutée plusieurs fois, les instances sont représentées par l'indice k. La k-ème instance de la partition P_j est caractérisée par un temps de début $S_{j,k} \in \mathcal{J}_\Theta$ et un temps final $E_{j,k} \in \mathcal{J}_\Theta$ d'exécution, comme la Figure 3.2 représente.

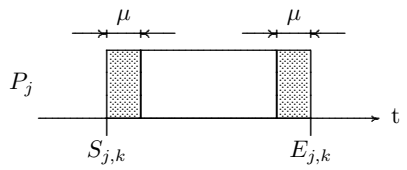


FIGURE 3.2: Définition de la k-ème instance de la partition P_j

Une *instance réelle* est définie comme une instance caractérisée par $S_{j,k} < E_{j,k}$, c'est-à-dire, un temps d'exécution est alloué à la partition P_j sur sa k-ème instance. Autrement, quand $S_{j,k} = E_{j,k}$, la k-ème instance de P_j est nommée comme une *instance virtuelle*. On note que le concept d'instance virtuelle est requise dans la modélisation due au fait qu'on ne sait pas *a priori* le nombre des instances réelles nécessaires pour un problème d'ordonnancement spécifique. En réalité, le nombre des instances réelles d'une partition est directement optimisé par le solveur pour satisfaire les contraintes en même temps que la fonction coût est minimisée.

Dans ce contexte, on considère $uppb \in \mathbb{N}^*$ comme la limite supérieure du nombre des instances de la partition. Cette limite supérieure est requise pour obtenir un nombre fixé de variables dans la modélisation. Pour cette raison, le nombre de variables ne dépend pas du nombre d'instances réelles, mais seulement de la valeur fixée de la limite supérieure $uppb$.

Pour caractériser si la k-ème instance de la partition P_j est réelle ou virtuelle, une variable binaire $\beta_{j,k} \in \{0,1\}$ est introduite de la façon suivante :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \beta_{j,k} = \begin{cases} 1, & \text{si } E_{j,k} - S_{j,k} \geq 2\mu + 1; \\ 0, & \text{si } E_{j,k} - S_{j,k} = 0. \end{cases} \tag{3.1}$$

avec $\mu \in \mathbb{N}$ comme l'overhead de la commutation des partitions (normalisé par Δt), comme la Figure 3.2 montre. On note que cette définition exige que si une instance est réelle, le correspondant temps d'exécution alloué est suffisant pour les temps des overheads (ouverture et fermeture) et au moins une unité de temps pour l'exécution de la tâche.

Formulation du problème

Pour un ensemble de tâches spécifiques τ_i, l'objectif du problème de partage temporel consiste en trouver une configuration $\mathcal{C} = \{(\alpha_{i,j})_{i,j}, (S_{j,k}, E_{j,k})_{j,k}\}$, telle que les contraintes temporelles de toutes les tâches sont satisfaites. Essentiellement, il consiste en assigner chaque tâche τ_i à une partition P_j et ajuster les caractéristiques temporelles des instances de la partition, de cette façon toutes les tâches satisfont leur date limite. La Figure 3.3 illustre un exemple.

Article_taches/figures/Integration.pdf

FIGURE 3.3: Problème d'intégration

3.2.2 Caractérisation des temps d'exécution et de commutation d'une instance

L'ordonnancement final est entièrement décrit par la configuration $\mathcal{C}$. Néanmoins pour formuler les contraintes du problème d'ordonnancement, l'utilisation de variables intermédiaires est requise. On note que ces variables sont entièrement caractérisées par les trois variables principales $\alpha_{i,j}$, $S_{j,k}$ et $E_{j,k}$.

Temps d'ouverture et de fermeture d'une instance

Les variables intermédiaires suivantes sont introduites pour comparer les temps d'ouverture et de fermeture d'une instance. D'abord, $\gamma_{j_2,k_2}^{j_1,k_1}$ caractérise si la k_1-ème instance de la partition P_{j_1} commence après le début de la k_2-ème instance de la partition P_{j_2} :

$$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb},\ \gamma_{j_2,k_2}^{j_1,k_1} = \begin{cases} 1, & \text{si } S_{j_1,k_1} \geq S_{j_2,k_2}; \\ 0, & \text{autrement.} \end{cases} \tag{3.2}$$

Ensuite, $\delta_{j_2,k_2}^{j_1,k_1}$ caractérise si la k_1-ème instance de la partition P_{j_1} commence après la fin de la k_2-ème instance de la partition P_{j_2} :

$$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb},\ \delta_{j_2,k_2}^{j_1,k_1} = \begin{cases} 1, & \text{si } S_{j_1,k_1} \geq E_{j_2,k_2}; \\ 0, & \text{autrement.} \end{cases} \tag{3.3}$$

Inversement, $\delta_{j_2,k_2}^{'j_1,k_1}$ caractérise si la k_2-ème instance de la partition P_{j_2} finit après le début de la k_1-ème instance de la partition P_{j_1} :

$$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb},\ \delta_{j_2,k_2}^{'j_1,k_1} = \begin{cases} 1, & \text{si } E_{j_2,k_2} \geq S_{j_1,k_1}; \\ 0, & \text{autrement.} \end{cases} \tag{3.4}$$

Finalement, $comp_{j_2,k_2}^{j_1,k_1}$ caractérise si la fin de la k_2-ème instance de la partition P_{j_2} coïncide avec le début de la k_1-ème instance de la partition P_{j_1} :

$$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb},\ comp_{j_2,k_2}^{j_1,k_1} = \begin{cases} 1, & \text{si } E_{j_2,k_2} = S_{j_1,k_1}; \\ 0, & \text{autrement.} \end{cases} \tag{3.5}$$

Temps d'exécution d'une instance

Pour compter le temps d'exécution effectif d'une partition par-dessus toutes les périodes, les prochaines variables sont introduites :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^S_{j,k,l} = \begin{cases} 1, & \text{si } S_{j,k} \leq l; \\ 0, & \text{autrement.} \end{cases} \tag{3.6}$$

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^E_{j,k,l} = \begin{cases} 1, & \text{si } E_{j,k} \leq l; \\ 0, & \text{autrement.} \end{cases} \tag{3.7}$$

Alors, il est possible d'introduire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^-_{j,k,l} = \epsilon^S_{j,k,l} - \epsilon^E_{j,k,l},$$

qui caractérise si l'instant l appartient à la k-ème instance de la partition P_j :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^-_{j,k,l} = \begin{cases} 1, & \text{si } S_{j,k} \leq l < E_{j,k}; \\ 0, & \text{autrement.} \end{cases} \tag{3.8}$$

Temps de commutation d'une instance

Pour caractériser le temps de commutation l pour la partition P_j quand la k-ème instance est réelle, les prochaines variables sont introduites :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^+_{j,k,l} = \epsilon^S_{j,k,l} + \epsilon^E_{j,k,l},$$

cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^+_{j,k,l} = \begin{cases} 2, & \text{si } E_{j,k} \leq l; \\ 1, & \text{si } S_{j,k} \leq l < E_{j,k}; \\ 0, & \text{si } l < S_{j,k}. \end{cases} \tag{3.9}$$

Avec la convention $\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \epsilon^+_{j,k,-1} = 0$, on définit :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \zeta_{j,k,l} = \epsilon^+_{j,k,l} - \epsilon^+_{j,k,(l-1)},$$

cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \zeta_{j,k,l} = \begin{cases} 2, & \text{si } S_{j,k} = l \text{ et } E_{j,k} = l; \\ 1, & \text{si } S_{j,k} = l \text{ ou exclusive } E_{j,k} = l; \\ 0, & \text{si } S_{j,k} \neq l \text{ et } E_{j,k} \neq l. \end{cases} \tag{3.10}$$

Les instances virtuelles sont caractérisées par :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \eta_{j,k,l} = \begin{cases} 1, & \text{si } \zeta_{j,k,l} = 2; \\ 0, & \text{autrement.} \end{cases} \tag{3.11}$$

Finalement, c'est possible de caractériser le temps de commutation pour les instances réelles de la façon suivante :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \theta_{j,k,l} = \zeta_{j,k,l} - 2\eta_{j,k,l},$$

cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \theta_{j,k,l} = \begin{cases} 1, & \text{si } S_{j,k} = l \text{ ou exclusive } E_{j,k} = l; \\ 0, & \text{autrement.} \end{cases} \tag{3.12}$$

Temps d'ouverture et de fermeture d'une instance

Pour limiter la formulation, il est nécessaire de caractériser si le temps de commutation d'une instance réelle correspond au temps d'ouverture ou de fermeture de l'instance. En considérant la convention $\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \epsilon^E_{j,k,-1} = 0$, la prochaine variable est introduite de la façon suivante :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \kappa_{j,k,l} = \theta_{j,k,l} - (\epsilon^E_{j,k,l} - \epsilon^E_{j,k,l-1}),$$

cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \kappa_{j,k,l} = \begin{cases} 1, & \text{si } S_{j,k} = l \text{ et } E_{j,k} \neq l; \\ 0, & \text{si } S_{j,k} \neq l; \\ -1, & \text{si } S_{j,k} = E_{j,k} = l. \end{cases} \tag{3.13}$$

Le temps d'ouverture d'une instance réelle est donc caractérisé par :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \lambda_{j,k,l} = \begin{cases} 1, & \text{si } \kappa_{j,k,l} = 1; \\ 0, & \text{autrement.} \end{cases}$$

Cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \lambda_{j,k,l} = \begin{cases} 1, & \text{si } S_{j,k} = l \text{ et } E_{j,k} \neq l; \\ 0, & \text{autrement.} \end{cases} \tag{3.14}$$

Finalement, le temps de fermeture d'une instance réelle est caractérisé de la façon suivante :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \nu_{j,k,l} = \theta_{j,k,l} - \lambda_{j,k,l},$$

cela veut dire :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \nu_{j,k,l} = \begin{cases} 1, & \text{si } E_{j,k} = l \text{ et } S_{j,k} \neq l; \\ 0, & \text{autrement.} \end{cases} \tag{3.15}$$

On note que la variable $\nu_{j,k,l}$ ne peut pas être égale à -1 selon les Définitions (3.12) et (3.14), $\lambda_{j,k,l} = 1 \Rightarrow \theta_{j,k,l} = 1$.

3.2.3 Caractérisation du temps d'exécution de la tâche avec préemption

L'objectif de cette sous-section est d'introduire les variables intermédiaires nécessaires pour formuler les contraintes d'ordonnancement. Pour atteindre cet objectif, l'hypothèse suivante est réalisée.

Hypothèse 1 Deux tâches différentes qui ont la même priorité sont distribuées dans deux partitions différentes :

$$\forall i_1 \neq i_2 \in \mathcal{I}_n,\ \rho_{i_1} = \rho_{i_2} \Rightarrow \forall j \in \mathcal{I}_m,\ \alpha_{i_1,j} + \alpha_{i_2,j} \leq 1.$$

Comme les prochaines sections expriment avec plus de détails, cette hypothèse est nécessaire pour garantir que à n'importe quel instant l, le temps d'exécution est alloué à une seule tâche.

Priorité de la tâche dans une partition

La priorité d'une tâche dans une partition donnée est caractérisée par :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \varrho_{i,j} = \rho_i \alpha_{i,j},$$

avec les interprétations suivantes :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \varrho_{i,j} = \begin{cases} \rho_i, & \text{si } \tau_i \in P_j; \\ 0, & \text{autrement.} \end{cases} \tag{3.16}$$

Cette variable permet d'organiser la tâche par rapport aux priorités des autres tâches qui se trouvent dans la même partition :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall tot \in \mathcal{I}_{\rho_M},\ \upsilon_{i,j}^{tot} = \begin{cases} 1, & \text{si } \varrho_{i,j} = tot; \\ 0, & \text{autrement.} \end{cases} \tag{3.17}$$

On note que avec l'hypothèse 1, pour une partition spécifique P_j et un niveau de priorité tot, il existe au plus une tâche τ_i avec qui $\upsilon_{i,j}^{tot} = 1$.

Temps d'exécution d'une tâche

Le temps d'exécution d'une tâche est une fonction qui dépend des paramètres suivants :

— le temps d'exécution d'une partition ;
— les overheads ;
— la distribution des tâches à l'intérieur des partitions ;
— la priorité relative des tâches qui se trouvent à l'intérieur de la partition ;
— la périodicité de la tâche ;
— le pire temps d'exécution de la tâche ;
— l'historique du temps d'exécution de la tâche.

Particulièrement, le dernier paramètre implique à considérer une stratégie récursive sur l'indice temporel $l \in \mathcal{J}_{\Theta-1}$ du temps d'exécution de la tâche. Dans ce contexte, le temps d'exécution d'une tâche est dénoté par :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \xi_{i,j,l} = \begin{cases} 1, & \text{si le temps d'execution } l \text{ est pour } \tau_i \in P_j; \\ 0, & \text{autrement.} \end{cases} \tag{3.18}$$

L'objectif est de limiter, d'une façon récursive sur l'indice temporel $l \in \mathcal{J}_{\Theta-1}$, la variable $\xi_{i,j,l}$ pour répondre à la Définition (3.18). Pour cette raison, on considère que la variable $\xi_{i,j,l}$ est bien définie jusqu'à l'instant $l-1$. Pour définir $\xi_{i,j,l}$ à l'instant l, on définit d'abord l'historique du temps d'exécution de la tâche τ_i dans la partition P_j jusqu'à l'instant $l-1$ sur son actuel période :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \Phi_{i,j,l} = \sum_{l'=\lfloor l/T_i \rfloor T_i}^{l-1} \xi_{i,j,l'}. \tag{3.19}$$

Pour déterminer si la tâche τ_i a besoin de temps d'exécution additionnel dans la partition P_j sur la période T_i, la variable définie dans (3.19) est comparée avec le pire temps d'exécution

de la tâche C_i :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \Phi^C_{i,j,l} = \begin{cases} 1, & \text{si } \Phi_{i,j,l} < C_i \alpha_{i,j}; \\ 0, & \text{autrement.} \end{cases} \tag{3.20}$$

On note que si l est multiple de T_i, la somme dans (3.19) devient zéro, ce qui indique que $\Phi_{i,j,l} = 0$. Pour cette raison, le compteur sur le temps d'exécution pour la tâche τ_i est réinitialisé à zéro à chaque nouvelle période T_i. Par conséquent, quand une nouvelle période commence, $\Phi^C_{i,j,l} = 1$ ce qui indique que τ_i a besoin de temps d'exécution additionnel dans P_j.

Pour déterminer si l'instant l est réservé à la partition P_j au-delà des temps des overheads, on introduit :

$$\forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \phi_{j,l} = \sum_{k=1}^{uppb} \epsilon^-_{j,k,l} - \mathcal{O}_{j,l}, \tag{3.21}$$

avec $\mathcal{O}_{j,l}$ qui caractérise si l'instant l est réservé à un overhead de la partition P_j :

$$\mathcal{O}_{j,l} = \sum_{k=1}^{uppb} \left[\sum_{l'=l-\mu+1}^{l} \lambda_{j,k,l'} + \sum_{l'=l+1}^{l+\mu} \nu_{j,k,l'} \right].$$

On note que $\phi_{j,l}$ peut être interprété de la façon suivante : $\phi_{j,l} = 1$ si l'instant l est réservé à l'exécution de la partition P_j au-delà des temps des overheads, autrement $\phi_{j,l} = 0$.

Pour caractériser si la tâche, selon sa priorité, a besoin de temps d'exécution et si le temps d'exécution est disponible pour la partition correspondante (au-delà des overheads), la prochaine variable est définie :
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M}$,

$$\chi^{tot}_{i,j,l} = \begin{cases} 1, & \text{si } v^{tot}_{i,j} = 1,\ \Phi^C_{i,j,l} = 1 \text{ et } \phi_{j,l} = 1; \\ 0, & \text{autrement.} \end{cases} \tag{3.22}$$

On note que pour un instant concret l, le non-chevauchement des instances de la partition implique qu'il existe au plus une P_j avec $\phi_{j,l} = 1$. De plus, pour un niveau de priorité spécifique tot, l'Hypothèse 1 garantie que il existe au plus une tâche $\tau_i \in P_j$ telle que $v^{tot}_{i,j} = 1$. Pour cette raison, pour n'importe quel instant l et niveau de priorité tot, il existe au plus une partition P_j et une tâche $\tau_i \in P_j$ telles que $\chi^{tot}_{i,j,l} = 1$.

Dans ce cas, le temps d'exécution doit être alloué à la tâche $\tau_i \in P_j$ avec $\chi^{tot}_{i,j,l} = 1$ et avec le maximum tot. Pour faire cela, on définit la variable intermédiaire $\psi^{tot}_{i,j,l}$ qui devient $\psi^{tot}_{i,j,l} = 1$ pour la dernière configuration et $\psi^{tot}_{i,j,l} \leq 0$ pour d'autres configurations. Cela veut dire :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \psi^{\rho_M}_{i,j,l} = \chi^{\rho_M}_{i,j,l}, \tag{3.23}$$

et

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M-1},\ \psi^{tot}_{i,j,l} = \chi^{tot}_{i,j,l} - \sum_{tot'=tot+1}^{\rho_M} \sum_{i'=1}^{n} \chi^{tot'}_{i',j,l}. \tag{3.24}$$

Maintenant, il reste à filtrer les valeurs de $\chi_{i,j,l}^{tot}$ afin de caractériser la configuration dans laquelle $\chi_{i,j,l}^{tot} = 1$. Le filtrage est réalisé de la façon suivante :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M},\ \omega_{i,j,l}^{tot} = \begin{cases} 1, & \text{si } \psi_{i,j,l}^{tot} = 1; \\ 0, & \text{autrement.} \end{cases} \tag{3.25}$$

Dans ce moment, pour un instant concret l, il existe au plus une tâche τ_i, une partition P_j et un niveau de priorité tot tels que $\omega_{i,j,l}^{tot} = 1$. Cela veut dire que l'instant l est alloué à la tâche $\tau_i \in P_j$. Ce permet de définir récursivement la valeur de $\xi_{i,j,l}$ de la façon suivante :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \xi_{i,j,l} = \sum_{tot=1}^{\rho_M} \omega_{i,j,l}^{tot}. \tag{3.26}$$

On note que si à l'instant l, tous les $\omega_{i,j,l}^{tot}$ sont zéro (le temps d'exécution n'est pas disponible ou demandé), la Définition (3.26) force que tous les $\xi_{i,j,l}$ soient zéro.

Pour initialiser la définition récursive de $\xi_{i,j,l}$, il est suffisant de voir que (3.19) implique que $\Phi_{i,j,0} = 0$, ce qui est compatible avec le modèle à l'instant initial.

3.2.4 Résummé des variables du problème

Les variables du problème sont résumées dans la Table 3.1, en spécifiant la dimension et la valeur. On note que pour des caractéristiques d'une tâche, l'ordonnancement devient entièrement caractérisé par la configuration $\mathcal{C} = \{(\alpha_{i,j})_{i,j}, (E_{j,k}, S_{j,k})_{j,k}\}$. Toutes les autres variables peuvent être interprétées comme des variables intermédiaires qui sont entièrement déterminées par la configuration $\mathcal{C}$. Néanmoins, pour la modélisation et l'optimisation, elles sont traitées comme des variables principales avec un comportement complètement contraint.

3.3 Formulation des contraintes du problème

Cette section introduit les contraintes du modèle. Afin de rendre le processus d'optimisation acceptable à niveau de complexité de calcul, toutes les contraintes sont formulées sur la forme des inégalités ou égalités linéaires. Pour cette raison, le problème peut être résolu en utilisant des algorithmes de programmation linéaires.

Pour clarifier les expressions de cette section, le prochain préliminaire lemme est introduit :

Lemme 3.1. *On considère $a < b \in \mathbb{Z}$. La fonction suivante :*

$$\begin{array}{rccc} f: & \mathcal{I}_{a,b} \times \mathcal{I}_{a,b} & \rightarrow & \{0,1\} \\ & (x,y) & \rightarrow & \begin{cases} 1, & \text{si } x \geq y; \\ 0, & \text{autrement.} \end{cases} \end{array}$$

est entièrement modélisée par la prochaine contrainte linéaire :

$$\frac{x - y + 1/2}{b - a + 1} \leq f(x,y) \leq \frac{x - y}{b - a + 1} + 1.$$

Table 3.1: Variables du problème

Variables	Dimension	Valeur
$\alpha_{i,j}$	$\{0,1\}$	$n \times m$
$S_{j,k}$	$\mathcal{J}_{\Theta}$	$m \times uppb$
$E_{j,k}$	$\mathcal{J}_{\Theta}$	$m \times uppb$
$\beta_{j,k}$	$\{0,1\}$	$m \times uppb$
$\gamma_{j_2,k_2}^{j_1,k_1}$	$\{0,1\}$	$m \times (m-1) \times uppb^2$
$\delta_{j_2,k_2}^{j_1,k_1}$	$\{0,1\}$	$m \times (m-1) \times uppb^2$
$\delta'^{j_1,k_1}_{j_2,k_2}$	$\{0,1\}$	$m \times (m-1) \times uppb^2$
$comp_{j_2,k_2}^{j_1,k_1}$	$\{0,1\}$	$m \times (m-1) \times uppb^2$
$\epsilon_{j,k,l}^{S}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\epsilon_{j,k,l}^{E}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\epsilon_{j,k,l}^{-}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\epsilon_{j,k,l}^{+}$	$\mathcal{I}_{0,2}$	$m \times uppb \times (\Theta+1)$
$\zeta_{j,k,l}$	$\mathcal{I}_{0,2}$	$m \times uppb \times (\Theta+1)$
$\eta_{j,k,l}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\theta_{j,k,l}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\kappa_{j,k,l}$	$\mathcal{I}_{-1,1}$	$m \times uppb \times (\Theta+1)$
$\lambda_{j,k,l}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\nu_{j,k,l}$	$\{0,1\}$	$m \times uppb \times (\Theta+1)$
$\varrho_{i,j}$	$\mathcal{J}_{\rho_M}$	$n \times m$
$\upsilon_{i,j}^{tot,+}$	$\{0,1\}$	$n \times m \times \rho_M$
$\upsilon_{i,j}^{tot,-}$	$\{0,1\}$	$n \times m \times \rho_M$
$\upsilon_{i,j}^{tot}$	$\{0,1\}$	$n \times m \times \rho_M$
$\xi_{i,j,l}$	$\{0,1\}$	$n \times m \times \Theta$
$\Phi_{i,j,l}$	$\mathcal{J}_{\Theta-1}$	$n \times m \times \Theta$
$\Phi_{i,j,l}^{C}$	$\{0,1\}$	$n \times m \times \Theta$
$\phi_{j,l}$	$\{0,1\}$	$m \times \Theta$
$\chi_{i,j,l}^{tot}$	$\{0,1\}$	$n \times m \times \Theta \times \rho_M$
$\psi_{i,j,l}^{tot}$	$\mathcal{I}_{1-\rho_M,1}$	$n \times m \times \Theta \times \rho_M$
$\omega_{i,j,l}^{tot}$	$\{0,1\}$	$n \times m \times \Theta \times \rho_M$

Démonstration. L'objectif est caractérisé selon $x, y \in \mathcal{I}_{a,b}$ (avec $c \in \mathbb{Z}$) dans la prochaine inégalité :

$$\frac{x-y+1/2}{b-a+1} \leq c \leq \frac{x-y}{b-a+1} + 1. \tag{3.27}$$

Par exemple, dans le cas que $x - y \geq 0$, cela implique que :

$$0 < \frac{1}{2(b-a+1)} \leq \frac{x-y+1/2}{b-a+1} \leq \frac{b-a+1/2}{b-a+1} < 1,$$

et

$$1 \leq \frac{x-y}{b-a+1} + 1 \leq \frac{b-a}{b-a+1} + 1 < 2.$$

Pour cette raison, l'inégalité (3.27) devient possible si et seulement si $c = 1$.

Par contre, dans le cas que $x - y < 0$, l'inégalité devient :

$$-1 < \frac{a-b+1/2}{b-a+1} \leq \frac{x-y+1/2}{b-a+1} \leq \frac{-1/2}{b-a+1} < 0,$$

et

$$0 < \frac{1}{b-a+1} = \frac{a-b}{b-a+1} + 1 \leq \frac{x-y}{b-a+1} + 1 < 1.$$

Pour cette raison, l'inégalité (3.27) devient possible si et seulement si $c = 0$. □

3.3.1 Distribution des tâches entre partitions

Cette sous-section décrit les contraintes appliquées pour la distribution des tâches en utilisant la variable $\alpha_{i,j}$.

— Chaque tâche est répartie dans seulement une partition. Formellement, pour tous $j_1 \neq j_2 \in \mathcal{I}_m$, $P_{j_1} \cap P_{j_2} = \emptyset$:

$$\forall i \in \mathcal{I}_n,\ \sum_{j=1}^{m} \alpha_{i,j} = 1. \tag{3.28}$$

— Il n'y a pas des partitions vides. Formellement, pour tous $j \in \mathcal{I}_m$, $P_j \neq \emptyset$:

$$\forall j \in \mathcal{I}_m,\ \sum_{i=1}^{n} \alpha_{i,j} \geq 1. \tag{3.29}$$

— Afin d'éviter des configurations équivalentes qui sont redondantes dans le processus de distribution, les prochaines contraintes sont imposées :

$$\alpha_{1,1} = 1, \qquad \forall j \in \mathcal{I}_{2,m},\ \alpha_{1,j} = 0. \tag{3.30}$$

$$\forall i \in \mathcal{I}_{2,n},\ \forall j \in \mathcal{I}_{2,m},\ \begin{cases} \alpha_{i,j} \leq \displaystyle\sum_{i'=1}^{i-1} \alpha_{i',j-1}, \text{si } j \leq i; \\ \alpha_{i,j} = 0, \text{si } j > i. \end{cases} \tag{3.31}$$

Dans cette réglage, la tâche τ_1 est intégrée dans la partition P_1 ($\tau_1 \in P_1$) et les autres tâches sont intégrées successivement avec l'indice $i \in \mathcal{I}_{2,n}$. Essentiellement, si les tâches $\tau_2, \ldots, \tau_{i-1}$ ont été intégrées, la tâche τ_i peut être intégrée dans la partition P_j ($\tau_i \in P_j$), seulement si pour tous $j' \in \mathcal{I}_{j-1}$, $P_{j'} \neq \emptyset$.

— Si deux tâches τ_{i_1} et τ_{i_2}, avec $i_1 \neq i_2$, doivent être placées dans deux partitions différentes, alors

$$\forall j \in \mathcal{I}_m,\ \alpha_{i_1,j} + \alpha_{i_2,j} \leq 1. \tag{3.32}$$

— Si deux tâches τ_{i_1} et τ_{i_2}, avec $i_1 \neq i_2$, doivent être placées dans la même partition, alors

$$\forall j \in \mathcal{I}_m,\ \alpha_{i_1,j} = \alpha_{i_2,j}. \tag{3.33}$$

On note que pour une distribution fixée *a priori*, $\alpha_{i,j}$ devient une constante.

3.3.2 Contraintes des partitions

Cette sous-section introduit les contraintes liées au comportement des instances des partitions. Elle spécifie aussi une importante partie des variables intermédiaires requises pour formuler les contraintes d'ordonnancement dans les sous-sections suivantes.

Temps de début et final des instances

Les contraintes suivantes sont imposées sur le temps de début et final des instances d'une partition.

— Les temps de début et final d'une instance sont dans le domaine du temps :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ 0 \leq S_{j,k}, E_{j,k} \leq \Theta. \tag{3.34}$$

— Le temps de début d'une instance se trouve avant de son temps final. De plus, si une instance est réelle ($\beta_{j,k} = 1$), le correspondant temps d'exécution alloué doit être suffisant pour au moins comprendre les deux overheads et une unité de temps, spécifié dans (3.1) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ (2\mu + 1)\beta_{j,k} \leq E_{j,k} - S_{j,k} \leq \Theta\beta_{j,k}. \tag{3.35}$$

— Pour une partition donnée P_j, les instances réelles sont placées au début dans l'ordre de numération de k :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb-1},\ \beta_{j,k+1} \leq \beta_{j,k}. \tag{3.36}$$

Par conséquent, on a que si $\beta_{j,k} = 0 \Rightarrow \beta_{j,k} = \beta_{j,k+1} = \ldots = \beta_{j,uppb} = 0$. Cela veut dire que si la k-ème instance est virtuelle, les k, $(k+1)$, ... , *uppb*-ème instances sont aussi virtuelles. Par opposition, si la k-ème instance est réelle, alors 1, 2, ... , k-ème instances sont réelles aussi.

— Le temps de début d'une instance virtuelle est contraint par :

$$\forall j \in \mathcal{I}_m,\ 0 \leq S_{j,1} \leq \Theta\beta_{j,1}, \tag{3.37}$$

et

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb-1},\ 0 \leq S_{j,k+1} - E_{j,k} \leq \Theta\beta_{j,k+1}. \tag{3.38}$$

Cela veut dire que si la $(k+1)$-ème instance est virtuelle ($\beta_{j,k+1} = 0$), on a que $S_{j,k+1} = E_{j,k+1} = E_{j,k}$, comme la Contrainte indique (3.35). De plus, (3.35) et (3.38) imposent que $S_{j,1} \leq E_{j,1} \leq S_{j,2} \leq \ldots \leq E_{j,uppb-1} \leq S_{j,uppb} \leq E_{j,uppb}$.

— Le temps entre deux instances réelles de la même partition P_j est suffisant pour permettre l'exécution d'une autre partition, c'est-à-dire, en considérant (au moins) les deux overheads et une unité de temps :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb-1},\ (2\mu+1)\beta_{j,k+1} \leq S_{j,k+1} - E_{j,k}. \tag{3.39}$$

— Deux instances réelles n'ont pas de chevauchement. D'abord, (3.2) et (3.3) sont accomplis à travers de la prochaine contrainte :
$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb}$,

$$\frac{S_{j_1,k_1} - S_{j_2,k_2} + 1/2}{\Theta+1} \leq \gamma_{j_2,k_2}^{j_1,k_1} \leq \frac{S_{j_1,k_1} - S_{j_2,k_2}}{\Theta+1} + 1, \tag{3.40}$$

$$\frac{S_{j_1,k_1} - E_{j_2,k_2} + 1/2}{\Theta+1} \leq \delta_{j_2,k_2}^{j_1,k_1} \leq \frac{S_{j_1,k_1} - E_{j_2,k_2}}{\Theta+1} + 1. \tag{3.41}$$

Ensuite, la contrainte de non-chevauchement est formalisée de la façon suivante :

$$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb},\ \gamma_{j_2,k_2}^{j_1,k_1} \leq \delta_{j_2,k_2}^{j_1,k_1} + (2 - \beta_{j_1,k_1} - \beta_{j_2,k_2}). \tag{3.42}$$

Cela implique que si $S_{j_2,k_2} \leq S_{j_1,k_1} \Longrightarrow E_{j_2,k_2} \leq S_{j_1,k_1}$ tant que les deux instances sont réelles, parce que dans ce cas $2 - \beta_{j_1,k_1} - \beta_{j_2,k_2} = 0$.
On note que si au moins une des deux instances considérées est virtuelle, la contrainte (3.42) est relaxée quand $2 - \beta_{j_1,k_1} - \beta_{j_2,k_2} \geq 1$.

— Le temps entre deux instances de différentes partitions est zéro ou suffisant pour l'exécution d'une autre partition, c'est-à-dire, au moins les deux overheads et une unité de temps. D'abord, (3.4) et (3.5) sont accomplis par :
$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb}$,

$$\frac{E_{j_2,k_2} - S_{j_1,k_1} + 1/2}{\Theta+1} \leq \delta'^{j_1,k_1}_{j_2,k_2} \leq \frac{E_{j_2,k_2} - S_{j_1,k_1}}{\Theta+1} + 1, \tag{3.43}$$

$$\frac{\delta_{j_2,k_2}^{j_1,k_1} + \delta'^{j_1,k_1}_{j_2,k_2} - 3/2}{2} \leq comp_{j_2,k_2}^{j_1,k_1} \leq \frac{\delta_{j_2,k_2}^{j_1,k_1} + \delta'^{j_1,k_1}_{j_2,k_2} - 2}{2} + 1. \tag{3.44}$$

Ensuite, la contrainte est formulée de la façon suivante :
$\forall j_1 \neq j_2 \in \mathcal{I}_m,\ \forall k_1, k_2 \in \mathcal{I}_{uppb}$,

$$(2\mu+1)(1 - comp_{j_2,k_2}^{j_1,k_1}) - (2\mu+1+\Theta)\delta'^{j_1,k_1}_{j_2,k_2} \leq S_{j_1,k_1} - E_{j_2,k_2}. \tag{3.45}$$

Si $E_{j_2,k_2} \leq S_{j_1,k_1}$, deux possibilités existent : $E_{j_2,k_2} = S_{j_1,k_1}$ ou bien $S_{j_1,k_1} - E_{j_2,k_2} \geq 2\mu + 1$. On note que si $E_{j_2,k_2} > S_{j_1,k_1}$, la contrainte es relaxée de la façon suivante : $-\Theta \leq S_{j_1,k_1} - E_{j_2,k_2}$, ce qui est trivial depuis (3.34).

Temps d'exécution et de commutation d'une instance

Les temps d'exécution et de commutation d'une instance sont identifiés de la façon suivante.

— Instant de début d'une instance (3.6) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \frac{l - S_{j,k} + 1/2}{\Theta+1} \leq \epsilon_{j,k,l}^S \leq \frac{l - S_{j,k}}{\Theta+1} + 1. \tag{3.46}$$

— Instant final d'une instance (3.7) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \frac{l - E_{j,k} + 1/2}{\Theta + 1} \leq \epsilon^E_{j,k,l} \leq \frac{l - E_{j,k}}{\Theta + 1} + 1. \tag{3.47}$$

— Temps d'exécution d'une instance (3.8) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^-_{j,k,l} = \epsilon^S_{j,k,l} - \epsilon^E_{j,k,l}. \tag{3.48}$$

— Temps de début et final d'une instance (3.9) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \epsilon^+_{j,k,l} = \epsilon^S_{j,k,l} + \epsilon^E_{j,k,l}. \tag{3.49}$$

Avec la convention :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \epsilon^+_{j,k,-1} = 0. \tag{3.50}$$

La prochaine variable est introduite pour caractériser les temps de commutation des instances réelles et virtuelles (3.10) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \zeta_{j,k,l} = \epsilon^+_{j,k,l} - \epsilon^+_{j,k,(l-1)}. \tag{3.51}$$

— Temps de commutation d'une instance virtuelle (3.11) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \frac{\zeta_{j,k,l} - 3/2}{2} \leq \eta_{j,k,l} \leq \frac{\zeta_{j,k,l} - 2}{2} + 1. \tag{3.52}$$

— Temps de commutation d'une instance réelle (3.12) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \theta_{j,k,l} = \zeta_{j,k,l} - 2\eta_{j,k,l}. \tag{3.53}$$

— Temps de commutation d'une instance par rapport à sa fin (3.13) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \kappa_{j,k,l} = \theta_{j,k,l} - (\epsilon^E_{j,k,l} - \epsilon^E_{j,k,l-1}), \tag{3.54}$$

avec la convention :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \epsilon^E_{j,k,-1} = 0. \tag{3.55}$$

— Temps de début d'une instance réelle (3.14) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \frac{\kappa_{j,k,l} - 1/2}{2} \leq \lambda_{j,k,l} \leq \frac{\kappa_{j,k,l} - 1}{2} + 1. \tag{3.56}$$

— Forcement de le début d'une instance à l'instant initial $l = 0$:

$$\sum_{j=1}^{m} \sum_{k=1}^{uppb} \lambda_{j,k,0} = 1. \tag{3.57}$$

— En évitant que deux instances réelles commencent simultanément :

$$\forall l \in \mathcal{I}_\Theta,\ \sum_{j=1}^{m} \sum_{k=1}^{uppb} \lambda_{j,k,l} \leq 1. \tag{3.58}$$

— Temps final des instances réelles (3.15) :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \forall l \in \mathcal{J}_\Theta,\ \nu_{j,k,l} = \theta_{j,k,l} - \lambda_{j,k,l}. \tag{3.59}$$

3.3.3 Contraintes d'ordonnancement

Cette sous-section introduit les contraintes liées à l'ordonnancement du système partitionné.

Contraintes liées à la priorité de la tâche

Les contraintes utilisées pour caractériser le niveau de priorité d'une tâche dans une partition sont introduites ensuite. On note que si la distribution des tâches dans une partition est connue *a priori*, les prochaines contraintes sont appliquées sur des constantes ; Cela veut dire qu'elles peuvent être calculées *a priori*. Concrètement, la priorité de la tâche est contrainte de la façon suivante.

— Priorités des tâches qui se trouvent dans la partition (3.16) :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \varrho_{i,j} = \rho_i \alpha_{i,j}. \tag{3.60}$$

— Tâches organisées selon le niveau de priorité (3.17) :
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall tot \in \mathcal{I}_{\rho_M}$,

$$\frac{\varrho_{i,j} - tot + 1/2}{\rho_M + 1} \leq \upsilon_{i,j}^{tot,+} \leq \frac{\varrho_{i,j} - tot}{\rho_M + 1} + 1, \tag{3.61}$$

$$\frac{tot - \varrho_{i,j} + 1/2}{\rho_M + 1} \leq \upsilon_{i,j}^{tot,-} \leq \frac{tot - \varrho_{i,j}}{\rho_M + 1} + 1, \tag{3.62}$$

$$\frac{\upsilon_{i,j}^{tot,+} + \upsilon_{i,j}^{tot,-} - 3/2}{2} \leq \upsilon_{i,j}^{tot} \leq \frac{\upsilon_{i,j}^{tot,+} + \upsilon_{i,j}^{tot,-} - 2}{2} + 1. \tag{3.63}$$

Contraintes liées au temps d'exécution d'une tâche

Cet ensemble de contraintes est défini dans deux parties. La première partie détaille les contraintes non-récursives, pendant que la deuxième partie détaille les contraintes définies récursivement au cours du temps l.

Partie non-récursive :

— Le temps d'exécution disponible pour une partition donnée P_j (sans overheads) est défini à partir de (3.21) :

$$\forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \phi_{j,l} = \sum_{k=1}^{uppb} \epsilon^{-}_{j,k,l} - \mathcal{O}_{j,l}, \tag{3.64}$$

avec $\mathcal{O}_{j,l}$ qui caractérise si l'instant l est réservé à un overhead de la partition P_j :

$$\mathcal{O}_{j,l} = \sum_{k=1}^{uppb} \left[\sum_{l'=l-\mu+1}^{l} \lambda_{j,k,l'} + \sum_{l'=l+1}^{l+\mu} \nu_{j,k,l'} \right],$$

avec la prochaine convention :

$$\forall j \in \mathcal{I}_m,\ \forall k \in \mathcal{I}_{uppb},\ \begin{cases} \forall l \in \mathcal{I}_{\Theta+1,\Theta+\mu-1},\ \nu_{j,k,l} = 0; \\ \forall l \in \mathcal{I}_{-\mu+1,-1},\ \lambda_{j,k,l} = 0. \end{cases} \tag{3.65}$$

Partie récursive :

Les prochaines équations sont définies récursivement avec en augmentant l'indice temporel l à travers de $\mathcal{J}_{\Theta-1}$.

— L'historique de la tâche τ_i dans la partition P_j est défini à partir de l'instant $l-1$ sur l'actuelle période T_i. Grâce à (3.19) et (3.20), il est connu si cette tâche a besoin de temps d'exécution additionnel sur la période actuelle :
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},$

$$\Phi_{i,j,l} = \sum_{l'=\lfloor l/T_i \rfloor T_i}^{l-1} \xi_{i,j,l'}, \tag{3.66}$$

$$\frac{C_i\alpha_{i,j} - \Phi_{i,j,l}}{\Theta+1} \leq \Phi^C_{i,j,l} \leq \frac{C_i\alpha_{i,j} - \Phi_{i,j,l} - 1/2}{\Theta+1} + 1. \tag{3.67}$$

— Grâce à (3.22), si une tâche (en fonction de son niveau de priorité) a besoin de temps d'exécution et il est disponible pour la partition (sans overheads) est caractérisé par :
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M},$

$$\frac{v^{tot}_{i,j} + \Phi^C_{i,j,l} + \phi_{j,l} - 5/2}{3} \leq \chi^{tot}_{i,j,l} \leq \frac{v^{tot}_{i,j} + \Phi^C_{i,j,l} + \phi_{j,l} - 3}{3} + 1. \tag{3.68}$$

— Grâce à (3.23) et (3.24) :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \psi^{\rho_M}_{i,j,l} = \chi^{\rho_M}_{i,j,l}, \tag{3.69}$$

et
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M-1},$

$$\psi^{tot}_{i,j,l} = \chi^{tot}_{i,j,l} - \sum_{tot'=tot+1}^{\rho_M} \sum_{i'=1}^{n} \chi^{tot'}_{i',j,l}. \tag{3.70}$$

— Grâce à (3.25), $\psi^{tot}_{i,j,l} = 1$ est caractérisée par :
$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \forall tot \in \mathcal{I}_{\rho_M},$

$$\frac{\psi^{tot}_{i,j,l} - 1/2}{\rho_M+1} \leq \omega^{tot}_{i,j,l} \leq \frac{\psi^{tot}_{i,j,l} - 1}{\rho_M+1} + 1. \tag{3.71}$$

— Finalement, le temps d'exécution à l'instant l est caractérisé par (3.26) :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \xi_{i,j,l} = \sum_{tot=1}^{\rho_M} \omega^{tot}_{i,j,l}. \tag{3.72}$$

Contraintes liées à la date limite

Afin de garantir que chaque tâche τ_i dans une partition P_j satisfait sa date limite D_i à chaque période T_i ($D_i \leq T_i$), la prochaine contrainte est considérée :

$$\forall i \in \mathcal{I}_n,\ \forall j \in \mathcal{I}_m,\ \forall q \in \mathcal{J}_{\Theta/T_i-1},\ \sum_{l=qT_i}^{qT_i+D_i-1} \xi_{i,j,l} = C_i\alpha_{i,j}. \tag{3.73}$$

On note que si $\tau_i \notin P_j$, $\xi_{i,j,l} = 0$ pour toute l, c'est compatible avec la contrainte (3.73) parce que dans ce cas $\alpha_{i,j} = 0$. Par conséquent, la contrainte (3.73) est active seulement quand $\tau_i \in P_j$, cela veut dire que $\alpha_{i,j} = 1$.

Contraintes liées à l'ordre d'exécution d'une tâche

On considère τ_{i_1} et τ_{i_2} comme deux tâches qui ont la même périodicité ($T_{i_1} = T_{i_2}$) placées dans deux partitions différentes P_{j_1} et P_{j_2} respectivement. Si τ_{i_1} doit être exécutée avant que τ_{i_2}, sans fixer *a priori* j_1 et j_2, la contrainte suivante sont ajoutée :

$$\forall j \in \mathcal{I}_m,\ \forall l \in \mathcal{J}_{\Theta-1},\ \xi_{i_2,j,l} \leq 1 - \sum_{j'=1}^{m} \Phi^C_{i_1,j',l}. \tag{3.74}$$

Si la distribution des tâches est fixée *a priori* (j_1 et j_2 sont connus), (3.74) devient :

$$\forall l \in \mathcal{J}_{\Theta-1},\ \xi_{i_2,j_2,l} \leq 1 - \Phi^C_{i_1,j_1,l}. \tag{3.75}$$

3.4 Formulation des objectifs pour l'optimisation

Dans le problème qu'on considère, différentes fonctions objectives pour l'optimisation peuvent être définies. Selon la fonction utilisée la performance du processus d'optimisation change. Due au fait que ce modèle est flexible, quelques exemples de critères sont présentées dans la suite.

3.4.1 Minimisation du temps alloué aux partitions

Un critère possible est la minimisation du temps alloué aux partitions :

Critère 3.2.

$$\min \sum_{j=1}^{m} \sum_{k=1}^{uppb} (E_{j,k} - S_{j,k})$$

Le Critère 3.2 peut être étendu afin de garantir que la solution minimise le temps alloué aux partitions et le temps de réponse global pour toutes les tâches :

Critère 3.3.

$$\min \ (n\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} (E_{j,k} - S_{j,k}) + \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi^C_{i,j,l}$$

Proposition 3.4. *Si une configuration est une solution du problème avec le Critère 3.3, alors elle est solution aussi du problème avec le Critère 3.2.*

Démonstration. On note que si $\tau_i \notin P_j$ ($\alpha_{i,j} = 0$), alors $\Phi^C_{i,j,l} = 0$ par Définition (3.20). Comme la contrainte (3.28) exprime, pour chaque tâche τ_i, il existe une unique partition $P_{j(i)}$ telle que $\alpha_{i,j(i)} = 1$. Par conséquent, pour tout $j' \neq j(i)$, $\Phi^C_{i,j',l} = 0$. Ensuite on a que pour tout $l \in \mathcal{J}_{\Theta-1}$:

$$0 \leq \sum_{i=1}^{n} \sum_{j=1}^{m} \Phi^C_{i,j,l} = \sum_{i=1}^{n} \Phi^C_{i,j(i),l} \leq n,$$

Cela devient :

$$0 \leq \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi_{i,j,l}^{C} \leq n\Theta.$$

On assume que la configuration considérée $\mathcal{C} = \{(\alpha_{i,j})_{i,j}, (E_{j,k}, S_{j,k})_{j,k}\}$ est une solution du problème avec le Critère 3.3, mais elle n'est pas une solution du problème avec le critère 3.2. Cela indique qu'il existe une configuration $\mathcal{C}' = \left\{(\alpha'_{i,j})_{i,j}, (E'_{j,k}, S'_{j,k})_{j,k}\right\}$ telle que :

$$\sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E'_{j,k} - S'_{j,k}\right) \leq \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E_{j,k} - S_{j,k}\right) - 1,$$

ce qui devient :

$$\begin{aligned}
&(n\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E'_{j,k} - S'_{j,k}\right) + \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi_{i,j,l}^{'C} \\
&\leq (n\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E_{j,k} - S_{j,k}\right) - (n\Theta + 1) + n\Theta \\
&\leq (n\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E_{j,k} - S_{j,k}\right) - 1 \\
&\leq (n\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E_{j,k} - S_{j,k}\right) + \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi_{i,j,l}^{C} - 1.
\end{aligned}$$

Ce résultat contredit la supposition de que $\mathcal{C}$ est une solution du problème avec le Critère 3.3. Pour cette raison, $\mathcal{C}$ est aussi solution du problème avec le Critère 3.2. □

On note que le critère 3.3 peut être spécifié afin de minimiser le temps de réponse pour un sous-ensemble de tâche concret τ_i, $i \in \mathcal{I} \subset \{1, \ldots, n\}$. Cela devient la prochaine formulation :

Critère 3.5.

$$\min \quad (\mathrm{Card}(\mathcal{I})\Theta + 1) \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left(E_{j,k} - S_{j,k}\right) + \sum_{i \in \mathcal{I}} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi_{i,j,l}^{C}$$

Proposition 3.6. *Si une configuration est solution du problème avec le Critère 3.5, alors elle est aussi solution du problème avec le Critère 3.2.*

Démonstration. La démonstration devient identique à celle exprimée dans la Proposition 3.4. □

3.4.2 Minimisation du temps de réponse

Un autre critère possible est la minimisation du temps de réponse global pour toutes les tâches :

Critère 3.7.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}$$

Le problème avec le Critère 3.7 est que la solution n'est pas contrainte pour minimiser le temps alloué aux partitions. Particulièrement, il peut exister une solution qui contient des instances réelles sans aucune exécution d'une tâche. Ce phénomène peut être corrigé en considérant la formulation suivante :

Critère 3.8.

$$\min \ (\Theta+1)\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}+\sum_{j=1}^{m}\sum_{k=1}^{uppb}(E_{j,k}-S_{j,k})$$

Proposition 3.9. *Si une configuration est solution du problème avec le Critère 3.8, alors elle est solution aussi du problème avec le Critère 3.7.*

Démonstration. En se basant sur les contraintes de non-chevauchement (3.35), (3.38) et (3.42), la prochaine inégalité est définie :

$$\sum_{j=1}^{m}\sum_{k=1}^{uppb}(E_{j,k}-S_{j,k})\leq\Theta.$$

La suite de la démonstration devient similaire aux arguments présentés dans la démonstration de la Proposition 3.4. □

Encore, le critère peut être étendu afin de minimiser le temps de réponse d'un sous-ensemble concret de tâches τ_i, $i \in \mathcal{I} \subset \{1,\ldots,n\}$. On revient sur le Critère 3.7 pour considérer :

Critère 3.10.

$$\min \sum_{i\in\mathcal{I}}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}$$

et par rapport au Critère 3.8 :

Critère 3.11.

$$\min \ (\Theta+1)\sum_{i\in\mathcal{I}}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}+\sum_{j=1}^{m}\sum_{k=1}^{uppb}(E_{j,k}-S_{j,k})$$

Proposition 3.12. *Si une configuration est solution du problème avec le Critère 3.11, alors elle est aussi solution du problème avec le Critère 3.10.*

Démonstration. La démonstration est identique à celle présentée dans la Proposition 3.9. □

L'inconvénient principal du Critère 3.11 est que l'objectif n'est pas défini pour les tâches τ_i, $i \in \overline{\mathcal{I}}$. Ce problème peut être spécifié afin de minimiser le temps de réponse des tâches τ_i, $i \in \mathcal{I} \subset \{1, \ldots, n\}$, pendant qu'une minimisation du temps alloué aux partitions est réalisé au début. De plus, c'est possible de garantir la minimisation du temps de réponse sur les tâches τ_i, $i \in \overline{\mathcal{I}}$, comme l'expression suivante indique :

Critère 3.13.

$$\min \ \left(\mathrm{Card}(\overline{\mathcal{I}})\Theta + 1\right) \left[(\Theta+1) \sum_{i \in \mathcal{I}} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi^C_{i,j,l} + \sum_{j=1}^{m} \sum_{k=1}^{uppb} (E_{j,k} - S_{j,k}) \right] + \sum_{i \in \overline{\mathcal{I}}} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi^C_{i,j,l}$$

Proposition 3.14. *Si une configuration est solution du problème avec le Critère 3.13, alors elle est aussi solution du problème avec le Critère 3.10 et 3.11.*

Démonstration. Avec

$$\sum_{i \in \overline{\mathcal{I}}} \sum_{j=1}^{m} \sum_{l=0}^{\Theta-1} \Phi^C_{i,j,l} \leq \mathrm{Card}(\overline{\mathcal{I}})\Theta,$$

la démonstration qu'une solution du problème avec le Critère 3.13 est aussi solution avec le Critère 3.11 est similaire à la démonstration de la Proposition 3.4. Alors, elle est aussi solution avec le Critère 3.10 grâce à la Proposition 3.12. □

3.4.3 Fonctions objectives quand $\mu = 0$

Pour une configuration avec des overheads négligeables ($\mu = 0$), les formulations précédentes peuvent donner des solutions avec un grand nombre d'instances. Afin de pénaliser ce type de solutions, on peut introduire un critère basé sur une pondération d'importance :

$$\sum_{j=1}^{m} \sum_{k=1}^{uppb} p_{j,k} \beta_{j,k},$$

avec $(p_{j,k})_{j \in \mathcal{I}_m, k \in \mathcal{I}_{uppb}} \in \mathbb{N}^{m \times uppb}$ comme les coefficients du poids. Entre d'autres possibilités, on pourrait choisir :

— une pénalisation uniforme :

$$\forall j \in \mathcal{I}_m, \forall k \in \mathcal{I}_{uppb}, \ p_{j,k} = 1;$$

— une pénalisation progressive par rapport au nombre de l'instance :

$$\forall j \in \mathcal{I}_m, \forall k \in \mathcal{I}_{uppb}, \ p_{j,k} = k.$$

Par exemple, en considérant la minimisation du temps de réponse global, on pourrait modifier le Critère 3.2 de la façon suivante :

Critère 3.15.

$$\min \ \sum_{j=1}^{m} \sum_{k=1}^{uppb} \left[q \left(E_{j,k} - S_{j,k} \right) + r p_{j,k} \beta_{j,k} \right]$$

avec $q, r \in \mathbb{N}^*$ comme les coefficients du poids. Des modifications similaires peuvent être appliquées sur les autres critères.

3.4.4 Complexité du problème

Le problème est composé de

$$\begin{aligned} N_{\text{var}} =& m(13 \times uppb + 2n + \Theta) \\ &+ m(10 \times uppb \times \Theta + 3n\rho_M + 3n\Theta - 4uppb^2) \\ &+ m(4m \times uppb^2 + 3n\Theta\rho_M) \end{aligned}$$

variables listées dans la Sous-section 3.2.4, et de :

$$\begin{aligned} N_{\text{const}} =& n + \Theta - 2m \\ &+ m(21 \times uppb + 2n + \Theta) \\ &+ m\left(13 \times uppb \times \Theta + 6n\rho_M + 4n\Theta + n\sum_{i=1}^{n}\frac{\Theta}{T_i} - 10 \times uppb^2\right) \\ &+ m \times (10m \times uppb^2 + 5n\Theta\rho_M) \end{aligned}$$

contraintes listées dans les Sous-sections 3.3.1, 3.3.2, et 3.3.3 hors de (3.32), (3.33), (3.73), (3.74) et (3.75). On note que pour des applications pratiques, Θ a un ordre de magnitude plus grand que m, n, $uppb$ et ρ_M. Pour cette raison, la complexité asymptotique du modèle est de $\mathcal{O}(m \times n \times \Theta \times \rho_M)$ variables avec $\mathcal{O}(m \times n \times \Theta \times \rho_M)$ contraintes.

3.5 Étude de cas

Dans cette étude de cas, on considère le problème d'ordonnancement avec la configuration de $n = 10$ tâches, $m = 3$ partitions, une limite supérieure sur le nombre d'instances de $uppb = 10$ et des overheads à $\mu = 1$. Les caractéristiques des tâches sont présentées dans la Table 3.2, celles qui imposent que $\Theta = 60$. Finalement, il est requis que τ_2 soit exécutée avant τ_1, τ_6 avant τ_3 et τ_{10} avant τ_9.

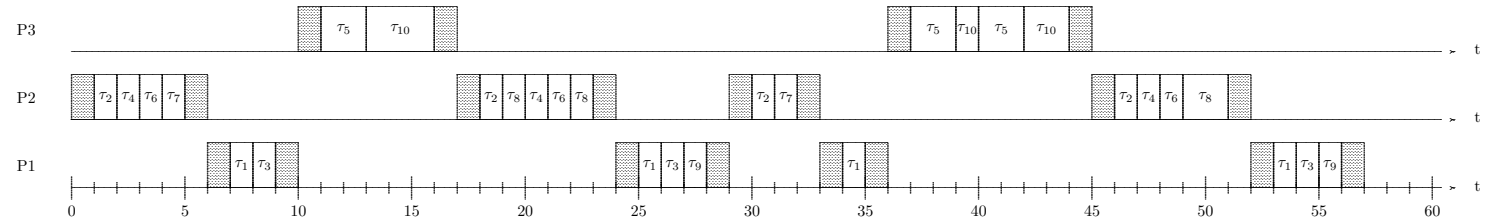
Table 3.2: Paramètres des tâches

Tâche	C_i	D_i	T_i	ρ_i
τ_1	1	12	15	10
τ_2	1	15	15	9
τ_3	1	15	20	8
τ_4	1	20	20	7
τ_5	2	20	20	6
τ_6	1	20	20	5
τ_7	1	20	30	4
τ_8	2	25	30	3
τ_9	1	30	30	2
τ_{10}	3	30	30	1

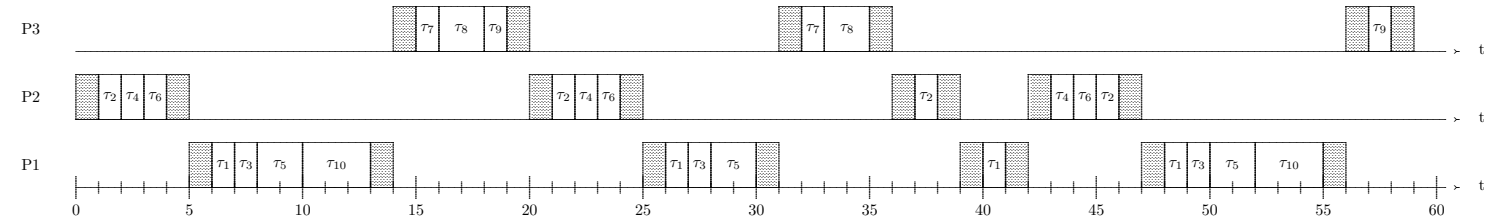
L'optimisation a été réalisée dans un PC avec 2.40GHz quad-core intel i7-3630QM, 16.00GB RAM et `IBM ILOG CPLEX Optimization Studio` sur la version 12.5.1.0. Dans cette étude de cas, les trois problèmes suivants ont été considérés :

— La minimisation du temps alloué aux partitions grâce au Critère 3.3 ;
— La minimisation du temps de réponse global grâce au Critère 3.8 ;
— La minimisation du temps de réponse des tâches τ_1 et τ_{10} grâce au Critère 3.13.

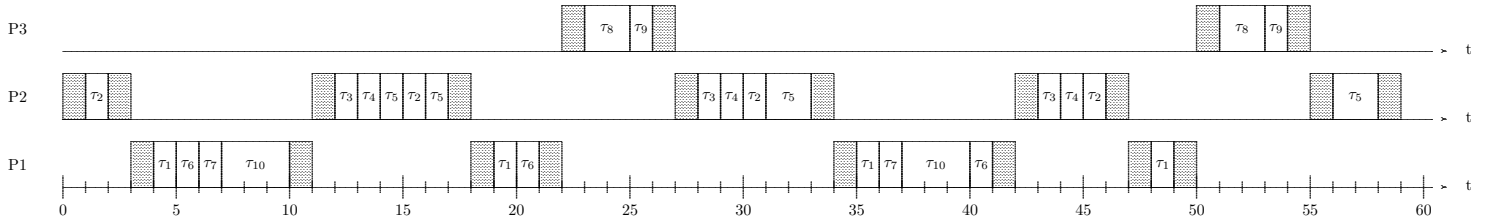
Dans cette configuration, l'implémentation devient 167452 contraintes et 99331 variables. L'optimisation a été arrêtée après quatre heures de calcule pour les deux premières configurations et la troisième a été complétée après trois heures et cinq minutes. Les résultats sont illustrés dans la Figure 3.4. On note que l'overhead de commutation d'une instance est symbolisée par un rectangle gris.



(a) Résultat de la minimisation du temps alloué aux partitions grâce au Critère 3.3



(b) Résultat de la minimisation du temps de réponse grâce au Critère 3.8



(c) Résultat de la minimisation du temps de réponse des tâches τ_1 et τ_{10} grâce au Critère 3.13

FIGURE 3.4: Résultats de l'optimisation

Les observations suivantes sont obtenues à partir de l'analyse des résultats.

— Les configurations $\mathcal{C}$ qui sont admissibles selon les contraintes temporelles et de l'ordre d'exécution des tâches existent grâce au fait que le solveur trouve une solution qui satisfait toutes les contraintes.
— La comparaison de trois configurations résultantes montre que la solution dépend du

critère choisi :

- Le temps alloué aux partitions $\sum_{j=1}^{m}\sum_{k=1}^{uppb}(E_{j,k}-S_{j,k})$ pour la Figure 3.4a (égal à 57) est inférieur à celui de les Figures 3.4b et 3.4c (les deux égaux à 59) ;
- De la même façon, le temps de réponse global $\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}$ pour la Figure 3.4b (égal à 255) est inférieur à celui de les Figures 3.4a et 3.4c (égaux à 274 et 268 respectivement) ;
- Le temps de réponse des tâches τ_1 et τ_{10}, $\sum_{i\in\{1,10\}}\sum_{j=1}^{m}\sum_{l=0}^{\Theta-1}\Phi^{C}_{i,j,l}$, pour la Figure 3.4c (égal à 40) est inférieur à celui de les Figures 3.4a et 3.4b (égaux à 63 et 67 respectivement).

- Même si la limite supérieure du nombre des instances est $uppb = 10$, le nombre des instances réelles est moins de cinq pour les trois configurations considérées. De plus, pour une partition spécifique, le nombre d'instances change selon le critère choisi. Par exemple, la partition P_2 présente quatre instances réelles pour les Figures 3.4a et 3.4b, mais cinq pour la Figure 3.4b.
  Finalement, le nombre des instances réelles n'est pas le même pour toutes les partitions d'une même configuration. La Figure 3.4c montre que les partitions P_1, P_2 et P_3 ont deux, cinq et quatre instances réelles respectivement. Pour cette raison, la capacité du modèle pour ajuster le nombre des instances réelles selon le critère choisi est bien démontrée.
- En comparant la Figure 3.4a avec la Figure 3.4b, la première présente un temps alloué aux partitions petit due au fait que le solveur essaie de sauvegarder une instance réelle pour la partition P_3 et donc de cette façon réduire le nombre de commutations (moins overheads). Comme la tâche $\tau_5 \in P_3$ a une périodicité de $T_5 = 20$, elle requière trois exécutions complètes sur $\Theta = 60$ pendant que P_3 présente seulement deux instances réelles. Pour réconcilier ces deux aspects, la deuxième instance réelle de la partition P_3 est allouée à l'instant $t = 2T_5$. Pour cette raison, elle permet d'exécuter τ_5 deux fois dans la deuxième instance réelle.
- Cet exemple démontre la capacité du modèle pour distribuer (d'une façon efficiente) les tâches entre les partitions différemment selon le critère. Par exemple, $P_3 = \{\tau_5, \tau_{10}\}$ pour la Figure 3.4a, $P_3 = \{\tau_7, \tau_8, \tau_9\}$ pour la Figure 3.4b et $P_3 = \{\tau_8, \tau_9\}$ pour la Figure 3.4c.
- La capacité du modèle pour gérer la préemption des tâches est démontrer dans la Figure 3.4a. La tâche $\tau_{10} \in P_3$ est partiellement exécutée pendant une unité de temps à l'instant $l = 39$ et préemptée à l'instant $l = 40$ par une tâche de plus haute priorité ($\tau_5 \in P_3$). La tâche τ_{10} complète donc son exécution à l'instant $l = 42$ et $l = 43$. La tâche τ_8 est aussi préemptée à l'instant $l = 20$ par la tâche τ_4. De la même façon, dans la Figure 3.4c la tâche $\tau_5 \in P_2$ est partiellement exécutée pendant une unité de temps à l'instant $l = 14$ et préempté à l'instant $l = 15$ par une tâche de plus haute priorité ($\tau_2 \in P_2$). La tâche τ_5 complète son exécution à l'instant $l = 16$.

3.6 Conclusion

Ce chapitre aborde le modelage et optimisation de l'allocation de partitions et l'ordonnancement de différentes tâches dans un système avec différent niveaux critiques. Comme le modèle présenté est entièrement linéaire, l'utilisation d'algorithmes de programmation linéaire devient une proposition efficiente pour résoudre et optimiser le problème. Dans ce contexte, on a montré que le critère a un fort impact sur la solution.

Il est important de remarquer que le modèle proposé est vraiment flexible, ce qui permet d'ajouter des contraintes supplémentaires pour représenter des interactions additionnelles. Il serait intéressant par la suite d'ajouter une extension à ce modèle pour la partition temporelle d'un système avec multiples niveaux critiques sur une plateforme multi-cœurs.

3.7 Références

[1] ARINC. Arinc specification 651 : Design guidance for integrated modular avionics. *Aeronautical Radio, Inc, Annapolis, MD*, 1991.

[2] John Rushby. Partitioning in avionics architectures : Requirements, mechanisms, and assurance. Technical report, DTIC Document, 2000.

[3] ARINC. Arinc specification 653 : Avionics application software standard interface. *Avionics Application Software Interface, Annapolis, MD*, 1997.

[4] Hyun-Wook Jin and Sanghyun Han. Temporal partitioning for mixed-criticality systems. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–4. IEEE, 2011.

[5] Zhong Deng and Jane W-S Liu. Scheduling real-time applications in an open environment. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 308–319. IEEE, 1997.

[6] Robert I Davis and Alan Burns. Hierarchical fixed priority pre-emptive scheduling. In *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, pages 10–pp. IEEE, 2005.

[7] Tei-Wei Kuo and Ching-Hui Li. A fixed-priority-driven open environment for real-time applications. In *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*, pages 256–267. IEEE, 1999.

[8] Giuseppe Lipari and Enrico Bini. Resource partitioning among real-time applications. In *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, pages 151–158. IEEE, 2003.

[9] Saowanee Saewong, Ragunathan Raj Rajkumar, John P Lehoczky, and Mark H Klein. Analysis of hierarchical fixed-priority scheduling. In *null*, page 173. IEEE, 2002.

[10] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pages 2–13. IEEE, 2003.

[11] Joseph Porter and Csanád Szabó. Partition configuration for real-time systems with dependencies. In *Proceedings of the 21st International conference on Real-Time Networks and Systems*, pages 87–96. ACM, 2013.

[12] Domiţian Tămaş-Selicean and Paul Pop. Design optimization of mixed-criticality real-time embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3) :50, 2015.

[13] Arvind Easwaran, Insup Lee, Oleg Sokolsky, and Steve Vestal. A compositional scheduling framework for digital avionics systems. In *Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA'09. 15th IEEE International Conference on*, pages 371–380. IEEE, 2009.

[14] Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[15] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization.* John Wiley & Sons, 2014.

[16] Anurag Agarwal, Hasan Pirkul, and Varghese S Jacob. Augmented neural networks for task scheduling. *European Journal of Operational Research*, 151(3) :481–502, 2003.

[17] Irina N Lushchakova and Vitaly A Strusevich. Scheduling incompatible tasks on two machines. *European Journal of Operational Research*, 200(2) :334–346, 2010.

[18] Nguyen Huynh Tuong, Ameur Soukhal, and Jean-Charles Billaut. A new dynamic programming formulation for scheduling independent tasks with common due date on parallel machines. *European Journal of Operational Research*, 202(3) :646–653, 2010.

[19] Maciej Drozdowski. Scheduling multiprocessor tasks - an overview. *European Journal of Operational Research*, 94(2) :215–230, 1996.

[20] Jacek Błażwicz and Zhen Liu. Scheduling multiprocessor tasks with chain constraints. *European Journal of Operational Research*, 94(2) :231–241, 1996.

[21] Jacek Błażewicz, Paolo Dell'Olmo, Maciej Drozdowski, and Przemysław Maczka. Scheduling multiprocessor tasks on parallel processors with limited availability. *European Journal of Operational Research*, 149(2) :377–389, 2003.

[22] C Oğuz, M Fikret Ercan, TC Edwin Cheng, and Yu-Fai Fung. Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. *European Journal of Operational Research*, 149(2) :390–403, 2003.

[23] Łukasz Kuszner and Michał Małafiejski. A polynomial algorithm for some preemptive multiprocessor task scheduling problems. *European journal of operational research*, 176(1) :145–150, 2007.

[24] Klaus H Ecker. Scheduling of resource tasks. *European Journal of Operational Research*, 115(2) :314–327, 1999.

[25] Klaus H Ecker and Jatinder ND Gupta. Scheduling tasks on a flexible manufacturing machine to minimize tool change delays. *European Journal of Operational Research*, 164(3) :627–638, 2005.

[26] Mohan Krishnamoorthy, AT Ernst, and Davaa Baatar. Algorithms for large scale shift minimisation personnel task scheduling problems. *European Journal of Operational Research*, 219(1) :34–48, 2012.

[27] Shih-Wei Lin and Kuo-Ching Ying. Minimizing shifts for personnel task scheduling problems : A three-phase algorithm. *European Journal of Operational Research*, 237(1) :323–334, 2014.

[28] Ali Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 2015.

Chapitre 4

Reconnaissances

Ce travail a été possible grâce au support de tout le département Génie Électrique. Surtout, à l'engagement de Guchuan Zhu qui m'a orienté quand j'ai été perdu. De plus, j'aimerais remercier l'aide de Hugo Lhachemi qui a été fondamentale pour développer le deuxième chapitre. Finalement, l'apport de David Saussié a aidé à bien structurer les idées et formaliser le travail.