HAL
open science

Systèmes véhiculaires à domaines de sécurité et de criticité multiples : une passerelle systronique temps réel

Philippe Thierry

To cite this version:

Philippe Thierry. Systèmes véhiculaires à domaines de sécurité et de criticité multiples : une passerelle systronique temps réel. Informatique. Université Paris-Est, 2014. Français. NNT : 2014PEST1102 . tel-01134229

HAL Id: tel-01134229

https://theses.hal.science/tel-01134229v1

Submitted on 23 Mar 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ÉCOLE DOCTORALE UNIVERSITÉ PARIS-EST

Mathématiques et STIC

THÈSE

en vue de l'obtention du titre de

DOCTEUR

de l'Université Paris-Est

Spécialité : INFORMATIQUE

PHILIPPE THIERRY

Systèmes véhiculaires à domaines de sécurité et de criticité multiples : une passerelle systronique temps réel

---

soutenue le mercredi 2 juillet 2014

Jury

Directeur :	**LAURENT GEORGE**	–	Université Paris-Est (LIGM)
Rapporteurs :	**YE-QIONG SONG**	–	Université de Lorraine
	YVON TRINQUET	–	Université Nantes 1 (IRCCyN)
Examinateurs :	**FABIEN GERMAIN**	–	SYSGO France
	JEAN-FRANÇOIS HERMANT	–	ECE Paris
	ISABELLE PUAUT	–	Université Rennes 1
	YVES SOREL	–	INRIA Rocquencourt (AOSTE)

Doctorat préparé au laboratoire
UPEC – LiSSi (EA 3956)
Laboratoire Images, Signaux et Systèmes Intelligents
Domaine Chérioux
122 rue Paul Armangot
94400 Vitry sur Seine

Doctorat en collaboration avec
Thales Communications & Security
Sofware Architecture & Tools, laboratoire Architectures & Technologies
4, avenue des Louvresses
Bâtiment H
92260 Genevilliers Cedex

À ma mère, qui n'aura pu me voir finir mon doctorat, et à mon père,

À mon fils nouveau-né et à ma femme,
À tous les autres.

Remerciements

On m'avais dit que le diplôme d'ingénieur, ce n'est pas suffisant pour pouvoir *tenir* une carrière entière. On me disait "tu devrais faire un MBA", car oui, avoir un diplôme de finances, allez savoir pourquoi, il paraît que ça sert dans l'ingénierie... Mais voila, la finance, je déteste ça... Même à la maison, c'est ma femme qui gère les comptes (sûrement beaucoup mieux que moi).
Mais au final, j'ai toujours préféré la technique. Même si dans l'industrie, elle n'est à mon sens pas considérée à sa juste valeur. J'ai donc voulu faire un doctorat. N'allez pas penser que j'ai fait un doctorat juste pour le plaisir d'avoir un diplôme en plus. Si ça avait été le cas, j'aurais fini par m'échapper de ce long et obscur tunnel qu'est une thèse, bien avant sa fin. J'ai choisi un doctorat car cela avait un sens pour moi : me spécialiser dans un domaine technique qui m'intéresse, et m'enrichir auprès de gens qui ont su m'apprendre beaucoup de choses durant ces dernières années. Il est maintenant temps de tous les remercier.

Il n'y a pas d'ordre particulier à mes remerciements, il ne faut donc pas en chercher un. Néanmoins, je ne peux pas ne pas commencer ceux-ci par les deux personnes qui m'ont accompagné pendant ces (presque) cinq ans, m'ont enrichi de leur compétences et de leur connaissances, ainsi que de leur carnet d'adresses.
Merci donc à JEAN-MARC LACROIX, mon responsable scientifique, qui a su me faire faire autre chose qu'un doctorat pendant toutes ces années, et qui a su me faire participer à diverses affaires, appels d'offre, revues d'architecture, tous en lien avec mon domaine de spécialisation, et qui m'ont permis d'enrichir mes travaux avec des problématiques industrielles concrètes.
Merci également à LAURENT GEORGE, mon directeur de thèse. Merci d'avoir accepté de m'encadrer pendant toutes ces années, d'avoir pris du temps pour me former dans ce domaine complexe qu'est le temps réel, et d'avoir accepter ma tendance maladive à vouloir mettre des problématiques de sécurité partout. Merci de m'avoir emmener à toutes ces conférences, de m'avoir appris à être rigoureux et synthétique, dans tous ces articles et pour finir dans ce présent document. Bravo à lui pour sa patience dans le cadre de la relecture de ce document, malgré mes *TOCs* d'usage de certains mots ou expressions, dont il a ordonné la chasse durant plusieurs mois !

Il serait inconcevable de ne pas remercier mes deux rapporteurs, qui ont eu le courage de prendre en charge ce document et le tirer vers le haut, de par leur vision experte sur les différents sujets qu'il porte. Merci également à tous les autres membres de mon jury, tout d'abord

pour avoir accepté d'en faire partie, mais également d'avoir été présents à divers moments de ma thèse, pour me conseiller et m'aider dans mes travaux. Merci donc à ISABELLE PUAUT, YVES SOREL et bien sûr FABIEN GERMAIN pour leurs paroles expertes. Merci également tout particulièrement à JEAN-FRANÇOIS HERMANT, qui a beaucoup participé à mes travaux et qui m'a beaucoup aidé dans les premiers articles que j'ai publié, tant par son expertise que par sa rigueur que je ne peux que profondément respecter.

Une thèse, c'est beaucoup de solitude dans son travail. Je souhaite donc remercier tous les camarades qui ont permis (et souvent initié) de bonnes tranches de rigolades, tout au long de ces années. Merci donc, dans le désordre le plus total, à PIERRE COURBIN, SYLVAIN LEROY, CLÉMENT DUART ou encore, côté Thales, à Zig et Zog (à savoir BERTRAND FUGANIOLLI et VINCENT HUITRIC). Merci également à FRANCK CURO, tant pour son travail sur la diode logicielle que pour son bon esprit durant toutes ses années. Enfin, merci encore à tous ceux que je pourrais avoir oubliés.

Pour terminer avec ces remerciements, j'aimerais conclure avec une personne qui m'est très cher. Une thèse, c'est également beaucoup de travail le soir et le week-end, pour des préparation d'articles ou encore dans des phases de rédaction. Merci donc à ma femme pour sa patience d'ange tout au long de ces quatre années de travaux durant lesquelles je lui ai volé beaucoup de temps à deux.

Résumé

Dans le cadre de ma thèse, j'ai cherché à répondre, en accord avec Thales Communications & Security, au besoin naissant, dans les systèmes véhiculaires, de passerelles permettant d'interconnecter les différents réseaux embarqués. Ces différents réseaux sont en charge de fonctions très hétérogènes, comme par exemple :

- L'ensemble des fonctions d'aide à la conduite (ABS, limiteur et régulateur de vitesse, etc.)
- Les différentes fonctions multimédia (lecteur multimédia, système en charge des connexions Bluetooth, etc.)

Dans le cadre des véhicules militaires, plusieurs réseaux viennent encore s'ajouter à ceux-ci. Le besoin d'une passerelle d'interconnexion vient du fait qu'il est nécessaire de permettre le transfert de données entre ces réseaux. Ce besoin a été expliqué par la Direction Générale de l'Armement (DGA), et commence également à apparaître sur les marchés à l'export. Le transfert de données doit pouvoir se faire tout en évitant une contagion des contraintes de certification portées par chacun de ces réseaux, pour des raisons de coût financier de la solution. Cela implique donc que la passerelle en charge de ce transfert possède les propriétés de cloisonnement nécessaires pour garantir que les réseaux les plus contraints ne sont pas impactés par cette interconnexion.

Afin de pouvoir construire une telle passerelle, en me basant sur une implémentation logicielle, j'ai travaillé en trois étapes :

1. J'ai formalisé sous forme d'exigences les différentes contraintes liées aux besoins et enjeux de l'usage d'une telle passerelle dans les véhicules militaires. Ces contraintes et enjeux m'ont été fournis par les équipes Thales en charge de l'intégration des différents réseaux dans les véhicules militaires de type blindés légers, dans le cadre de plusieurs entretiens. L'usage d'exigences formelles permet de valider la pertinence de la solution technique par rapport aux contraintes. Cela évite également de définir une solution technique trop complexe et trop riche par rapport au besoin initial.
   La définition des exigences formelles et des hypothèses sur le matériel sous-jacent est proposée dans le Chapitre III.1.

2. Une fois l'ensemble de ces exigences formalisées, j'ai déterminé une solution logicielle, ainsi que les contraintes sur le matériel, permettant de répondre à celles-ci. Les exigences sont globalement séparées en deux grandes familles :
   - Les exigences liées à la sécurité, auxquelles je réponds dans le cadre du Chapitre III.2, en m'appuyant sur les mécanismes de compartimentation logicielle de type Multi-Level Security et en proposant des mécanismes complémentaires pour répondre aux différentes exigences de sécurité
   - Les exigences liées à la sureté de fonctionnement, auxquelles je réponds, pour la partie temps réel, dans le Chapitre III.3.1. Je prends alors en compte, dans la réponse aux exigences de temps réel, les contraintes apportées par le besoin sécuritaire. J'ai alors étudié les propriétés d'ordonnancement hiérarchique et de criticité mixte, nécessaires du fait des contraintes de puissance et de consommation exprimées par le client.
3. Afin de valider la solution, j'ai choisi un certain nombre de composants logiciels majeurs de la solution, afin de valider leurs propriétés et leur comportement. La validation s'est effectuée sous formes de plusieurs publications [San+12 ; Thi+11 ; TGH11 ; TGL13], d'un maquettage général décrit dans le chapitre III.4 et de deux implémentations se rapprochant d'un produit industriel, ayant donné lieu à trois brevets [TLC13 ; Thi+13a ; Thi+13b].

Ma thèse a ainsi permis de valider une première solution technique permettant de répondre à la plupart des contraintes apportées par le client. Cette solution a de plus pour but d'être un framework ouvert, où il est possible d'ajouter ou de retirer des modules sécuritaires ou temps réel en fonction du besoin, tout en restant sur la même architecture générale dont le but est de garantir un cloisonnement efficace, à la fois spatial et temporel. Ce principe permet ainsi de limiter le coût d'instanciation d'une telle passerelle dans divers types de véhicules militaires pour lesquels les contraintes peuvent varier.

Abstract

In this thesis, I have worked in collaboration with Thales Communications & Security, on the new need of network gateways in embedded vehicular systems. These networks manage complex heterogeneous networks, such as :

- network managing driving functions (ABS, ESP, speed regulator, etc.)
- multimedia networks (audio/video functions, local user network interfaces such as Bluetooth, etc.)

In military vehicles, other networks are integrated to these initial ones, for various military needs. The need of a network gateway is the consequence of the new interconnections of these networks, in order to transmit various data from a network to another. This need, for military vehicles, has been described by the French Direction Générale de l'Armement (DGA) and begins to appear in some foreign markets. The data transfer between networks must be done without impacting each network specific safety and/or security requirements, in order to avoid a certification requirement on less secure and/or safe networks. Such constraint imply that the gateway has to be compliant to specific partitioning requirements.

In order to propose a software solution for such a gateway, I've considered three steps :

1. I've formalized safety, security and physical requirements in order to support the various constraints associated to the needs of such a gateway in military vehicles. These constraints have been listed by Thales teams interacting with the French DGA for the lightweight military vehicles market. Using formalized requirements make the validation of the technical solution easier. These requirements also permit us to avoid potentially costly over-specifications.
2. Using formalized requirements, I've proposed a software solution and a list of properties and constraints on the underlying hardware to guarantee the compliance of the overall hardware/software solution. These requirements, properties and constraints are defined in the Chapter III.1.

   The requirements are globally separated in two families :
   - the security requirements, to which I respond in the Chapter III.2, through the usage of software partitioning based on existing works around MultiLevel Security, with new specific components and principles.
   - the safety requirements, to which I respond for real time aspects, in Chapter III.3.1. Then, I consider in my response to real time requirements, the security specific con-

straints. I've studied hierarchical scheduling and mixed criticality properties, needed in order to be compliant to the power and consumption requirements of such a gateway and for the partitioning principles due to security specific requirements.

3. In order to validate the solution, we have chosen some important new software components of the solution, and validated their specific properties and behavior. This validation has been done through various publications [San+12 ; Thi+11 ; TGH11 ; TGL13], a proof of concept described in Chapter III.4 and two pre-industrial implementations, protected by three patents [TLC13 ; Thi+13a ; Thi+13b].

During my thesis, I have then validated a first technical solution responding to most of French DGA constraints. This solution is designed as an open framework, where it is possible to add or to delete security-specific or real-time specific software modules depending on the need, without impacting the overall spatial and temporal partitioning properties.
Such principle allows the instantiation of multiple gateways for multiple vehicular-specific needs, based on the same overall architecture.

Table des matières

Résumé v

Abstract vii

I Problématique générale et concepts clefs 1

I.1 Contexte et Problématique générale 3

I.1.1 Problématique des systèmes véhiculaires militaires 3

I.1.2 Notions et définitions générales 4

I.1.3 Notions et terminologies véhiculaires 8

I.1.3.1 Généralités sur les réseaux véhiculaires 8

I.1.3.2 Les systèmes systroniques 8

I.1.4 Enjeux et limitations de l'existant 10

I.1.4.1 Les enjeux des systèmes systroniques 10

I.1.4.2 Limitations de l'existant 11

I.1.5 Synthèse et éléments de solution 13

I.2 Concepts 17

I.2.1 Principes généraux de la sécurité 17

I.2.1.1 Terminologie militaire de la sécurité 17

I.2.1.2 Fuite d'informations : Canaux cachés et canaux auxiliaires 20

I.2.1.3 Ce que cible la sécurité logicielle 21

I.2.1.4 Cibles de sécurité et certifiabilité 21

I.2.2 Principes généraux des systèmes temps réel 22

I.2.2.1 Rappels sur les politiques d'ordonnancement 22

I.2.3 Principes généraux de la virtualisation 23

I.2.3.1 Concepts liés à la virtualisation et la compartimentation 23

I.2.3.2 Les solutions de virtualisation type I 24

I.2.3.3 Les solutions de virtualisation type II 26

I.2.3.4 Les solutions de virtualisation applicative 26

I.2.4 Les interactions entre le matériel, la sécurité et le temps réel 27

I.2.4.1 Principes de base de la MMU 27
I.2.4.2 A propos des contrôleurs de cache 31
I.2.4.3 Les algorithmes de gestion de caches 37

II État de l'art des solutions de sécurité et de temps réel 41

II.1 État de l'art de l'interconnexion sécurisée de domaines de sécurité hétérogènes 43
II.1.1 Sécurisation par séparation physique des traitements 43
II.1.1.1 Interconnexion sécurisée par passerelle de type diode matérielle . . . 43
II.1.1.2 Firewall d'interconnexion de domaines de sécurité hétérogènes 45
II.1.2 Architectures MLS (MultiLevel Security) et MILS (Multi-Independent Level of Security) 46
II.1.2.1 Systèmes MLS, MILS et la haute sécurité 46
II.1.2.2 MLS et principe de virtualisation 48
II.1.3 Interconnexion sécurisée inter-domaines de sécurité 48
II.1.3.1 Le contrôle de flux et les passerelles de sécurité 48
II.1.4 Synthèse 49

II.2 État de l'art des solutions de virtualisation 51
II.2.1 Étude de la sécurité des solutions de virtualisation existantes 52
II.2.1.1 Étude de la solution Xen 52
II.2.1.2 Étude de la solution QubesOS 54
II.2.1.3 Étude de la solution LXC 55
II.2.1.4 Étude de la famille L4 56
II.2.1.5 Étude de la solution PikeOS 56
II.2.2 Synthèse 58

II.3 Focus sur l'ordonnancement hiérarchique et la criticité mixte 61
II.3.1 Solutions pour l'ordonnancement hiérarchique de tâches 61
II.3.1.1 Principes généraux de l'ordonnancement hiérarchique 61
II.3.1.2 Synthèse 64
II.3.2 État de l'art de l'ordonnancement de tâches à criticité mixte 65
II.3.2.1 Principe de la criticité mixte 65
II.3.2.2 Synthèse 66

III Proposition de solution logicielle de passerelle systronique 69

III.1 Rappel des contraintes et hypothèses générales 71
III.1.1 Matrice de conformité de la solution proposée 71

III.1.2 Définition des hypothèses générales 74
III.1.2.1 Hypothèses sur l'architecture matérielle 74

III.2 Proposition d'une architecture logicielle sécurisée pour le MLS orientée traitement de flux 77
III.2.1 Proposition d'architecture et choix structuraux 78
III.2.1.1 Principe de passerelle de traitement de flux MultiLevel Security . . . 78
III.2.1.2 Présence d'éléments non certifiables 79
III.2.1.3 Moniteurs de sécurité 80
III.2.1.4 Synthèse des choix structuraux 81
III.2.2 Caractérisation du risque sécuritaire 81
III.2.2.1 Définition des données à protéger 81
III.2.2.2 Intégration des problématiques réseaux 84
III.2.2.3 Définition des profils d'attaque 85
III.2.2.4 Synthèse du risque sécuritaire 87
III.2.3 Définition des moniteurs de sécurité 88
III.2.3.1 Protection de l'intégrité des données 88
III.2.3.2 Garantie de transmission de flux sans canal auxiliaire inverse 89
III.2.3.3 Conformité des données transmises à la politique de sécurité 92
III.2.3.4 Compartimentation garantie 94
III.2.4 Problématique d'exploitation des canaux auxiliaires 94
III.2.4.1 Caches processeurs et canaux cachés 94
III.2.4.2 Hypothèses sur le comportement du contrôleur de cache proposé . . . 95
III.2.4.3 Formalisme de l'interface de contrôle du contrôleur de cache 95
III.2.4.4 Description de l'interface de contrôle du partitionnement du contrôleur de cache 96
III.2.4.5 Principe et automate 97

III.3 Compatibilité d'une architecture sécurisée avec des contraintes de temps réel 103
III.3.1 Architectures compartimentées et temps réel 103
III.3.1.1 Rappel des exigences de temps réel 104
III.3.1.2 Hiérarchie et Time Division Multiplexing 104
III.3.2 Moniteurs de sécurité, fonctions temps réel et environnements non sécurisables 107
III.3.2.1 Compatibilité des moniteurs de sécurité avec le temps réel 107
III.3.2.2 Compatibilité des environnements non certifiables avec le temps réel . 107
III.3.3 Intégration du support des flux à criticité mixte 109
III.3.3.1 Rappels et enjeux 109
III.3.3.2 Modèle et Définitions 110
III.3.3.3 Ordonnançabilité d'un jeu de tâches à criticité mixte 114
III.3.3.4 Retarder l'accroissement de criticité 116

III.3.3.5 Marge de tolérance 117
III.3.3.6 Implémentation de la marge de tolérance 123
III.3.3.7 Réinitialiser la Criticité du système 126
III.3.3.8 Simulations 127
III.3.3.9 Conclusion 130

III.4 Vers une passerelle sécurisée et temps réel pour le traitement de flux multi-critiques : cas d'usage véhiculaire 131
III.4.1 Rappel du besoin vétronique 131
III.4.1.1 Évolution des systèmes véhiculaires 131
III.4.1.2 Problématiques techniques des architectures vétroniques civiles et militaires 133
III.4.2 Définition d'une architecture logicielle pour une passerelle systronique . . . 136
III.4.2.1 Définition d'une architecture de filtrage multi-fonctions pour les systèmes systroniques 136
III.4.3 Limitations de la solution technique et compléments 147

Conclusion générale 153

Glossaire des termes techniques 159

Bibliographie 163

Table des figures

I.2.1 Architecture logicielle générique des solutions de virtualisation de type I . . 24
I.2.2 Architecture logicielle d'une solution de virtualisation type II 27
I.2.3 Architecture logicielle d'une solution de virtualisation applicative 28
I.2.4 Principe général du mapping de page mémoire dans les espaces d'adressage des processus 30
I.2.5 Architecture en couche générique 31
I.2.6 Comparaison des architectures de types cache through et cache lookaside . 33
I.2.7 Automate à état de l'algorithme pseudo-LRU du Power Quicc 2 Pro 36
I.2.8 Intégration et déplacement des données accédées fréquemment 36
I.2.9 Principe de deuxième chance du SLRU 37

II.1.1 Architecture générale d'une passerelle de type diode matérielle 44

II.2.1 Architecture logicielle de la solution PikeOS 57

II.3.1 Répartition des tâches du noyau Linux par jeux autonomes, en fonction des exigences de réactivité 63

III.2.1 Filtrage des entrées/sorties par un élément logiciel de confiance 80
III.2.2 Exemple de séparation protocolaire couche transport 85
III.2.3 Répartition des surfaces d'attaque sur la passerelle de sécurité 86
III.2.4 Automate du moniteur de sécurité émetteur périodique 92
III.2.5 Design d'architecture macroscopique du Contrôleur de cache L1 data proposé 97
III.2.6 Automate à état du contrôleur de cache 98
III.2.7 Structure du registre PCR 98
III.2.8 Slot configuration table 100
III.2.9 Structure du registre CSSR 100

III.3.1 Ordonnancement hiérarchique sur base TDM 105
III.3.2 Traitements multi-critiques de flux réseaux 109
III.3.3 Modèle étendu des états de tâches 112
III.3.4 Non Optimalité de l'algorithme EDF 120

III.3.5 Espace de tolérance $A(1)$ 122
III.3.6 Exemple de $\mathsf{LET}_i^1(0)$ 125
III.3.7 Exemple de $\mathsf{LET}_i^2(0)$ 125
III.3.8 Récupération de la marge de tolérance 125
III.3.9 Diminution de criticité 127
III.3.10 Distribution triangulaire pour la tâche τ_i 128
III.3.11 Pourcentage moyen de travaux supprimés en fonction de la charge 129

III.4.1 Exemple d'intégration d'un système systronique pour véhicule blindé léger 134
III.4.2 Architecture logicielle type d'une passerelle filtrante 138
III.4.3 Contrôle du débit à l'émission de la passerelle par l'émetteur périodique . . 141
III.4.4 Latence au démarrage de la tâche de priorité la plus élevée 143
III.4.5 Architecture logicielle générale de la maquette 144
III.4.6 Test 1 : WCET mesurée pour Filtre, Diode et Émetteur périodique 146
III.4.7 Test 1 : Variation de période pour la tâche Filter 146
III.4.8 Test 1 : Variation de période pour la tâche Diode 146
III.4.9 Test 1 : Variation de période pour la tâche Émetteur périodique 146
III.4.10 Test 2 : Variation de la période de la tâche Filter 148
III.4.11 Test 2 : Variation de la période de la tâche Diode 148
III.4.12 Test 2 : Variation de la période de la tâche Émetteur périodique 148

Liste des tableaux

I.2.1 Temps d'accès mémoire sur un Intel Core 2 duo 32

III.1.0 Liste des exigences générales induites par la problématique 74

III.2.1 Profils d'attaques et impact sur les exigences sécuritaires 87
III.2.2 Description des champs du registre PCR 99
III.2.3 Description des registre de configuration des slots de cache 100
III.2.4 Description des champs du registre CSSR 101

III.3.1 Définition des variables de niveau compartiment et de niveau tâches 105
III.3.2 Formalisation des contraintes sur l'ordonnancement hiérarchique de tâches en mode TDM/EDF 106
III.3.3 Niveau de criticité des différentes tâches de traitement de flux 109
III.3.4 Points d'ordonnancement dans l'espace $A(1)$ 122
III.3.5 Points d'ordonnancement dans l'espace $A(2)$ 122
III.3.6 Points d'ordonnancement dans l'espace $A(3)$ 123
III.3.7 Exemple de trois tâches ordonnancées en priorité fixe 124
III.3.8 Pourcentage moyen de travail supprimés en fonction de $U_{\max}(\tau_i)$ 129

III.4.1 Description de la configuration utilisée pour la mesure 142
III.4.2 Description des choix d'ordonnancement de tâches pour la maquette 143
III.4.3 Profil du jeu de test 1 147
III.4.4 Profil du jeu de test 2 147

Liste de mes publications

Conférences internationales

[San+12] François SANTY, Laurent GEORGE, Philippe THIERRY et Joël GOOSSENS. “Relaxing mixed-criticality scheduling strictness for task sets scheduled with FP”. Dans : *Proceeding of the 24th Euromicro Conference on Real-Time Systems (ECRTS’2012)*. IEEE Computer Society, 2012, pages 155–165 (cf. pages vi, viii, 15).

[TGL13] Philippe THIERRY, Laurent GEORGE et Jean-Marc LACROIX. “A framework for a secure embedded filtering connector for multi-criticality systronic systems”. Dans : *Proceedings of the 18th International Conference on Emerging Technologies & Factory Automation (ETFA’2013)*. IEEE Computer Society, 2013 (cf. pages vi, viii, 15).

[Thi+11] Philippe THIERRY et al. “Toward a predictable and secure data cache algorithm : a cross-layer approach”. Dans : *Proceedings of the 10th International Symposium on Programming and Systems (ISPS’2011)*. IEEE Computer Society, 2011, pages 148–155 (cf. pages vi, viii, 14).

Workshops

[TGH11] Philippe THIERRY, Laurent GEORGE et Jean-François HERMANT. “Real-time scheduling analysis for ARINC-based virtualized systems”. Dans : *Proceedings of the 10th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP’2011)*. 2011, pages 106–109 (cf. pages vi, viii, 14).

Brevets

[TLC13] Philippe THIERRY, Jean-Marc LACROIX et Olivier CAZADE. *Appareil informatique comportant un environnement d’exécution et un compartiment réservé*

avec une redirection vers ledit compartiment de requêtes entre l'environnement d'exécution et un dispositif externe, et système informatique comportant un tel appareil informatique. Patent FR 13 00146. Jan. 2013 (cf. pages vi, viii, 81).

[Thi+13a] Philippe THIERRY, Jean-Marc LACROIX, Franck CURO, Dominique RAGOT et Fabien GERMAIN. *Equipement de sécurité de cloisonnement entre des premier et second domaines, augmenté d'une fonctionnalité d'audit*. Patent FR 13 03074. Déc. 2013 (cf. pages vi, viii, 15, 90, 91).

[Thi+13b] Philippe THIERRY, Jean-Marc LACROIX, Franck CURO, Dominique RAGOT et Fabien GERMAIN. *Equipement de sécurité de cloisonnement entre des premier et second domaines, comportant un composant de contrôle*. Patent FR 13 03075. Déc. 2013 (cf. pages vi, viii, 15, 90).

Première partie

Problématique générale et concepts clefs

Chapitre I.1

Contexte et Problématique générale

Sommaire

I.1.1 Problématique des systèmes véhiculaires militaires **3**
I.1.2 Notions et définitions générales **4**
I.1.3 Notions et terminologies véhiculaires **8**
I.1.3.1 Généralités sur les réseaux véhiculaires 8
I.1.3.2 Les systèmes systroniques 8
I.1.4 Enjeux et limitations de l'existant **10**
I.1.4.1 Les enjeux des systèmes systroniques 10
I.1.4.2 Limitations de l'existant 11
I.1.5 Synthèse et éléments de solution **13**

I.1.1 Problématique des systèmes véhiculaires militaires

Le client étatique français (Direction Générale de l'Armement) demande aujourd'hui d'interconnecter divers réseaux embarqués dans un véhicule. Dans le cadre des véhicules militaires, ces réseaux peuvent transmettre des informations ayant des contraintes de temps réel, de fiabilité mais également de sécurité.
La contrainte sécuritaire est présente dans les véhicules militaires, mais également dans d'autres domaines comme le médical (accès aux données médicales du patient dans le véhicule) ou la police (accès aux différentes bases de données). Dans le même temps, les exigences de temps réel et de fiabilité restent fortement présentes dans le cadre des véhicules militaires. Ainsi la fiabilité de la gestion du système d'arme, en charge de débrayer les mécanismes d'attaque lors de l'ouverture d'une des portes du véhicule, est critique, afin d'éviter tout accident. De la même manière, les contraintes de temps réel sont fortement présentes dans le cas du système de gestion du blindage transparent, en charge de remonter l'ensemble des flux des caméras extérieures au conducteur, dans un véhicule sans fenêtre.

Jusqu'à présent, les domaines à fortes contraintes de temps réel, de fiabilité et de sécurité restaient physiquement séparés. De nos jours, avec l'accroissement de la complexité des éléments déployés dans les véhicules, il est fortement demandé par les différents clients étatiques de fusionner ces domaines sur un même matériel, à la seule exception des éléments à forte contrainte de fiabilité, afin de limiter le poids, le volume et la consommation des différents éléments technologiques présents.

De plus, il est également demandé de permettre une meilleure centralisation de la supervision des différents composants, que ces derniers fassent partie d'un domaine à forte contrainte de temps réel, de fiabilité ou d'un domaine à forte contrainte de sécurité.

Ainsi, il devient nécessaire de définir une passerelle permettant d'interagir à la fois avec des domaines à fortes contraintes temps réel, des domaines à fortes contraintes de fiabilité et plusieurs domaines de sécurité. Cette passerelle doit alors assurer à la fois :

- un cloisonnement strict des domaines de sécurité disjoints ;
- le transfert de données du domaine temps réel tout en assurant une latence de traversée maximum garantie ;
- le transfert de données de supervision du domaine à fortes contraintes de fiabilité sans pour autant impacter ces éléments.

Afin de présenter plus précisément les enjeux et limitations associés à l'intégration d'une passerelle pour interconnecter les différents réseaux embarqués dans le véhicule, il est nécessaire de définir tout d'abord un certain nombre de concepts. Ceux-ci sont décrits dans les Sections I.1.2 et I.1.3. La description précise des enjeux, ainsi que les premiers éléments de solutions, sont détaillés dans la Section I.1.4.

I.1.2 Notions et définitions générales

Notions d'architecture

Dans le cadre des architectures véhiculaires, il existe un grand nombre de systèmes.

DÉFINITION I.1.2.1
On appelle système un ensemble d'éléments matériels et logiciels interagissant ensemble dans le but d'effectuer une fonction ou un ensemble de fonctions connexes donné.

DÉFINITION I.1.2.2
On appelle système de systèmes un ensemble complexe faisant s'interconnecter plusieurs systèmes entre eux. Une telle interconnexion peut être par exemple la conséquence d'un besoin de supervision centralisée.

Dans un véhicule militaire, on retrouve un système de systèmes, en charge de gérer, entre autres :

- le système d'armes, en charge de gérer l'armement et le blindage ;
- le système de bord (appelé aussi réseau de bord du fait de son très grand nombre de liens d'interconnexion), en charge de gérer les composants moteur ;
- le système de communication, en charge de gérer les communications entre le véhicule et l'extérieur (autres véhicules, le quartier général, etc.).

DÉFINITION I.1.2.3
Lorsqu'un système de systèmes est construit pour être intégré à un véhicule, et est en charge d'associer des systèmes véhiculaires (réseau de bord) à des systèmes de communications complexes, on parle alors de système systronique.

Pour être intégrable dans un véhicule, un élément matériel doit être compatible avec un facteur de forme spécifique.

DÉFINITION I.1.2.4
On appelle facteur de forme les dimensions, le type d'alimentation, le nombre de ports ou encore le positionnement des connecteurs. Il existe un grand nombre de facteurs de formes industriels, comme le microTCA [Wal08].

Dans le cadre de ma thèse, je cible les problématiques d'interconnexions internes aux systèmes systroniques.

Notions et terminologies de sécurité

Principes généraux de la sécurité

Un système d'information est dit sécurisé lorsqu'il respecte, pour les données qu'il traite, les 3 principes de sécurité suivants :
- disponibilité ;
- intégrité ;
- confidentialité.

Les définitions suivantes correspondent à des principes sécuritaires. Elles peuvent ne pas correspondre aux définitions liées à la sûreté de fonctionnement. Ces définitions sont présentes dans le National Information Assurance Glossary [Sch10].

DÉFINITION I.1.2.5
La disponibilité (au sens sécuritaire) est la propriété d'un objet à être accessible et utilisable sur demande d'une entité autorisée.

DÉFINITION I.1.2.6
L'intégrité (au sens sécuritaire) est une propriété d'un objet à ne pas avoir été modifié par un quelconque sujet non autorisé.

Dans la pratique, l'intégrité des données est le plus souvent vérifiée par des algorithmes de hachage ou des sommes de contrôle (par exemple le CRC32 des trames réseau Ethernet ou encore les algoritmes SHA1 ou MD5 pour des besoins plutôt sécuritaires). On s'appuie également sur la cryptographie asymétrique pour générer des signatures, afin de garantir l'intégrité de la somme de contrôle.

DÉFINITION I.1.2.7
La confidentialité est la propriété d'un objet à n'être accessible qu'à des sujets ayant les droits suffisants pour accéder à l'information qu'il porte.

DÉFINITION I.1.2.8
On appelle domaine de sécurité un système dans lequel l'ensemble des éléments doivent respecter les mêmes contraintes d'intégrité, de confidentialité et de disponibilité.

DÉFINITION I.1.2.9
Pour garantir la sécurité d'un domaine de sécurité donné, on définit une politique de sécurité, en charge de préciser l'ensemble des fonctions de sécurité nécessaire au respect des contraintes de sécurité du domaine.

Le niveau de sécurité d'un système se détermine grâce à un standard international : les Critères Communs (CC) pour l'évaluation sécuritaire des technologies de l'information [Pro14]. Ceux-ci définissent 7 niveaux d'assurance (confer Définition I.1.2.12), appelés EAL (Evaluation Assurance Level), de EAL 1 (le plus faible) à EAL 7 (le plus élevé).

La fragilité d'un système en terme de sécurité dépend entre autres de sa surface d'attaque.

DÉFINITION I.1.2.10
On appelle surface d'attaque l'ensemble des portes d'entrées permettant d'impacter la sécurité du système. Il peut s'agir d'une interface réseau non protégée ou encore de processus logiciels possédant des failles exploitables. La mise en place de fonctions de sécurité (comme par exemple un pare-feu de confiance) permet de réduire la surface d'attaque.

Notions liées à la fiabilité et au temps réel

Les systèmes véhiculaires militaires intègrent des fonctions impactant la vie des passagers. En conséquence, ils possèdent des contraintes de fiabilité et des contraintes de temps réel.
La fiabilité définit la capacité d'un système à exécuter ses traitements sans erreur. Ainsi, un système fiable à 99,9% assure un traitement sans erreur dans 99,9% de sa durée d'exécution. Plus le système assure une fiabilité proche de 100%, plus son niveau de fiabilité est élevé.
On dit qu'un système est temps réel lorsqu'il doit garantir des délais maximums dans l'exécution de ses fonctions. Il est en charge d'exécuter un ensemble de tâches. Une tâche correspond à une fonction donnée du système.

DÉFINITION I.1.2.11
On appelle système critique un système pouvant mettre en danger des vies humaines ou du matériel en cas de défaillance. C'est le cas par exemple des systèmes avioniques ou véhiculaires.

DÉFINITION I.1.2.12
On appelle niveau d'assurance le niveau de certification exigé sur l'ensemble comprenant à la fois les différents composants logiciels et matériels. Il peut s'agir d'une certifiaction en terme de fiabilité ou de temps réel (e.g. certification SIL) ou en terme de sécurité (via les Critères Communs). Dans le cadre des systèmes véhiculaires, on s'appuie sur la norme SIL (Safety Integrity Level) [SS10], graduée de 1 (niveau le plus faible) à 4 (niveau le plus fort). Cette dernière s'instancie, dans le cadre du logiciel embarqué véhiculaire, au travers des règles édictées par l'association MISRA (Motor Industry Software Reliability Association) [Ass13].

DÉFINITION I.1.2.13
Un système est dit sûr si son niveau d'assurance est compatible avec son niveau de criticité. Il possède donc des propriétés de fiabilité et des propriétés de temps réel adéquates pour assurer la protection des vies humaines ou du matériel correspondant à son niveau de criticité. Les problématiques de sécurité telles que définies dans la Section I.1.2 ne font pas partie des propriétés d'un système sûr.

Il existe trois modèles d'exécution temps réel :
- le modèle périodique : les instanciations successives d'une tâche (appelées travail) s'effectuent avec une inter-arrivée constante ;
- le modèle sporadique : les instanciations successives d'une tâche s'effectuent avec une inter-arrivée variable, mais dont la plus petite inter-arrivée est connue et non nulle ;
- le modèle apériodique : il n'y a pas d'instanciation successive d'une tâche.

Dans le cadre de ma thèse, les traitements sont induits par des événements extérieurs asynchrones (correspondant à des flux réseau). En conséquence, je m'appuie sur un modèle sporadique. C'est ce dernier qui est donc considéré par la suite.

Chaque tâche temps réel se définit par les propriétés suivantes :
- un coût d'exécution pire cas, dénoté WCET (Worst Case Execution Time) ;
- une période, déterminant l'inter-arrivée minimum entre deux travaux de la tâche ;
- une échéance, qui peut être :
  - relative : la durée maximum entre l'instanciation du travail et la terminaison de son exécution
  - absolue : l'instant avant lequel la tâche doit s'être complètement exécutée

Un système logiciel temps réel est en charge d'assurer que l'ensemble des travaux terminent leur exécution avant leur échéance de terminaison au plus tard. Pour cela il s'appuie sur des al-

gorithmes d'ordonnancement. Ceux-ci sont en charge d'assurer un ordonnancement des travaux respectueux de l'ensemble de leurs échéances.

I.1.3 Notions et terminologies véhiculaires

I.1.3.1 Généralités sur les réseaux véhiculaires

De nos jours, les véhicules embarquent de plus en plus de matériel et de logiciel. Ces derniers sont impliqués pour l'aide à la conduite (ABS, ESP, etc.) dans le cadre d'un réseau de bord. Ce réseau est en réalité une interconnexion d'un grand nombre de réseaux basés par exemple sur des bus série de type CAN [Iso], agrégeant l'ensemble des éléments d'aide à la conduite. Le réseau de bord est principalement basé sur des automates matériels avec une faible présence du logiciel. Ce réseau possède de fortes contraintes temps réel et de fiabilité, sans considération particulière en terme de sécurité.

Le réseau de bord n'est aujourd'hui plus le seul réseau présent dans les véhicules. Un deuxième réseau a fait son apparition au fur et à mesure des années. Il s'agit du réseau multimédia. A l'inverse du réseau de bord, ce dernier possède de plus faibles exigences de temps réel, et de plus fortes exigences de sécurité. Les différentes fonctions qu'il porte n'impactent pas la sûreté de fonctionnement du véhicule tant que le réseau de bord et le réseau multimédia ne sont pas interconnectés. Cette sûreté découle entre autres du besoin de respect des latences pire cas des traitements critiques. Néanmoins, les services multimédia fournis par le véhicule nécessitent d'être protégés d'éléments logiciels extérieurs non maîtrisés. Ainsi, le système multimédia du véhicule doit être protégé contre les terminaux mobiles s'y connectant, comme les téléphones portables ou les tablettes.

Dans le cadre du besoin véhiculaire militaire, le réseau multimédia fournit principalement des mécanismes de communication radio entre véhicules (au sein d'un groupe) et entre le véhicule et sa base de commandement. Il fournit également des mécanismes de supervision locaux, comme la gestion des caméras extérieures et/ou intérieures ou du système d'armes si ce dernier existe. Le réseau d'interconnexion du véhicule est riche et hétérogène. En effet, ce dernier s'appuie sur différents types d'interconnexions (courte portée, longue portée, satellitaire, etc.). Ces différents réseaux sont utilisés pour un grand nombre de transmissions. Il peut s'agir de la réception des ordres de mission, de communications phoniques ou visiophoniques, de la réception ou de l'émission de positions GPS. Le véhicule possède donc un véritable système systronique.

I.1.3.2 Les systèmes systroniques

Avec l'intégration de plus en plus forte des réseaux IP dans les systèmes embarqués (véhiculaires, avioniques non temps réel, transport), l'interconnexion de réseaux locaux initialement

disjoints avec un ou des réseau(x) d'infrastructure est de plus en plus courante. Les exigences sécuritaires qui sont alors associées à une telle interconnexion peuvent être fortes, selon le type de données transmises.
L'interconnexion des réseaux locaux est poussée par différents besoins. Dans le cadre des systèmes embarqués véhiculaires, le besoin initial, toujours fort aujourd'hui et dont la résolution reste complexe, correspond à l'intégration sur un même matériel de plusieurs réseaux indépendants. Il s'agit d'un partage, sur un unique matériel pour des raisons d'exigences de volume, de poids ou de consommation, de plusieurs traitements logiques dont le cloisonnement doit être garanti. Ce cloisonnement est exigé pour des raisons de sûreté afin que le partage du matériel ne vienne pas impacter le bon fonctionnement des environnements logiciels.

De plus, dans les systèmes embarqués militaires, la colocalisation impacte des réseaux dont la confidentialité, l'intégrité et la disponibilité des données peuvent être différentes. Ces derniers nécessitent de plus d'être interconnectés. Chacun de ces réseaux ayant des contraintes de sécurité différentes, ils appartiennent à des domaines de sécurité disjoints. En conséquence, il doit exister des mécanismes permettant d'interconnecter ceux-ci de manière sécurisée, afin de valider l'ensemble des flux passant d'un domaine à l'autre. La colocalisation de ces domaines sur un même matériel implique que ces mécanismes soient logiques et non physiques.
Le besoin d'interconnexion de domaines de sécurité a des réponses dans les systèmes d'informations, au travers de l'usage de chiffreurs, de filtres et d'éléments matériels en coupure [Sec12]. Ces solutions restent aujourd'hui difficiles à intégrer sur les systèmes embarqués, du fait de cette colocalisation. En effet, il s'agit la plupart du temps de solutions impliquant plusieurs éléments matériels disjoints, principalement sur des architectures de type x86 et sont en général intégrables sous forme d'une lame format 1U ou supérieur (format classique des *appliances*, pour permettre une bonne intégration dans les baies).

De manière générale, on définit les propriétés de sécurité d'un domaine de sécurité comme l'association de contraintes de confidentialité, d'intégrité et de disponibilité. Dans le cadre d'un système systronique, la confidentialité et l'intégrité des données émises ou reçues sont prépondérantes, mais la disponibilité n'est pas en reste, avec des exigences de maintien des communications potentiellement très fortes malgré l'hostilité du milieu de déploiement (désert, jungle, etc.).
Par exemple, le réseau de communication avec la coalition et le réseau de communication avec le pays possèdent des exigences de confidentialité différentes. Certaines informations pouvant être des données exclusives au pays. Ainsi, un système systronique possède plusieurs domaines de sécurité, dont le nombre varie selon le véhicule et la mission.

Un véhicule militaire possède également des exigences de temps réel. C'est le cas pour la gestion des caméras, du blindage réactif ou encore de la désactivation du système d'armes lors

de l'ouverture d'une porte extérieure.
Ces différents éléments ont des exigences de temps réel plus ou moins nombreuses. Afin de limiter le coût de certification et les contraintes physiques des éléments de la chaîne de traitement temps réel, le calcul du WCET de celle-ci est plus ou moins pessimiste.
Si le calcul du coût d'exécution pire cas est faiblement pessimiste, il peut arriver, avec une faible occurrence, que le coût d'exécution dépasse le coût d'exécution pire cas considéré dans le design du système. Si le calcul est fortement pessimiste, le dépassement n'est en principe pas possible. Dans ce dernier cas cependant, le coût de certification mais aussi la puissance des éléments matériels embarqués sont impactés. La puissance nécessaire au respect strict des exigences temps réel peut être alors incompatible avec des contraintes physiques liées au véhicule (volume, température et consommation).
Afin de pouvoir répondre aux différentes exigences liées au besoin véhiculaire, nous considérons différentes familles de traitements :

- les traitements à très forte exigence de sûreté ou de sécurité, dont la latence d'exécution doit être à tout prix maintenue (système d'armes, blindage transparent) ;
- les traitements à forte exigence de sûreté ou de sécurité, dont la latence d'exécution doit être respectée au mieux (caméras extérieures, communications avec les autres véhicules).

Les deux familles sont donc exécutées de manière conjointe, sauf si la famille de plus forte priorité génère une charge supérieure à celle qui lui est initialement allouée. Le système n'étant plus capable d'exécuter les deux familles conjointement, il devient nécessaire de modifier son comportement afin de garantir malgré tout l'échéance des travaux les plus critiques. On parle dans ce cas d'un système à criticité mixte, comme décrit dans le Chapitre II.3.2. Un tel système permet alors de respecter au mieux à la fois les exigences de sécurité et de sûreté de fonctionnement, mais aussi les exigences physiques qui découlent de l'environnement véhiculaire.

I.1.4 Enjeux et limitations de l'existant

I.1.4.1 Les enjeux des systèmes systroniques

Comme il a été vu précédemment, les enjeux dans les systèmes systroniques sont importants, du fait de l'intégration continue de nouveaux systèmes de communication, tant en terme de liens de communication locaux (interne au véhicule), semi-locaux (i.e. entre des véhicules d'une même patrouille) et longue distance (entre la patrouille et les quartiers généraux) qu'en terme de traitements graphiques et vidéos. Il en résulte un système complexe interconnectant des éléments :

- fortement temps réel (devant garantir l'échéance de l'ensemble des tâches) ;
- temps réel souple (pouvant autoriser des dépassements d'échéance si ceux-ci sont ni trop nombreux ni consécutifs) ;
- non temps réel (sans principe d'échéance).

Aux contraintes temporelles auxquelles répondent les modèles temps réel s'ajoutent des contraintes de fiabilité. Elles sont présentes sur la grande majorité des éléments d'un système systronique, du fait de l'impact de ce système sur les vies humaines. Ainsi, il est impératif que le système d'armes soit fiable.

De manière orthogonale, ces mêmes éléments impliquent des contraintes de sécurité variables, se déclinant en plusieurs domaines de sécurité.
L'interconnexion de tous ces domaines (tant de sécurité que temps réel) est néanmoins demandée. En effet, il est nécessaire dans un véhicule muni d'un système systronique de :

- cloisonner de manière efficace les domaines de sécurité et de sûreté ;
- transférer de l'information désensibilisée d'un domaine de haut niveau de sécurité à un domaine de moindre niveau de sécurité ;
- remonter de l'information d'un domaine de plus bas niveau de sécurité à un domaine de plus haut niveau de sécurité ;
- remonter des données d'un domaine temps réel dur avec de fortes contraintes de sûreté vers un domaine de plus faible niveau de sûreté, avec des contraintes de sécurité plus fortes.

Ces interconnexions sont principalement la conséquence de l'apparition de terminaux de contrôles locaux et portatifs agrégeant un grand nombre de données. Ces terminaux doivent être compatibles avec des contraintes physiques fortes (poids, volume, consommation électrique) et avec des contraintes de certification fortes (principalement liées au temps réel mais aussi, par exemple, à la résistance aux vibrations dans les véhicules). Le coût associé à la production de tels produits est donc élevé. Ces derniers ne doivent alors pas être multipliés pour des raisons de maîtrise des coûts. La séparation des domaines de sécurité et parfois de sûreté n'est donc pas toujours physique. On s'appuie alors sur une séparation logique de ces domaines. Il revient alors au logiciel d'assurer cette séparation. Il lui incombe également de fournir des mécanismes d'interconnexion respectueux de l'ensemble des exigences de sécurité et de sûreté.

I.1.4.2 Limitations de l'existant

Aujourd'hui, les systèmes systroniques maintiennent trois réseaux complètement indépendants :

- le système de bord, en charge des éléments d'aide à la conduite et devant respecter des contraintes temps réel et de fiabilité ;
- le système d'arme, ayant ces mêmes propriétés ;
- le système de communication, sans propriété de sûreté mais avec un grand nombre de contraintes de sécurité.

Il est demandé de pouvoir interconnecter ces trois systèmes afin de remonter des informations du système de bord au système de communication, mais également du système d'armes au système de communication.
Il est de plus demandé une modification profonde du système de communication afin de permettre un partage de certains moyens de communication semi-locaux et distants entre différents domaines de sécurité. En effet, la multiplication des antennes génère à la fois un surcoût et une problématique de qualité de signal radio liée aux interférences mutuelles des antennes.
De plus, du fait de l'incompatibilité des exigences de performance avec les exigences physiques, l'intégration de solutions aptes à permettre des traitements de criticité multiple est nécessaire au niveau de ces éléments d'interconnexion entre ces différents domaines. Ces solutions ne sont aujourd'hui pas au catalogue des industriels, et aucun produit existant ne permet ce type de comportement.

Pour la sûreté de fonctionnement, l'existant correspond souvent aux avancées technologiques qui ont été faites dans l'avionique, telles que l'IMA (Integrated Modular Architecture), permettant une répartition optimisée des fonctions pour assurer le respect des contraintes temps réel. Les niveaux élevés d'assurance (DO-178B pour le logiciel et DO-254 pour le matériel) assurent également une grande fiabilité. Ces solutions répondent donc parfaitement aux besoins de sûreté de fonctionnement. Elles sont alors tout à fait à leur place dans le réseau de bord. Néanmoins, lorsque sûreté et sécurité doivent être associés, ces solutions atteignent leurs limites. Il devient nécessaire de concevoir des mécanismes logiques (dans le cas des systèmes systroniques) permettant de respecter à la fois des besoins de sûreté et de sécurité.

Pour la sécurité, l'existant est principalement basé sur les travaux intégrés depuis des années dans la sécurisation des systèmes d'informations complexes. Malheureusement, ces solutions s'appuient principalement sur des éléments matériels positionnés en coupure entre chaque domaine de sécurité et sur la cryptographie.
Les éléments en coupure sont malheureusement difficilement intégrables. En effet :

- ils sont souvent volumineux et consommateurs de courant ;
- ils ne sont pas certifiables pour les systèmes contraints avec des exigences physiques comme la résistance aux vibrations ;
- leur facteur de forme n'est pas prévu pour de l'embarqué.

Les solutions de cryptographie ne s'intègrent pas toujours aisément :

- dans les systèmes de communication vehiculaires, la cryptographie est faite au niveau de l'émetteur/récepteur radio plutôt que dans le réseau. De plus, elle est souvent associée à des mécanismes d'évasion de fréquence afin de réduire encore plus le risque d'écoute ;
- pour la gestion des domaines de sécurité locaux, les chiffreurs utilisés pour les systèmes d'information sont volumineux, lourds et consommateurs en énergie. Ils doivent alors

être substitués par un algorithme de chiffrement étatique logiciel ou matériel de type carte cryptographique de petite taille intégrable en début de mission ;
– les mécanismes d'interconnexion assurant à la fois des exigences de sûreté et de sécurité fortes sont inexistants, du fait de la jeunesse de ce besoin, tant dans le domaine véhiculaire que dans celui de l'avionique.

I.1.5 Synthèse et éléments de solution

On constate qu'il devient aujourd'hui difficile d'échapper au besoin d'un système communicant intégrant de multiples niveaux de sûreté et de sécurité interagissant de concert. Pour permettre une interconnexion de ces niveaux tout en assurant un cloisonnement suffisant pour ne pas impacter les exigences initiales de chacun d'entre eux, de nouveaux mécanismes doivent être étudiés afin de pouvoir répondre à l'ensemble des exigences des systèmes systroniques. Les exigences que doivent respecter ces solutions sont de trois types :
– sûreté de fonctionnement, principalement en terme de garantie de latence, mais également en terme de protection du domaine de plus haut niveau de sûreté ;
– sécurité, pour garantir la confidentialité, l'intégrité et la disponibilité des données des différents domaines de sécurité lorsque ces derniers nécessitent d'être interconnectés ;
– physique, afin de garantir que le logiciel est suffisamment peu coûteux à l'exécution pour respecter les exigences de consommation.

Une telle solution doit pouvoir se plier à ces trois familles d'exigences afin de ne pas multiplier les produits devant être intégrés au système systronique. C'est un besoin qui dérive à la fois des exigences physiques et des exigences financières, le coût des véhicules devant être maîtrisé.

L'ensemble des exigences décrites dans ce chapitre sont synthétisées sous forme d'une matrice d'exigences dans le Chapitre III.1. C'est au travers de la conformité à chacune des exigences de cette matrice qu'est décrite ma solution.

Dans le cadre de ma thèse, je propose une solution logicielle pour répondre à la problématique d'interconnexion des réseaux véhiculaires dans le cadre d'un système systronique. Le besoin de modularité est pris en compte afin de rendre la solution configurable pour répondre aux différents besoins d'interconnexion de réseaux, et plusieurs contraintes de sécurité et de sûreté sont considérées afin de répondre à plusieurs exigences précises :
– garantie de transfert de données à sens unique ;
– filtrage des données selon une politique de sécurité ;
– maîtrise du profil des flux transférés ;
– réduction des risques de transfert de données non homologués ;

– prise en compte dans le transfert de flux des différences de criticité des flux à faire transiter et des traitements à effectuer sur ces derniers.

Dans le cadre de ma thèse, la solution logicielle est prévue pour être utilisée en tant que passerelle, tant entre plusieurs domaines de sûreté qu'entre plusieurs domaines de sécurité. Les différents modules proposés ont une faible empreinte et une forte autonomie, afin de permettre une certification du logiciel.

La passerelle étant l'élément en coupure entre les différents domaines de sécurité, celle-ci doit garantir une maîtrise complète des données qui transitent. La passerelle est potentiellement connectée à deux domaines de sécurité en même temps. Cela implique que différentes parties de son logiciel s'exécutent dans différents domaines de sécurité. Une telle architecture implique une étude plus poussée du cloisonnement entre les deux éléments logiciels. Ce cloisonnement s'effectue à plusieurs niveaux :

– via l'usage d'une architecture de type MLS (MultiLevel Security), intégrant des éléments logiciels de confiance en coupure entre ces deux domaines. Ces éléments logiciels doivent être certifiables et performants, et sont donc définis comme des éléments simples devant répondre à une seule fonction. Dans ma thèse, j'appelle ces éléments des moniteurs de sécurité ;
– via la prise en compte de la problématique de canaux cachés et canaux auxiliaires, mécanismes permettant de faire transiter de l'information via un élément souvent matériel au travers d'une étude comportementale. Le matériel ciblé est souvent le contrôleur de cache, qui permet, via des mesures de temps d'exécution, de récupérer de l'information sur le comportement d'un autre logiciel exécuté sur le même CPU. Afin de répondre à cette problématique, nous avons proposé une solution [Thi+11] permettant de mieux contrôler le comportement du cache L1, afin de contrecarrer cette attaque.

La passerelle doit également pouvoir répondre à des exigences de garantie de temps de latence maximum. L'usage d'une architecture de type MLS a pour conséquence l'usage d'un ordonnancement de type hiérarchique, sur base TDM (Time Division Multiplexing). Afin de définir une méthode pour déterminer l'ordonnançabilité de tâches temps réel sur une base d'ordonnancement TDM, nous avons formalisé les propriétés nécessaires au respect des latences [TGH11] permettant de définir la taille des slots temporels du TDM. Cette formalisation est décrite dans le Chapitre III.3.1.2.

Les principales exigences des systèmes systroniques sont des exigences de poids, de volume et de consommation. Leur impact principal est la limitation de la puissance processeur embarquée, impactant potentiellement l'ordonnançabilité de l'ensemble des tâches lorsque les calculs de coûts d'exécution sont pessimistes. Afin de pouvoir intégrer à la fois des tâches fortement critiques et des tâches plus faiblement critiques, il faut prévoir, dans le cadre de l'intégration

de l'ensemble des tâches, de potentiels débordements de la part des tâches très critiques. On s'appuie sur un ordonnancement à criticité mixte. Afin de pouvoir gérer au mieux cette problématique, une étude a été faite [San+12] afin de :

- réduire les contraintes du changement de niveau de criticité, afin de limiter le nombre de tâches devant être désactivées ;
- déporter au maximum le changement de niveau de criticité, au travers de l'intégration de la marge de tolérance [BGM07] sur les durées de dépassement des WCET ;
- formaliser l'instant où le niveau de criticité peut être réduit.

La problématique de changement de niveau de criticité et de réduction de ce dernier est étudiée dans le Chapitre III.3.3.

Afin de valider la solution générale et intégrer plusieurs moniteurs de sécurité, un maquettage a été réalisé et des mesures de performance effectuées [TGL13]. Trois moniteurs de sécurité ont été réalisés :

- Une diode logicielle garantissant un transfert de données à sens unique
- Un filtre DPI (Deep Packet Inspection) pour valider le contenu des données en transit
- Un émetteur périodique, afin de décorréler le profil de flux reçu du profil de flux émis, pour éviter un canal caché sur l'étude de ce profil.

La fonction diode a également été implémentée dans le cadre d'un preuve de concept répondant à l'ensemble des exigences sécuritaires (architecture MLS, compartimentation forte, usage d'un ordonnancement TDM, nettoyage du cache lors du changement de domaine). Afin de valider la preuve de concept, une CSPN (Cible de Sécurité de Premier Niveau) a été faite, associée à un dépôt de brevet [Thi+13a][Thi+13b], suivie d'une pré-évaluation EAL5+ du micro-noyau sous-jacent.

Chapitre I.2

Concepts

Sommaire

I.2.1 Principes généraux de la sécurité **17**
I.2.1.1 Terminologie militaire de la sécurité 17
I.2.1.2 Fuite d'informations : Canaux cachés et canaux auxiliaires 20
I.2.1.3 Ce que cible la sécurité logicielle 21
I.2.1.4 Cibles de sécurité et certifiabilité 21
I.2.2 Principes généraux des systèmes temps réel **22**
I.2.2.1 Rappels sur les politiques d'ordonnancement 22
I.2.3 Principes généraux de la virtualisation **23**
I.2.3.1 Concepts liés à la virtualisation et la compartimentation 23
I.2.3.2 Les solutions de virtualisation type I 24
I.2.3.3 Les solutions de virtualisation type II 26
I.2.3.4 Les solutions de virtualisation applicative 26
I.2.4 Les interactions entre le matériel, la sécurité et le temps réel **27**
I.2.4.1 Principes de base de la MMU 27
I.2.4.2 A propos des contrôleurs de cache 31
I.2.4.3 Les algorithmes de gestion de caches 37

I.2.1 Principes généraux de la sécurité

I.2.1.1 Terminologie militaire de la sécurité

Les niveaux de confidentialité

Il existe un grand nombre de niveaux de confidentialité formalisés dans le vocabulaire militaire. Pour la France, on parle des niveaux suivants, du plus faible au plus élevé :

- Diffusion Restreinte (DR). Bien que très employé, c'est le seul niveau pour lequel aucune peine n'est définie dans le code pénal en cas de non-respect des exigences de confidentialité associées.
- Confidentiel Défense (CD). Premier niveau impliquant un risque pénal. Ce niveau peut être présent jusque dans les véhicules blindés positionnés sur le théâtre d'opération.
- Secret Défense (SD). Ce niveau est en général présent jusque dans les centres de contrôle militaires locaux au théâtre d'opération. Il s'agit des centres de contrôle militaires déployés dans les pays en guerre dans les bâtiments civils protégés (type aéroport) ou dans des bâtiments militaires proches de la zone de conflit (frégate, porte-avions).
- Très secret défense (TSD). Plus haut niveau de sécurité.

Il existe de plus un grand nombre de niveaux de confidentialité liés à l'OTAN (Organisation du Traité de l'Atlantique Nord, NATO en anglais), ou à diverses organisations impliquant une force armée. Il existe ainsi les niveaux Nato Restricted (NR), Nato Confidential (NC), Nato Secret (NS) et encore bien d'autres niveaux. Il existe une problématique claire de comparaison des niveaux France avec les niveaux OTAN. Ainsi, peut-on transférer des données du niveau NS au niveau SD ? Cette problématique est encore aujourd'hui traitée au cas par cas.

Lorsqu'un niveau de confidentialité est mis en place, c'est au travers d'un système d'information respectueux d'une politique de sécurité donnée. Cette politique de sécurité (nommée PSSI - Politique de Sécurité d'un Système d'Information) définit des exigences de sécurité pour assurer la confidentialité, l'intégrité et la disponibilité (au sens sécuritaire du terme) du domaine. Ces exigences sont mises en place sous forme :

- d'un choix d'architecture réseau, logicielle et matérielle ;
- de la définition d'algorithmes de chiffrements spécifiques et de solutions implémentant ces algorithmes ;
- d'une définition d'un processus de gestion du domaine (protection physique et logique) ;
- d'une définition des responsabilités et des rôles de chaque intervenant dans le domaine.

Plus le niveau de confidentialité est élevé, plus la PSSI est stricte et complexe.

Changement de niveau et problématique rouge-noir

La terminologie dite *rouge/noir* est une terminologie militaire, définie ci-après :

DÉFINITION I.2.1.1
On appelle domaine rouge un domaine de sécurité dans lequel les données transitant ne sont pas protégées. On parle alors de données rouges. Le réseau faisant transiter ces données dans un domaine rouge est alors appelé réseau rouge.

DÉFINITION I.2.1.2
On appelle domaine noir un domaine de sécurité dans lequel les données transitant sont modifiées pour en assurer la confidentialité, l'intégrité et la disponibilité. On dit alors que les données sont noircies.

DÉFINITION I.2.1.3
Un système est dit rouge-noir lorsqu'il interconnecte un domaine rouge à un domaine noir. Une telle interconnexion implique un élément de confiance en bordure des domaines, en charge d'intégrer un mécanisme de protection des données du domaine rouge avant de les transmettre au domaine noir. Cet élément de confiance est également en charge de valider toute transmission venant du domaine noir vers le domaine rouge.

En général, un domaine noir est en charge de faire transiter des données dont le niveau de sécurité est supérieur à celui du domaine lui-même. Ainsi, faire transiter des données de niveau Confidentiel Défense dans un domaine de niveau Diffusion Restreinte ne peut être fait que si ces données sont préalablement noircies.
Noircir des données implique en général un chiffrement de ces dernières, mais pas seulement. Le profil de flux peut être modifié afin de rendre difficile la détection du type de communication sur la base de son profil [Ber+05]. Du bruit peut ainsi être généré et du *padding* incorporé afin d'empêcher toute déduction sur simple écoute du flux noir.

La problématique de passage d'un domaine rouge à un domaine noir est appelée problématique rouge/noir. On la retrouve principalement dans les réseaux radio militaires, mais elle reste un principe général applicable à toute interconnexion de réseaux militaires.

Niveaux de confidentialité et droit d'en connaître

DÉFINITION I.2.1.4
On appelle besoin d'en connaître l'ensemble de connaissances minimum suffisant pour un sujet (une personne ou encore un processus logiciel) pour mener à bien une tâche.

DÉFINITION I.2.1.5
On appelle droit d'en connaître l'ensemble de connaissances formalisé dans une politique de gestion des droits afin de répondre au(x) besoin(s) d'en connaître d'un sujet.

Ainsi, une personne ayant une habilitation Confidentiel Défense sur un projet donné n'a pas besoin pour autant de connaître l'ensemble des informations classifiées Confidentiel Défense de ce dernier. On lui fournit alors uniquement les informations nécessaires à son travail, appelées droit d'en connaître. Le principe est le même pour les processus logiciels faisant du traitement sur des données d'un niveau de confidentialité donné.

I.2.1.2 Fuite d'informations : Canaux cachés et canaux auxiliaires

Assurer la confidentialité est une tâche complexe. En effet, limiter l'accès à la donnée n'est pas suffisant pour assurer sa protection. Ainsi, il est possible de récupérer des informations sur la donnée de plusieurs manières, sans y accéder directement. On parle alors de canal auxiliaire ou de canal caché, selon la méthode.

Canal auxiliaire et étude comportementale

DÉFINITION I.2.1.6
On appelle canal auxiliaire un élément comportemental lié au fonctionnement de la solution permettant à un attaquant, de manière non-invasive (e.g. via une mesure de temps), de déterminer des informations sur un élément auquel il n'a pas les droits d'accès.

La plupart des études sur le sujet sont dans le domaine de la cryptanalyse [DS09][OST06]. Ces différentes études démontrent la capacité, au travers de l'étude de la consommation électrique des différents éléments d'un processeur [Ger06][DS09] ou via l'étude de l'impact des caches processeurs sur le comportement temporel de l'algorithmique [OST06], de récupérer des données sur la clef de chiffrement utilisée.
Le risque lié au canal auxiliaire est connu depuis longtemps et a permis de démontrer que des algorithmes cryptographiques réputés sûrs restent fragiles à ce type d'attaque. C'est le cas par exemple d'AES [DR98] ou de Serpent [ABK98].

Canal caché pour le transfert de données

DÉFINITION I.2.1.7
On appelle canal caché tout élément logiciel ou matériel dont il est possible de détourner l'usage afin de pouvoir faire transiter volontairement de l'information vers un sujet n'ayant pas les droits d'accès à celle-ci.

L'usage d'un canal caché implique un usage volontaire de part et d'autre pour émettre les données et pour les recevoir. Il est donc invasif, contrairement aux canaux auxiliaires.
Un exemple typique est de s'appuyer sur la durée intertrame minimale d'un flux réseau donné pour fournir une information [Ber+05]. En faisant varier la durée intertrame, il est possible d'émettre de l'information, qui peut être ainsi interprétée par un récepteur. De manière générale, les canaux cachés permettent l'émission de données de faible taille, car leur débit est en général très faible. Ils sont cependant tout à fait suffisants pour faire transiter des données de type cryptographie (clefs symétriques et/ou asymétriques) ou des messages courts.
Le principe d'exploitation d'un canal caché est également utilisé dans les systèmes virtualisés

pour transmettre de l'information entre les machines virtuelles. C'est en général le comportement des contrôleurs de caches qui sont alors exploités [Xu+11][Zha+12].

Lorsqu'un transfert de données non autorisé a lieu entre deux domaines de sécurité, il est important de pouvoir en imputer la responsabilité. On parle alors d'imputabilité.

DÉFINITION I.2.1.8
On appelle imputabilité la garantie de traçabilité entre une action effectuée sur le système et l'identité de l'utilisateur responsable de cette action.

Du fait des contraintes légales liées aux niveaux de confidentialité, le besoin d'imputabilité est omniprésent dans les systèmes militaires.

I.2.1.3 Ce que cible la sécurité logicielle

Dans leur globalité, les architectures de sécurité sont là pour répondre à ces trois besoins initiaux. Selon le niveau de certifiabilité, les architectures de sécurité peuvent prendre plusieurs formes, allant d'une séparation matérielle complète et stricte jusqu'à la simple intégration d'une politique de gestion de contrôle d'accès (propriétaire, groupe, droit d'accès) aux fichiers, comme par exemple les systèmes DAC (Discretionary Access Control), MAC (Mandatory Access Control) ou RBAC (Role-based Access Control) [SS94].

I.2.1.4 Cibles de sécurité et certifiabilité

La complexité du système dépend du besoin de certifiabilité. On définit initialement une Cible de Sécurité (Security Target).

DÉFINITION I.2.1.9
On appelle Security Target (ST, CdS en français) le document qui définit de manière complète et rigoureuse le problème sécuritaire associant un produit.

A partir de la Cible de Sécurité, on définit une cible d'évaluation (couramment appelée Target Of Evaluation), ainsi que le niveau d'assurance à atteindre.

DÉFINITION I.2.1.10
On appelle Target Of Evaluation (TOE) l'ensemble des éléments logiciels et matériels qui seront évalués dans le cadre d'une certification, en fonction du niveau d'assurance attendu, au sens des Critères Communs.

Le niveau de sécurité du TOE est donc dépendant du niveau souhaité. Les systèmes Linux bien configurés en terme de sécurité peuvent atteindre un niveau de certification de l'ordre de l'EAL 4. Il faut cependant prendre soin d'étudier précisément quelle est la TOE. En effet, le niveau de sécurité d'un système certifié dans le cadre d'un fonctionnement autonome n'est plus assuré si ce dernier est connecté à un réseau.

Associé à la TOE, on définit le socle de confiance couramment appelé Trusted Computing Base.

DÉFINITION I.2.1.11
On appelle Trusted Computing Base (TCB) l'ensemble des éléments logiciels et matériels de la solution cible qui sont considérés comme de confiance et dont la corruption génère une faille dans les propriétés de sécurité de la solution.

Ainsi les éléments de la TCB doivent non seulement être certifiables au plus haut niveau souhaité dans le cadre de l'évaluation du système, mais doivent être également protégés contre toute corruption extérieure (comme par exemple à partir d'un élément non-certifiable). On leur impose alors une exigence d'*inviolabilité*, comme décrite dans les exigences générales des systèmes MILS et MLS. Ces dernières sont rappelées dans le Chapitre II.1.2

Lorsque l'on s'appuie sur une architecture de type compartimentation stricte de fonctions exécutées de manière autonome (comme par exemple via des technologies de virtualisation), la solution peut être compatible avec le principe de certification composite, permettant une certification par delta.

DÉFINITION I.2.1.12
On appelle certification par composite un principe permettant, lorsque la TCB est non modifiée, de s'appuyer sur les précédentes certifications afin de refaire une certification de l'ensemble à moindre coût Cela permet de faire d'incrémenter la certification d'un produit.

Les systèmes compartimentés peuvent intégrer des compartiments certifiables implémentant un algorithme simple répondant à un besoin précis. Lorsque ce besoin est sécuritaire, il s'agit d'une fonction de sécurité.

DÉFINITION I.2.1.13
On appelle fonction de sécurité un traitement logiciel spécialisé dans la réponse à un besoin sécuritaire donné. Une fonction de sécurité peut être non-certifiable (par exemple en tant que processus logiciel d'un système d'exploitation riche) ou certifiable (sous forme d'un élément logiciel de petite taille et autonome ou s'appuyant sur un socle lui-même certifiable).

I.2.2 Principes généraux des systèmes temps réel

I.2.2.1 Rappels sur les politiques d'ordonnancement

Il existe différents ordonnanceurs temps réel. Ces derniers sont divisés en deux grandes familles d'ordonnanceurs, les ordonnanceurs en ligne et hors ligne.

L'ordonnancement en ligne Les choix d'ordonnancement se font durant l'exécution, en fonction de la liste des travaux présents et de leurs paramètres (par exemple en fonction de leurs échéances absolues).

L'ordonnancement hors ligne La politique d'ordonnancement est définie par avance. Elle est donc optimisée pour un cas particulier de jeu de tâches, mais n'est pas réactive en cas d'impact non prévu sur la dynamique d'exécution (effets de caches, de surcharge de bus, WCET trop grand ou trop petit, etc.).

Parmi tous ces ordonnanceurs, certains peuvent préempter un travail en cours d'exécution avant sa terminaison. On parle alors d'ordonnancement préemptif. A l'inverse, si un travail en cours d'exécution ne peut être arrêté avant sa terminaison, on parle d'ordonnancement non-préemptif. La préemption peut être liée à une gestion du temps gérée en interne par l'ordonnanceur (time-driven) ou événementiel (event-driven).
Dans le cadre de ma thèse, je cible deux ordonnanceurs :

- L'ordonnancement TDM (hors ligne, préemptif de type time-driven)
- L'ordonnancement EDF (en ligne, préemptif de type event-driven)

I.2.3 Principes généraux de la virtualisation

Il existe plusieurs types de virtualisation, en fonction de l'architecture logicielle et parfois matérielle qui entre en compte dans la solution.

I.2.3.1 Concepts liés à la virtualisation et la compartimentation

DÉFINITION I.2.3.1
On appelle machine virtuelle (ou VM, Virtual Machine) une implémentation logicielle d'une machine (i.e. d'un ordinateur) qui exécute des programmes comme une machine physique. Il existe deux familles de machines virtuelles :

- *une machine virtuelle système, représentant une plateforme matérielle complète et qui supporte l'exécution d'un système d'exploitation ;*
- *une machine virtuelle de processus, dont le but est d'exécuter un seul programme et qui permet d'assurer la portabilité de ce dernier. C'est par exemple le cas de la machine virtuelle Java.*

DÉFINITION I.2.3.2
On appelle Moniteur de Machine Virtuelle [PG74] (ou VMM, Virtual Machine Monitor) le composant logiciel en charge de permettre l'exécution d'une ou plusieurs machines virtuelles.

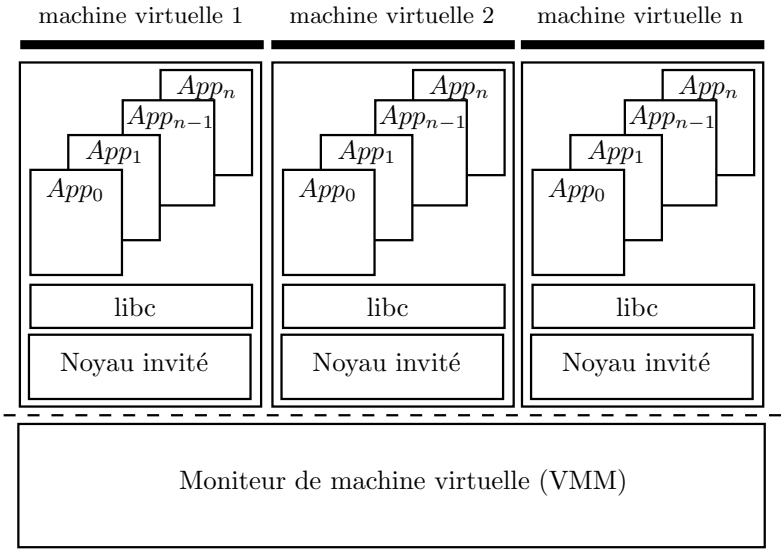

FIGURE I.2.1 – Architecture logicielle générique des solutions de virtualisation de type I

DÉFINITION I.2.3.3
On appelle Environnement Virtuel (ou VE, Virtual Environment) un compartiment logiciel en charge d'exécuter un environnement logiciel complet, à l'exception du noyau. C'est le cas par exemple des technologies OpenVZ ou LXC [Bha+11]. Une telle architecture logicielle ne nécessite pas de moniteur de machine virtuelle (ou VMM).

I.2.3.2 Les solutions de virtualisation type I

Les solutions de virtualisation de type I s'appuient sur un moniteur de machine virtuelle autonome, s'exécutant directement sur le matériel. Il est seul maître de l'environnement logiciel de la machine et assure l'ordonnancement de ces dernières. La Figure I.2.1 décrit l'architecture logicielle de ce type de solution.
Les interactions entre le moniteur de machine virtuelle et les machines virtuelles peuvent varier selon la solution utilisée pour assurer la bonne exécution de l'environnement invité (i.e. l'environnement logiciel de la machine virtuelle) et plus particulièrement du noyau invité.
Pour répondre à ce besoin, il existe trois solutions :

- La virtualisation complète
- La paravirtualisation
- La virtualisation matérielle

Principe de la virtualisation complète

Historiquement la plus ancienne, elle fut formalisée sous forme d'exigences par G.J. Popek et R.P. Goldberg [PG74] dès 1974. Le principe est de pouvoir exécuter un environnement logiciel dans une machine virtuelle tout en assurant que son comportement soit identique à une exé-

cution native. L'environnement invité n'est pas modifié, et c'est au travers des exceptions processeur dues à l'exécution en tant qu'environnement invité que le moniteur de machine virtuelle réagit à des traitements qui ne peuvent être exécutés directement par l'environnement invité sans impact sur la sécurité et la sûreté du système.
Dans leur Article [PG74], G.J. Popek et R.P. Goldberg décrivent les incompatibilités du matériel avec ce type de principe. En effet, seules les instructions générant des exceptions processeur peuvent être substituées par le moniteur de machine virtuelle. Les instructions sensibles [PG74] ne générant pas d'exception sont une problématique forte car ces dernières peuvent s'exécuter sans que le moniteur de machine virtuelle en soit informé. La conséquence peut être plus ou moins grave car ces dernières impactent la sûreté et la sécurité du système.

Afin de pouvoir virtualiser des environnements logiciels sans l'aide du matériel et sans modifier l'environnement à virtualiser, plusieurs solutions ont été mises en place, entre autres par VMWare. Dans ses solutions, ce dernier est capable de déterminer les instructions sensibles qui seront demandées par l'environnement invité et de modifier ces dernières afin d'injecter des traitements simulés en lieu et place. Ce type de solution est transparent pour l'environnement invité, mais génère des variations dans le coût d'exécution de ces traitements, du fait de la simulation. La mesure de ces variations par le logiciel fait partie des mécanismes lui permettant de détecter que son environnement d'exécution est virtualisé.

Principe de la paravirtualisation

Afin de limiter la problématique de simulation des traitements consécutifs à la détection d'instructions incompatibles avec la virtualisation, une solution a été définie visant à impliquer le noyau de l'environnement invité dans le support de la virtualisation. Le noyau est alors modifié pour substituer les instructions sensibles par des appels volontaires à des fonctions du moniteur de machine virtuelle, appelés hypercalls. Dans ce cas, le moniteur réagit soit en exécutant le traitement initial avec les droits qui sont les siens, soit en simulant le traitement.

Il existe un certain nombre de solutions implémentant les mécanismes de paravirtualisation. C'est par exemple le cas de Xen ou encore de la solution propriétaire PikeOS de Sysgo. La limitation du principe de virtualisation est due à la nécessité d'adaptation du noyau invité à l'Application Programming Interface (API) du moniteur de machine virtuelle. Cette API n'étant pas normalisée, cela implique une implémentation souvent spécifique à la solution de virtualisation utilisée. Cela implique également d'avoir accès au code source du noyau invité, limitant ainsi les choix en terme d'environnements paravirtualisables.

Principe de la virtualisation matérielle

Définie initialement par IBM, puis reprise depuis le début des années 2000 par Intel et AMD, la virtualisation matérielle consiste à intégrer dans le matériel des aides à la virtualisation. Le but est de limiter la complexité du moniteur de machine virtuelle, ce dernier reposant sur des instructions privilégiées de configuration du matériel pour gérer des environnements matériels d'exécution des machines virtuelles séparés. Un exemple de solution est l'architecture VT-d d'Intel, proposant des structures de configuration et d'hébergement matérielles nommées VMCS (Virtual Machine Control Structure) qui dupliquent les registres sensibles du processeur. Lorsque le moniteur de machine virtuelle ordonnance l'environnement virtualisé, il indique au processeur le VMCS à utiliser en lieu et place des registres utilisés par le moniteur lui-même. La solution s'appuie également sur des éléments de filtrage d'accès aux périphériques (nommés I/OMMU pour Intel, SMMU pour ARM Cortex A15, A53 et A57). Il n'existe à ce jour aucune norme et chaque fondeur possède ses propres instructions.
Cette solution n'est pas considérée ici car incompatible avec les exigences de portabilité.

I.2.3.3 Les solutions de virtualisation type II

Les solutions de virtualisation type II s'appuient sur la présence d'un hôte qui accueille le moniteur de machine virtuelle comme un processus. Ce dernier est donc lui-même ordonnancé par l'hôte. Ce type de solution de virtualisation correspond à des produits tels que VMWare Workstation ou encore VirtualBox. La Figure I.2.2 décrit l'architecture logicielle d'une telle solution.

Ces solutions étant basées sur des hôtes de type GNU/Linux ou Windows dans lesquels elles s'exécutent comme des tâches non temps réel, elles sont donc incompatibles avec les contraintes de temps réel, y compris lorsque ces dernières sont relativement souples. Elles sont également incompatibles des exigences de sécurité forte, du fait de la non-certifiabilité du système hôte.

Les solutions de moniteurs de machine virtuelle utilisant ce type d'architecture ne permettent pas d'assurer un cloisonnement spatial fort (répartition stricte en mémoire physique) ni un cloisonnement temporel fort (l'ordonnancement des machines virtuelles étant assujetti aux propriétés de l'ordonnanceur de l'environnement hôte).

I.2.3.4 Les solutions de virtualisation applicative

Les solutions de virtualisation applicative sont parentes des solutions de virtualisation de type II dans le sens où elles s'appuient sur un hôte. Cependant, celles-ci ne nécessitent pas la présence de noyau invité. Elles permettent d'instancier des Environnements Virtuels dont le noyau est partagé entre ces derniers ainsi qu'avec l'hôte. L'ordonnancement des tâches des VEs

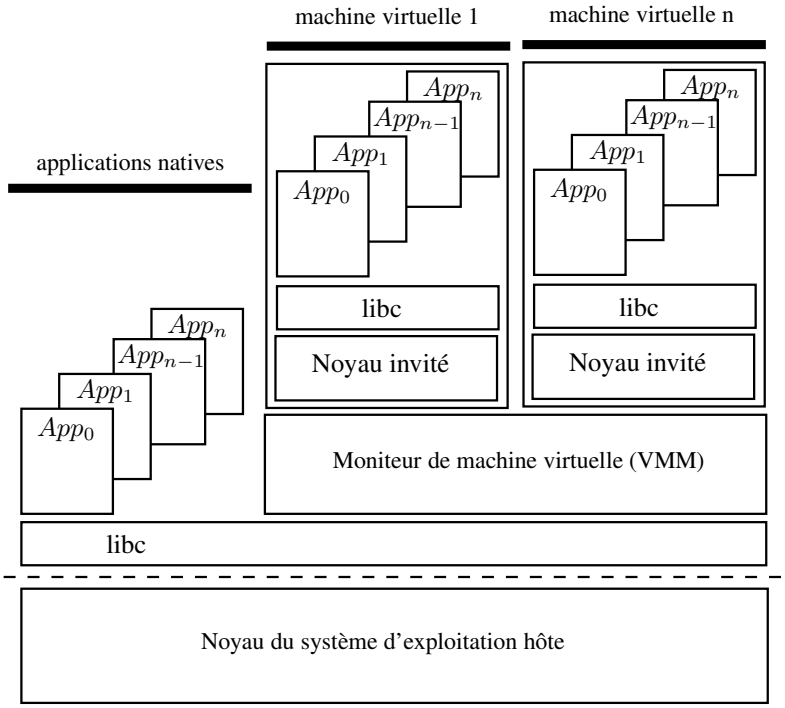

FIGURE I.2.2 – Architecture logicielle d'une solution de virtualisation type II

est géré directement par l'ordonnanceur de l'hôte, sous forme d'un groupe de tâches autonome. Ce type d'ordonnancement se rapproche ainsi du principe d'ordonnancement hiérarchique qui sera décrit dans le Chapitre II.3.1.
Un exemple de solution de virtualisation applicative est la solution LXC (LinuX Containers) de Linux. La Figure I.2.3 décrit l'architecture logicielle d'une telle solution.

Les solutions de virtualisation applicatives permettent de créer des VEs dont le système de fichiers, le jeu de tâches, les interfaces réseau et la vision du stockage de masse sont séparés.

I.2.4 Les interactions entre le matériel, la sécurité et le temps réel

I.2.4.1 Principes de base de la MMU

Un applicatif, construit dans un fichier binaire lors de l'édition de lien, possède un certain nombre de sections servant de conteneur à des données différentes, et ayant en conséquence des propriétés de sécurité (en terme d'accès) également différentes. De ce fait, lors du déploiement de ces différentes sections en mémoire vive, il est nécessaire de considérer pour chacune de ces sections des propriétés d'accès spécifiques.

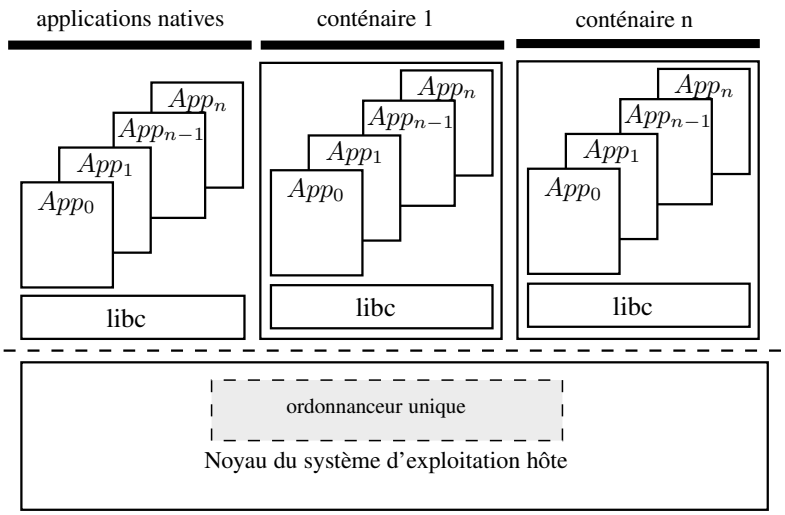

FIGURE I.2.3 – Architecture logicielle d'une solution de virtualisation applicative

Dans un fichier binaire au format ELF [Com95], il existe plusieurs sections, dont entre autres :

- la section `.text`, contenant la partie exécutable du programme
- la section `.rodata`, contenant les données en lecture seule (e.g. les chaînes de caractères)
- la section `.data`, contenant les données en lecture et écriture

Lorsque plusieurs applicatifs s'exécutent en concurrence, il est important de les cloisonner dans des environnements logiques séparés, ceci afin d'éviter que ces derniers puissent interagir sans contrôle, par simple accès à la mémoire physique.

Il existe plusieurs niveaux d'abstraction de la mémoire physique :

- Sans aucune abstraction, on est en mode dit *flat memory*. L'ensemble des logiciels s'exécutant en mémoire traitent des adresses physiques. Ainsi, ils connaissent leur position en mémoire, et ont une visibilité sur l'ensemble de la mémoire physique.
  De ce fait, il leur est possible d'accéder au contenu des autres applicatifs en cours d'exécution. Il devient ainsi aisé d'espionner les sections d'autres logiciels, comme les sections .data ou .rodata, dans lesquelles il est possible de récupérer par exemple des mots de passe.
  De plus, l'accès à la pile des autres applicatifs peut permettre de dériver leur exécution vers un bloc de code spécifique, en remplaçant l'adresse de la fonction parente par une adresse dont on maîtrise le contenu. Du fait que la pile est par essence accessible en écriture, tous les processus peuvent modifier les champs d'un processus donné en mode *flat memory*, à partir du moment où l'adresse physique de sa pile est connue.

- Décomposition de la mémoire physique en segments. Il s'agit du mode de segmentation présent uniquement sur les processeurs de la famille x86. Ce mode étant aujourd'hui remplacé par le système de pagination, il ne sera pas décrit ici.
- En s'appuyant sur une abstraction de la mémoire physique, appelé *mémoire virtuelle*. On parle alors de mode dit de *pagination mémoire*. Un processus donné possède son propre environnement mémoire appelé Espace d'adressage (AS pour Address Space), dont la taille correspond à la valeur maximum d'un mot mémoire (e.g. dans un processeur 32 bits, l'environnement mémoire virtuelle à une taille de quatre giga-octets).
  Le processus, lors de son exécution, ne traîte alors que des adresses virtuelles, dont la translation en adresses physiques se fait au travers d'un élément matériel nommé Memory Management Unit (MMU).

La gestion de la pagination est gérée par la MMU, sous les ordres du noyau du système d'exploitation. Ce dernier construit l'association entre mémoire physique et mémoire virtuelle dans des structures en mémoire nommées tables de pages (Pages Tables). Sur les architectures x86, ces dernières sont accédées par la MMU via un registre nommé PDBR. Ce dernier pointe vers la racine de l'arbre de mapping de mémoire virtuelle du processus en cours, et s'appuie sur ce dernier pour retrouver l'adresse mémoire physique associée à l'adresse mémoire virtuelle demandée par le cœur du processeur. Afin d'optimiser les accès à cet arbre de page, la MMU s'appuie sur un cache local nommé TLB (Translation Lookaside Buffer), réduisant ainsi les requêtes au contrôleur mémoire.

Le principe de fonctionnement de la pagination fournit plusieurs avantages :

- A chaque processus est associé un AS (espace d'adressage) qui lui est propre. Ainsi, il se voit seul en mémoire et toute tentative d'accès à la mémoire qui ne lui est pas officiellement fournie sera bloquée par la MMU. En effet, pour qu'un processus y accède, une association entre adresse physique et adresse virtuelle, locale à cet espace d'adressage doit avoir été faite par le noyau au préalable, via configuration de la MMU. Lors de la création d'un processus, seules les adresses physiques correspondant à ces propres données sont intégrées à l'espace d'adressage, interdisant à ce dernier d'accéder aux données de tout autre processus.
- De plus, la gestion de la fragmentation de la mémoire centrale, dépendant des allocations et désallocations successives de la mémoire, est rendue opaque en mémoire virtuelle. Des espaces mémoires physiques disjoints peuvent être positionnés de manière contigüe dans l'espace d'adressage virtuel, faisant croire au processus que ses données sont contigües sans pour autant qu'elles le soient.
- Le contrôleur MMU permet d'associer à chaque page mémoire des propriétés en terme d'accès :
  - Niveau d'exécution minimum (mode administrateur ou utilisateur)

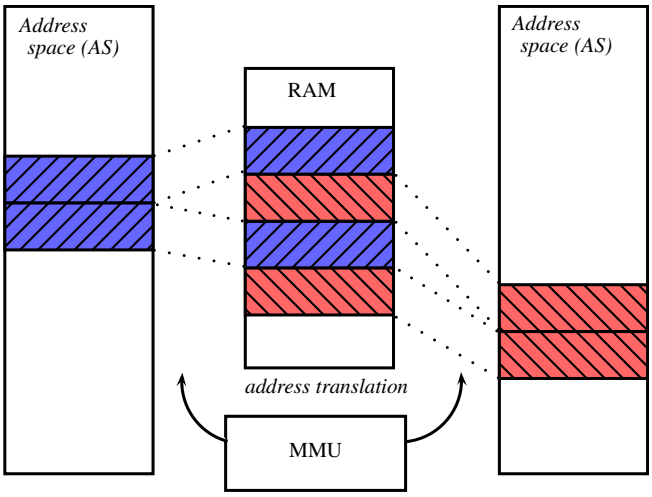

FIGURE I.2.4 – Principe général du mapping de page mémoire dans les espaces d'adressage des processus

– Droits d'accès (lecture, écriture et exécution)
  Ces droits d'accès permettent, dans un espace d'adressage donné, de limiter les accès à chaque page. Ainsi, les sections .rodata ou .text du processus ne sont pas accessibles en écriture. Une telle protection réduit la surface d'attaque, rendant plus difficile la corruption de l'application.

La Figure I.2.4 décrit le principe général de pagination entre mémoire physique et mémoire virtuelle.

Les attaques entre les différents processus ne sont pas les seules possibles. Il est également possible de corrompre le comportement d'un processus donné, pour lui faire faire un traitement non prévu.
Une des attaques les plus classiques est l'injection de *shellcode*. Un shellcode est un morceau de code binaire intégré sous forme d'une chaîne de caractère. Il est utilisé afin d'être intégré quelque part dans l'espace d'adressage du processus, avant de corrompre sa pile pour provoquer un branchement vers ce bloc d'instructions.
Cette attaque est donc faite en deux temps, en positionnant des instructions en mémoire puis en forçant le branchement. En général, les shellcodes sont placés dans la pile, en amont du positionnement courant du pointeur de pile. Il convient, pour éviter ce type d'attaque, de rendre l'espace mémoire utilisé pour la pile non-exécutable. A l'inverse, l'espace mémoire où se situe le code du processus doit lui être exécutable, mais ne doit pas être accessible en écriture, ce dernier étant immuable durant l'exécution du processus.
On définit ici des droits d'accès à des espaces mémoire (lecture, écriture, exécution) variants selon le type de données qui y sont hébergées. Les MMU modernes, lors de la création d'une

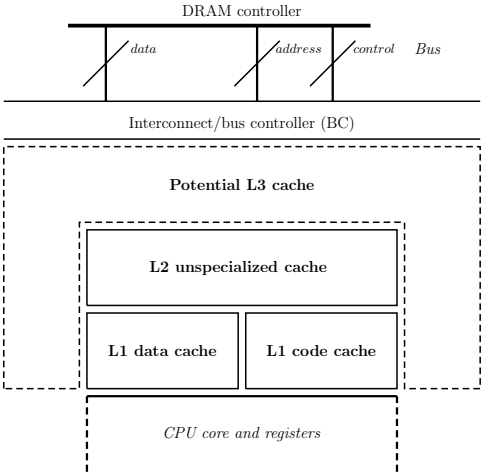

FIGURE I.2.5 – Architecture en couche générique

association entre mémoire physique et mémoire virtuelle, supportent ces trois propriétés, fournissant un durcissement supplémentaire pour réduire les risques de corruption d'un processus.

I.2.4.2 A propos des contrôleurs de cache

Description générale

Le cache est une mémoire statique à accès rapide, permettant d'accélérer les accès mémoire du processeur, en copiant les données (code et data) de manière à ce qu'elles soient accessibles directement par ce dernier sans nécessité de transfert sur le bus [Wik10].

La figure I.2.5 montre une architecture mémoire générique, avec plusieurs niveaux de cache, suivis d'un bus d'accès mémoire et du contrôleur mémoire.

Il existe dans les processeurs modernes plusieurs niveaux de cache :

1. Le cache niveau 1, le plus petit et accessible le plus rapidement. Ce cache est spécialisé : il définit un conteneur dit *cache code*, pour les éléments de code, et un conteneur dit *cache data*, pour le reste des données.
2. Le cache niveau 2, de taille plus conséquente, mais dont l'accès est moins rapide que le cache L1. Ce cache n'est pas spécialisé.
3. Le cache niveau 3, de taille encore plus grande. Ce dernier est en général partagé entre les cœurs sur les architectures multi-cœurs. Ce cache n'est pas spécialisé et n'est pas toujours présent.

Niveau de mémoire	Temps d'accès
Cache L1	2-8 nano secondes
Cache L2 (partagé)	5-12 nano secondes
Mémoire centrale	10-60 nano secondes

TABLE I.2.1 – Temps d'accès mémoire sur un Intel Core 2 duo

Le tableau I.2.1 [1] fournit un exemple de comparaison du temps d'accès à une donnée selon son positionnement en mémoire.

Les algorithmes de cache ont divers niveaux d'associativité :

- Les algorithmes de type *one-way associative* associent une cellule mémoire à une et une seule cellule de cache. Il s'agit d'un algorithme surjectif, du fait de la taille du cache. Le choix de remplacement étant de type $1:1$, aucun algorithme spécifique n'est utilisé. Les données sont écrites au seul emplacement possible en cache.
- Les algorithmes de type *N-way associative* associent une cellule mémoire avec N (resp. 2, 4, 8, ...) cellules de cache. Cette associativité permet d'optimiser l'écrasement des cellules du cache en donnant un choix de positionnement à la cellule mémoire à mettre en cache. Les algorithmes de ce type se différencient par leur politique de choix de la cellule à remplacer.

On a ainsi :

***N*-Way Set Associative** Cet algorithme permet, pour chaque élément mémoire à charger en cache, de pouvoir se placer à N endroits dans le cache. Ces N positions sont dépendantes de l'adresse physique en mémoire centrale de la donnée.
Le choix de la cellule parmi la liste des cellules admissibles se fait au travers d'un algorithme de gestion de cache tel que décrit plus loin.

Direct Mapped L'algorithme Direct Mapped est un algorithme simple, optimisant le coût d'exécution du choix de la cellule. Un élément mémoire n'est placé qu'à un seul endroit dans le cache, en fonction de son adresse physique, quelle que soit la donnée précédemment présente.

Lorsqu'une donnée n'est pas en cache, on parle de *cache-miss*. Le processeur va successivement vérifier les caches L1, L2 et L3 (si présent), avant d'initier une recopie à partir de la mémoire centrale.

Il existe plusieurs algorithmes de gestion du cache, souvent de type probabiliste. De récents travaux de recherche tendent à montrer que ces caches peuvent être pris en compte dans les

1. Confer http://expertester.wordpress.com/2008/05/30/core-2-duo-vs-pentium-dual-core/

architectures temps réel [Gua+12]. Cependant, les systèmes temps réel très contraints d'aujourd'hui, comme dans l'aviation ou le spatial, utilisent encore beaucoup d'architectures avec caches désactivés.

Positionnement du cache

Cache look-aside Cette architecture place le cache et la mémoire centrale côte à côte, comme le montre la partie gauche de la Figure I.2.6. Ainsi, le cœur du processeur peut accéder à la mémoire centrale sans passer par le cache.
Le principe de fonctionnement est le suivant :

- Le processeur démarre un cycle de lecture
- Si le cache possède la donnée en local (cache hit), il répond au processeur, et clôture le cycle de lecture sur le bus.
- Si le cache ne possède pas la donnée (cache miss), il laisse la mémoire centrale répondre au processeur, et espionne les données transitant sur le bus afin de se mettre à jour.

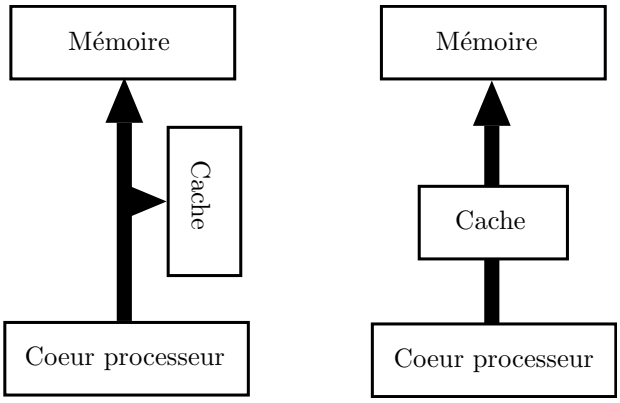

FIGURE I.2.6 – Comparaison des architectures de types cache through et cache lookaside

Cache look-through Cette architecture place le cache en coupure entre le cœur du processeur et la mémoire centrale, comme le montre la partie droite de la Figure I.2.6. Le processeur passe donc par le cache pour l'accès mémoire.
Le principe est le suivant :

- Le processeur démarre un cycle de lecture.
- Si le cache possède la donnée en local (cache hit), il répond au processeur.
- Si le cache ne possède pas la donnée, il transfère la requête de lecture sur le bus. La mémoire centrale renvoie la donnée, qui est mise en cache en même temps qu'elle est renvoyée au processeur.

Cette architecture provoque des cache-miss plus longs (vérification par le cache avant de faire la requête au contrôleur de bus), mais permet d'utiliser le cache indépendamment des autres cœurs du processeur (pour un cache local).

Consistance du cache

Du fait que le cache contienne des duplicatas d'une partie de la mémoire, il est important de s'assurer que la copie et l'originale restent identiques. À défaut, cela peut entraîner des erreurs logicielles difficilement détectables.

La consistance du cache peut être impactée dans différentes situations, comme par exemple :

- Mise à jour de la mémoire par un contrôleur DMA
- Écriture en mémoire par un autre processeur (architecture multi-CPU/multi-cœurs)
- Modification de la donnée en cache sur un autre processeur

Il existe différentes méthodes permettant de garantir la consistance des données en cache, comme par exemple :

Snooping Le snooping consiste en l'espionnage du bus d'adresse, afin de détecter une modification sur une adresse mémoire contenue en cache.
Il y a deux manières de réagir à une détection positive :

- Prévoir un rechargement de la cellule impactée (méthode dite *write-back*). Le rechargement ne se fait pas nécessairement sur le moment, mais peut être ordonnancé pour s'exécuter au moment du prochain accès à cette cellule, ou avant, selon les implémentations.
- Désactiver la cellule impactée. Le prochain accès aux données de cette cellule provoquera un cache-miss, suivi d'une recopie à partir de la mémoire centrale.

Snarfing Le snarfing consiste à associer la détection d'une modification d'une donnée en cache à l'espionnage du bus de donnée. En espionnant le bus, il est capable à tout instant de mettre à jour la cellule impactée et de maintenir la consistance.

Synchronisation entre le cache et la mémoire centrale

Lorsque des données sont modifiées en local sur le cache, il est nécessaire de mettre à jour la mémoire centrale afin de prendre en compte ce changement.

Write-back Le cache garde en mémoire, pour chaque cellule, son état. Lorsque la cellule est écrasée, ou que le cache est vidé, les données modifiées localement sont synchronisées avec la mémoire centrale.
La mise à jour de la mémoire centrale n'est pas synchrone avec la modification locale. Cela

implique, au niveau logiciel, de prévoir les problèmes d'incohérence sur les architectures multi-cœurs ou multi-processeurs, au travers de l'usage des *memory barriers*. Ces dernières ont pour but de fournir des mécanismes de synchronisation entre les différents cœurs afin d'assurer la cohérence d'une donnée partagée entre plusieurs instances logicielles exécutées en parallèle. L'implémentation des barrières mémoires est dépendante de l'architecture matérielle.

Write-through Une écriture sur cache implique une remontée synchrone de la mise à jour de la donnée, afin de maintenir une cohérence entre le contenu du cache et la mémoire.

Les algorithmes de gestion de cache

Least Recently Used L'algorithme LRU s'appuie sur les dates d'accès aux items du cache. En cas de cache-miss, c'est l'élément qui a été accédé il y a le plus longtemps qui est remplacé.

Most Recently Used L'algorithme MRU, à l'opposé du LRU, remplace l'élément dont l'accès est le plus récent. Il considère l'hypothèse selon laquelle un élément accédé très récemment ne le sera plus avant une certaine durée.

Pseudo-LRU L'algorithme Pseudo-LRU utilise un arbre binaire représentant l'ensemble des cellules de caches pouvant être remplacées. Chaque ensemble de cellules possède un tag fournissant une indication sur l'emplacement (branche droite, branche gauche) de la cellule qui sera remplacée. Ce tag est mis à jour à chaque traversée de l'arbre.

Il s'agit d'un algorithme jugé efficace en comparaison au LRU pour les caches dont l'associativité est supérieure à 4. Il est utilisé dans les architectures Intel 486, et dans les architectures PowerPC, comme le PowerQuicc de Freescale [Wik09]. Le schéma I.2.7 est tiré du manuel de référence du processeur e300, utilisé dans les PowerQuicc 2 pro.

Segmented LRU L'algorithme SLRU utilise deux segments de caches complémentaires, dont les cellules sont ordonnées de la dernière utilisée à la plus anciennement utilisée :

- Le segment probatoire (*probationary segment*), qui est celui sur lequel sont mises les données mémoire, en cas de cache miss. C'est également le seul segment où des données peuvent être retirées du cache.
- Le segment protégé (*protected segment*), qui contient la liste des cellules de cache ayant été accédées au moins deux fois.

Le schéma I.2.8 montre ainsi que l'algorithme de cache fait une différence entre des données accédées à faible fréquence et des données accédées régulièrement.

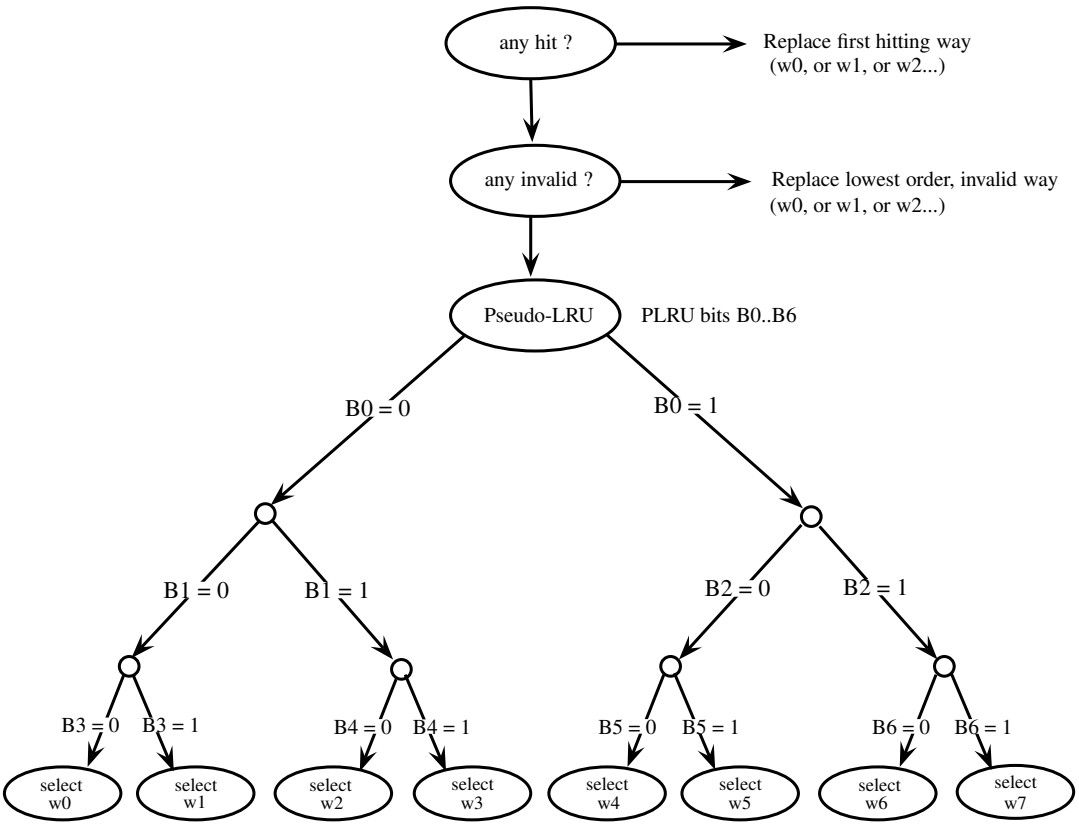

FIGURE I.2.7 – Automate à état de l'algorithme pseudo-LRU du Power Quicc 2 Pro

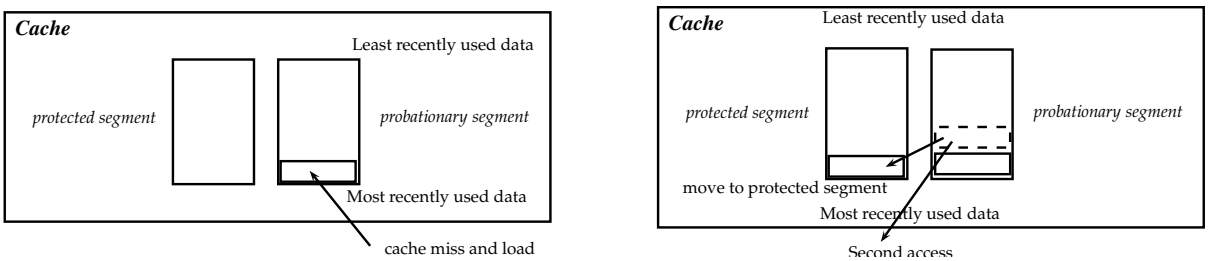

FIGURE I.2.8 – Intégration et déplacement des données accédées fréquemment

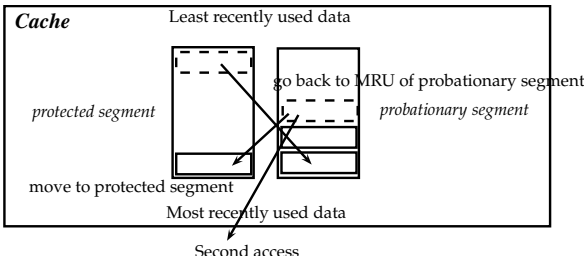

FIGURE I.2.9 – Principe de deuxième chance du SLRU

En effet, lorsqu'une cellule est accédée une seconde fois après un premier cache miss, celle-ci est transférée sur le segment protégé, en tête de liste.
Cela a pour conséquence, si le segment protégé est plein, de faire descendre la cellule la plus anciennement accédée sur le segment probatoire. Elle est alors positionnée en tête de liste, ceci afin de donner une deuxième chance à toute donnée accédée au moins deux fois. Lorsque le segment probatoire est plein et que de nouvelles données y sont intégrées, que ce soit via un déplacement d'une cellule du segment protégé ou du fait d'un cache miss, la cellule la moins récemment utilisée est écrasée.

Cet algorithme fait donc une différence entre les données souvent accédées (au moins deux fois) et les données rarement accédées (une fois) pendant leur durée de vie en cache. Il permet donc d'optimiser la gestion du cache lors d'accès mémoire peu fréquents.
La taille des deux segments est dépendante dynamiquement du schéma d'entrée/sortie.

Least Frequently Used Lors d'un cache miss, la cellule utilisée pour charger les données mémoire est celle dont le contenu est utilisé le moins fréquemment. En cas d'égalité de fréquence, l'algorithme LFU est utilisé pour départager les cellules.

I.2.4.3 Les algorithmes de gestion de caches

Les caches et la sécurité

L'impact des caches processeurs est non nul sur la sécurisation d'un système lorsque deux domaines de sécurité sont ordonnancés sur un même matériel. Cela est encore plus vrai si ces deux domaines sont ordonnancés sur un même cœur du processeur.
En effet, la modification de l'état des caches du processeur est induite par le comportement des différentes applications en cours d'exécution, générant un canal auxiliaire entre ces dernières, comme décrit plus loin.

Dans [Per05], l'auteur a ainsi démontré qu'il était possible, au travers de mesures des coûts d'accès mémoire d'une tâche, de détecter l'impact d'une autre tâche exécutée en concurrence,

dans le cadre d'un ordonnancement préemptif. Cet impact étant variant selon le comportement de celle-ci, il est alors possible d'en déduire certaines informations sur l'arbre d'exécution associé, et donc sur les choix qui ont été effectués par cette tâche. Une telle information est utile si l'attaquant a connaissance du code source de la tâche cible, afin d'associer le chemin d'exécution détecté à une succession d'algorithmes connus. Cette hypothèse est aujourd'hui plus réaliste dans le monde de l'open-source. Un tel mécanisme a été montré fonctionnel dans [Ber05][BM06] et dans [Ban11], où les auteurs démontrent la capacité pour une tâche à récupérer des informations sur les données d'entrée d'algorithmes cryptographiques utilisées par une tâche concurrente, grâce à l'étude du chemin d'exécution détecté. Dans [Ber05], la démonstration a été effectuée dans le cadre de deux tâches exécutées de manière simultanée sur un processeur Pentium 4 supportant l'hyperthreading. Dans [Ban11], l'auteur n'impose pas l'usage simultané du cœur processeur. L'espionnage se fait au travers de l'étude de son propre comportement en terme de mesure de temps d'accès mémoire, induit par le comportement de la tâche concurrente (grâce à la mesure des caches-miss imprévus).
De manière générale, l'espionnage des algorithmes cryptographiques au travers des collisions des lignes de caches est facilité par leur implémentation. En effet, ils s'appuyent sur des tables de taille fixe présentes en mémoire à des emplacements disjoints. A ce jour, les clefs de chiffrement restent donc un élément confidentiel difficile à protéger si tout ou partie de l'architecture logicielle est corrompue.

Aujourd'hui, lorsque deux compartiments s'exécutent en concurrence sur un même cœur processeur, il est d'usage de flusher et réécrire le contenu des cellules des caches à chaque préemption des compartiments. On considère ainsi qu'un compartiment donné ne protège pas ses propres tâches entre elles, mais que deux compartiments concurrents doivent être ordonnancés avec un cache dont le contenu est connu et prédictible.
Avec une telle architecture logicielle, la fonction de sécurité correspondant au flush de cache est considérée suffisante. Cette dernière peut en effet être bornée dans le temps, et garantit le contenu des cellules du cache lorsque le compartiment reprend son exécution.

Les caches et le temps réel

Les caches processeurs sont également impactant dans le cadre du respect des contraintes temps réel. Dans les systèmes industriels très exigeants, les caches sont souvent désactivés. Malheureusement, une telle configuration est de plus en plus difficile à mettre en œuvre. En effet, les architectures matérielles et logicielles sont de plus en plus construites pour être performantes grâce aux contrôleurs de cache (grâce à la présence de deux voire trois niveaux de cache, ou encore de mécanismes de synchronisation avancée dans le cadre des architectures MPSoC, etc.). La désactivation des caches génère un ralentissement difficile à accepter selon les hypothèses de performances initiales.

Il existe un grand nombre de travaux sur la prise en compte des contrôleurs de caches dans le temps réel. Ces travaux sont séparables en deux grandes familles :

- Prise en compte des besoins temps réel dans le design du contrôleur de caches et optimisation des cache-miss (cache locks et cache partitioning)
- Calcul du nombre pire cas de cache-miss au travers de l'étude du chemin d'exécution des différentes tâches en exécution concurrente, avec ou sans l'aide du contrôleur de cache

Des travaux existent sur l'usage des méthodes de cache-locking afin de limiter le nombre de cache-miss [CIBM01][Pua06], grâce au blocage de certaines lignes de cache dans le contrôleur, afin d'en assurer la présence lors d'un accès ultérieur. Il faut alors choisir judicieusement les lignes de cache devant être maintenues, sans pour autant réduire trop fortement le nombre de lignes de cache pouvant être remplacées pour ne pas impacter trop fortement l'exécution des autres tâches.

Lorsque les contrôleurs de caches ne permettent pas un tel contrôle par le logiciel, il faut alors étudier leur comportement en terme d'algorithme de remplacement de ligne de cache, afin de déterminer, en fonction du chemin d'exécution de la tâche courante, le taux de cache-miss associé. Le but est dans ce cas d'intégrer au mieux l'impact positif de la présence du cache du processeur sur l'exécution de la tâche.
Ainsi, dans [Gua+12], les auteurs ont montré qu'il était possible de prendre en compte l'impact d'un contrôleur de cache s'appuyant sur l'algorithme MRU dans le calcul du WCET d'une tâche. Ils ont ainsi montré qu'il était possible de déterminer, sous certaines conditions, que le maintien de certaines lignes de caches dans le contrôleur pouvait être prédictible, réduisant ainsi fortement le pessimisme dans la mesure du WCET.

État de l'art des problématiques temps réel dans le support MMU

Les contrôleurs de caches des CPU ont un impact reconnu sur le respect des contraintes temps réel d'un système. Il en va de même pour la TLB (Translation Lookaside Buffer).
Le TLB est le cache spécialisé permettant d'optimiser la traduction d'adresse que la MMU effectue entre l'adressage logique (tel que vu par le cœur processeur) et l'adressage physique (tel que vu au niveau du bus d'interconnexion). La présence du TLB découle du fait que les données de traduction sont gérées en mémoire centrale, induisant des accès mémoire fréquents par le contrôleur MMU afin de faire l'association mémoire logique/mémoire physique. Le TLB est donc un cache local permettant de garder en mémoire les dernières traductions d'adresses afin d'éviter une surconsommation des accès mémoire centrale par le contrôleur MMU.
Le contenu du TLB est impacté par l'ordonnancement de tâches, et doit donc être considéré dans le cadre de la gestion de tâches avec des contraintes de temps réel dur. Dans [GK08] les auteurs décrivent un mécanisme apparenté aux cache-locking afin de limiter l'impact du TLB sur le coût d'exécution des tâches temps réel.

Synthèse

L'usage des caches est aujourd'hui nécessaire afin de pouvoir exploiter correctement les capacités des processeurs modernes. Cependant, leur usage impacte à la fois la sécurité et le temps réel. Dans le cadre de ma thèse, je ne propose pas de design matériel de contrôleur de cache. Néanmoins, je propose un axe de solution, décrit dans le Chapitre III.2.4, afin de limiter l'impact des contraintes de sécurité sur les performances et le temps réel. Plus généralement, je considère la problématique sécuritaire associée à la présence d'entités logicielles de domaines de sécurité hétérogènes sur une même architecture matérielle, et son impact sur la gestion des caches.

Dans le cadre de la définition de la solution finale, je m'appuie néanmoins, sur demande de Thales, sur du matériel existant de type PowerPC pour des raisons de certifiabilité. En effet, les architectures PowerPC sont celles ayant le moins de bugs matériels référencés parmi les grandes familles (ARM, PowerPC, x86 et Sparc). Elles ont comme bonne propriété d'allier une consommation raisonnable à de bonnes performances. Aujourd'hui, les contrôleurs de caches utilisés par les cœurs PowerQuicc sont de type Pseudo-LRU, comme vu plus haut. Les cœurs e500mc, succédant aux cœurs e300 des PowerQuicc 2, fournissent de plus plusieurs propriétés intéressantes, dans le cadre des architectures QorIQ de Freescale [Fre08] :

- La capacité de dériver une partie du contrôleur de cache L2 en mémoire SRAM utilisable par le logiciel.
- Le support du *secure boot*, qui permet, via l'injection d'une clef de chiffrement dans une mémoire au travers de fusibles à usage unique, d'assurer l'intégrité des différents éléments de la chaîne de boot (bootloader, puis noyau du système d'exploitation).
- La présence de PAMU (Peripherical Access Management Unit) dont le but est de permettre au logiciel d'injecter une politique de filtrage des accès des différents composants matériels initiateurs sur le bus.

Les mécanismes de secure boot et de PAMU, du fait de leur récente intégration et pour des raisons de confidentialité liés aux travaux de Thales autour de leur usage, ne sont pas étudiés dans ma thèse.

Deuxième partie

État de l’art des solutions de sécurité et de temps réel

Chapitre II.1

État de l'art de l'interconnexion sécurisée de domaines de sécurité hétérogènes

Sommaire

II.1.1 Sécurisation par séparation physique des traitements **43**

II.1.1.1 Interconnexion sécurisée par passerelle de type diode matérielle . 43

II.1.1.2 Firewall d'interconnexion de domaines de sécurité hétérogènes . . 45

II.1.2 Architectures MLS (MultiLevel Security) et MILS (Multi-Independent Level of Security) **46**

II.1.2.1 Systèmes MLS, MILS et la haute sécurité 46

II.1.2.2 MLS et principe de virtualisation 48

II.1.3 Interconnexion sécurisée inter-domaines de sécurité **48**

II.1.3.1 Le contrôle de flux et les passerelles de sécurité 48

II.1.4 Synthèse **49**

Il existe un certain nombre de solutions dont le but est de permettre l'interconnexion de domaines de sécurité hétérogènes. Ces solutions répondent plutôt à des besoins de type SI, mais ces solutions restent un très bon point de départ dans le cadre de la définition d'une architecture logicielle sécurisée.

II.1.1 Sécurisation par séparation physique des traitements

II.1.1.1 Interconnexion sécurisée par passerelle de type diode matérielle

Description de la solution

Dans les systèmes militaires, les domaines de sécurité devant être interconnectés le sont au travers de passerelles certifiées s'appuyant sur des éléments physiques disjoints. Celles-ci as-

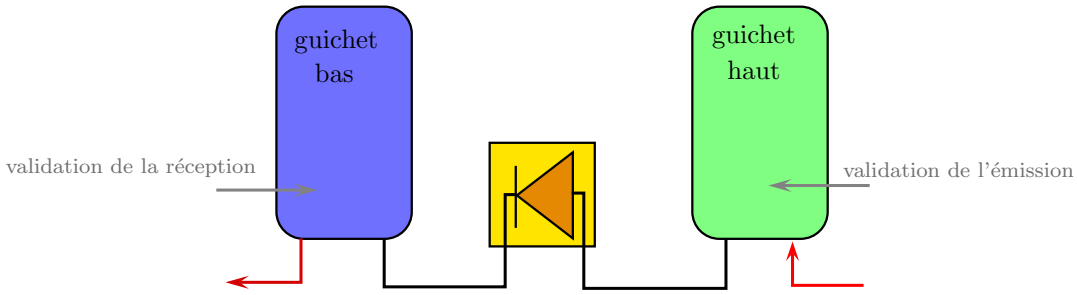

FIGURE II.1.1 – Architecture générale d'une passerelle de type diode matérielle

surent un transfert en sens unique de données entre le domaine de plus basse sécurité (source des données) et le domaine de plus haute sécurité (destination des données). La passerelle de sécurité a deux buts :

- Elle assure le transfert de données en sens unique. Aucune donnée du domaine de sécurité le plus élevé ne peut être transférée vers le domaine de sécurité le plus bas.
- Elle assure une opacité forte du domaine de sécurité le plus élevé d'un point de vue comportemental. Elle évite ainsi les canaux cachés (confer Chapitre I.2.1.2) s'appuyant sur l'étude du comportement du domaine de sécurité le plus élevé. Ces derniers permettraient en effet de remonter indirectement des données confidentielles vers le domaine de sécurité le plus bas.

Afin de décorréler les interfaces de la fonction diode (normalement très simples) de l'architecture réseau de chacun des deux domaines de sécurité, la diode s'appuie sur deux *guichets*.

DÉFINITION II.1.1.1

On appelle guichet un équipement logiciel en charge de transcoder une information d'un réseau à un autre. Il peut par exemple décapsuler une données transmise sur un réseau IP vers un lien point à point. Un guichet peut posséder des propriétés complémentaires, comme la validation de la donnée à transmettre, ou encore le chiffrement de cette dernière.

Ainsi, la fonction diode ne prend plus en compte les propriétés télécom de chacun des deux domaines et conserve une certaine simplicité d'implémentation, comme le décrit la Figure II.1.1 :

- Un *guichet bas*, en charge de récupérer les données potentiellement via une communication en mode connecté avec le domaine de plus basse sécurité pour les transférer à la fonction diode via un protocole spécifique.
- Un *guichet haut*, en charge de récupérer les données sortant de la fonction diode via un protocole spécifique et initiateur d'une connexion potentiellement en mode connecté avec le domaine de plus haute sécurité.

Dans l'architecture de la diode matérielle formalisée par Philippe Lagadec de la Direction Générale de l'Armement (DGA) en 2006 [Lag06], les guichets permettent d'intégrer une action humaine pour valider le transfert de données, ainsi que pour valider leur réception. Une telle architecture permet de garantir l'imputabilité (c'est à dire associe un responsable, confer Définition I.2.1.8) du transfert de données en cas d'usage invalide de la passerelle.

Avantages et limitations

Les diodes matérielles s'appuient sur le principe de séparation protocolaire pour limiter l'exploitation de canaux cachés utilisant des en-têtes protocolaires comme conteneur d'information. En conséquence, elles ne permettent pas d'intégrer le support d'une criticité multiple en terme d'ordonnancement de flux. Cette gestion doit donc être déportée au niveau du guichet haut (émetteur de la donnée). L'intervention humaine, conséquence du besoin d'imputabilité associé au transfert de données, est également problématique lorsque le besoin de temps réel est présent. Comme le décrit l'auteur, la diode matérielle permet cependant d'assurer à la fois :

- la confidentialité des données du récepteur
- l'intégrité et la disponibilité des données de l'émetteur

II.1.1.2 Firewall d'interconnexion de domaines de sécurité hétérogènes

Il existe peu de solutions permettant une interconnexion de domaines de sécurité disjoints. C'est le cas de la solution TSF 101 [Sec12] de Thales Security, permettant de faire transiter des données entre domaines de sécurité disjoints en fonction d'une politique de sécurité chargée dans l'équipement.

Description de la solution

La solution TSF 101 permet de faire transiter des données déclassifiées du domaine de sécurité le plus haut au domaine de sécurité le plus bas. Pour cela, elle s'appuie sur un critère de filtrage prédéfini et non modifiable dans le cadre de son utilisation. La solution est certifiée par les Critères Communs au niveau EAL 5.

Avantages et limitations

La solution permet d'interconnecter des réseaux de niveaux de sensibilité différents. Il s'agit d'un équipement impliquant un facteur de forme (confer Définition I.1.2.4) et des propriétés d'architecture ne lui permettant pas d'être intégré dans des environnements embarqués contraints. La solution ne cible pas non plus de propriétés temps réel, et n'est pas modulaire en terme de services logiciels, le besoin auquel cet équipement répond ne nécessitant pas ces propriétés.

II.1.2 Architectures MLS (MultiLevel Security) et MILS (Multi-Independent Level of Security)

II.1.2.1 Systèmes MLS, MILS et la haute sécurité

Un système MILS (Multiple Independent Level of Security) ou MLS (MultiLevel Security) est une architecture de haute sécurité basée sur le principe de séparation des flux d'information dont l'implémentation est faite dans un composant certifié. Ce dernier assure le cloisonnement de ces flux via un mécanisme évaluable au sens du standard international des Critères Communs [Pro14] pour un coût raisonnable.

Le mécanisme assurant le cloisonnement des différents niveaux de sécurité n'est pas imposé. Il peut être physique (séparation complète), matériel (implémentation d'un mécanisme matériel) ou logiciel (noyau de séparation sécurisé, usuellement nommé *Secure Separation Kernel* ou SSK). Dans le cadre de ma thèse, la solution considérée est le SSK afin d'être compatible avec les exigences de volume, de consommation et de faible coût.

Avant les premiers travaux sur les systèmes MILS (définis par le département de la défense américain au début des années 2000), la problématique de gestion de la sécurité dans les systèmes automatisés était étudiée au travers de principes de listes de contrôle d'accès. En 1973 [BL73], une méthode de gestion de contrôle pour des sujets ayant accès à des objets fut définie et démontrée mathématiquement. Il s'agit du RBAC [FCK95].

En parallèle des travaux de Bell et LaPadula [BL73 ; Bel05], une étude fut effectuée sur la capacité à vérifier formellement les architectures logicielles de sécurité basées sur un noyau [Rus81] (*"kernelized secure systems"*). L'auteur démontre la difficulté de certifier un système où le noyau est lui-même trop complexe. Il décrit alors le principe de noyau de sécurité, devant être de petite taille, et pouvant être accompagné de processus logiciels de confiance, également certifiables. Une telle architecture permet alors d'héberger des processus non certifiables dont le cloisonnement est assuré par le noyau. L'auteur montre que le noyau n'est alors plus sujet d'une politique de sécurité, cette dernière ciblant les processus logiciel qu'il porte. Plus globalement, l'auteur décrit les premiers principes d'un système rouge-noir (confer Définition I.2.1.3). Ce système est basé sur une architecture distribuée de sous-systèmes communiquant de manière sécurisée, en assurant que l'unique moyen de passage est maîtrisé par un élément de confiance permettant la transmission de données noircies (confer Définition I.2.1.2). Ce principe est encore utilisé aujourd'hui dans la définition de systèmes dont l'architecture est distribuée comme la radio logicielle [MM06], où il s'associe à une architecture de type MILS pour séparer les informations.

La définition des architectures MILS et MLS reprend ces travaux afin de définir une nouvelle approche contrastant fortement avec les principes décrits par Bell et LaPadula.
Les systèmes MILS et MLS s'appuient sur des composants ayant de fortes propriétés de sécurité édictées par le département de la défense américain. Ensemble, ces composant forment un système :

PROPRIÉTÉ II.1.2.1
Non contournable *: Un composant ne peut utiliser que le chemin de communication qui lui est fourni par le moniteur de sécurité.*

PROPRIÉTÉ II.1.2.2
Évaluable *: L'ensemble des composants considérés comme de confiance doivent être évaluable au niveau de sécurité demandé.*

L'évaluation de ces composants doit être suffisante pour assurer le niveau de sécurité de l'ensemble du système (i.e. les composants qui ne sont pas de confiance ne doivent pas impacter le niveau de sécurité du système).

PROPRIÉTÉ II.1.2.3
Validation à chaque accès *: Chaque accès ou message entre composants doit pouvoir être validé par un moniteur de sécurité.*

Le moniteur de sécurité peut être le SSK ou un processus de confiance dédié à ce traitement.

PROPRIÉTÉ II.1.2.4
Inviolable *: Le système assure le contrôle des accès en écriture aux moniteurs de sécurité, à leurs données ou à leur configuration, assurant ainsi leur intégrité.*

Un système MultiLevel Security correspond à l'usage d'un système informatique permettant de traiter en parallèle des données de niveaux de sécurité hétérogènes. Il permet l'accès simultané à des utilisateurs ayant des niveaux d'accréditation et de droits d'en connaître (confer Définition I.2.1.5) différents, tout en garantissant un cloisonnement strict entre ces différents utilisateurs. La définition a été formalisée dans le glossaire du Comité National des systèmes de sécurité (CNSSI) [Sch10].

Les architectures MLS ne se limitent pas à la problématique de contrôle des communications dans un environnement multi-niveaux de sécurité sur du matériel partagé. Elles permettent également la communication au travers d'une infrastructure compatible MLS. Dans [DWSHAF05], les auteurs décrivent les mécanismes répartis dans le réseau d'interconnexion permettant de faire communiquer des domaines de sécurité hétérogènes de manière sûre. Les quatre propriétés des systèmes MLS sont alors respectées.

II.1.2.2 MLS et principe de virtualisation

Le principe de MultiLevel Security s'appuie sur un environnement d'exécution de confiance assurant le cloisonnement des compartiments de domaines de sécurité différents. Cet environnement d'exécution est appelé un *Separation Kernel*. Le principe de Separation Kernel a été initialement défini en 1981 [Rus81] et reste aujourd'hui d'actualité, au travers d'un grand nombre de travaux tant universitaires qu'industriels [AF+06 ; Boe+08]

Parmi les mécanismes de séparation des domaines tels que décrits par la NSA dans la définition des architectures MILS, le Separation Kernel est aujourd'hui la solution la plus implémentée [GWV03 ; Hei+06 ; Whi+04]. Les systèmes MILS et MLS requièrent, en plus des différentes fonctions de sécurité dont le nombre et la complexité dépendent du niveau de certification demandé, une compatibilité avec les quatre propriétés II.1.2.1, II.1.2.2, II.1.2.3 et II.1.2.4.

La réponse au besoin de MultiLevel Security et de Multi-Independent Level of Security via l'usage d'un Separation Kernel permet de construire une architecture logicielle où se côtoient des compartiments hébergeant des fonctions logicielles plus ou moins complexes. Ces fonctions logicielles peuvent correspondre à des systèmes d'exploitation complets, lorsqu'il est nécessaire de déployer des services complexes comme les services réseaux IP. Le Separation Kernel se comporte alors comme un hyperviseur avec des propriétés de sécurité particulières. C'est sur cette base que je m'appuie dans le cadre de ma thèse comme base de l'architecture logicielle. Afin de pouvoir sélectionner le bon candidat pour un Separation Kernel, j'ai étudié les solutions existantes de virtualisation et les fonctions de sécurité qu'elles supportent.

Afin d'assurer le cloisonnement de ces services complexes, considérés comme n'étant pas de confiance, le Separation Kernel doit également être apte à héberger dans certains compartiments des moniteurs de sécurité certifiables. Ceux-ci sont donc de petite taille, sous forme d'une fonction simple s'exécutant directement au-dessus du Separation Kernel. Cependant, ces fonctions doivent pouvoir être dissociées du Separation Kernel en terme de binaires, ceci afin de limiter la complexité de chaque brique logicielle, et pour pouvoir être potentiellement compatible avec le principe de certification par composition.

II.1.3 Interconnexion sécurisée inter-domaines de sécurité

II.1.3.1 Le contrôle de flux et les passerelles de sécurité

Des travaux ont été effectués [Blu+11][Bet+09] pour répondre au besoin d'interconnexion de serveurs hébergeant des données dont le niveau de sensibilité et les profils utilisateurs associés sont variables. Ces travaux ciblent la problématique de l'usage du SOA (Service Oriented Architectures) pour les données potentiellement confidentielles [Blu+11] ou une fonction de stockage

de type base de données compatible MILS [Bet+09]. Ces travaux décrivent principalement la gestion des rôles et des attributs par les services, et comment faire interagir ces derniers avec des éléments logiciels de vérification des rôles et les gestionnaires de stockage. Ces travaux prennent cependant comme hypothèse que l'environnement d'exécution est de confiance, travaillant sur une formalisation de l'architecture applicative et les protocoles de communication utilisés.

II.1.4 Synthèse

Aujourd'hui, les solutions permettant d'interconnecter plusieurs domaines de sécurité s'appuient sur une séparation matérielle stricte, impliquant plusieurs équipements [Lag06][Sec12]. Plus globalement, les solutions d'interconnexion de domaines de sécurité ne sont pas conçus pour la modularité, le temps réel ou encore l'embarqué contraint. En effet, ces besoins sont plus récents, conséquence des exigences venant de l'avionique et du domaine véhiculaire, pour lesquelles les produits de sécurité ne sont pas encore définis.
Dans le cadre de ma thèse, je cherche à déterminer une solution logicielle afin de répondre aux besoins systroniques, en m'inspirant des solutions existantes qui ont su démontrer leur capacité à répondre aux exigences sécuritaires. Je cherche donc à définir une solution complète compatible avec les exigences des architectures MLS et dans le même temps fournissant suffisament de compléments logiciels pour répondre aux contraintes de la problématiques générale. Je dois donc tout d'abord sélectionner le bon Separation Kernel. Ce choix s'appuie sur un état de l'art des différentes solutions de compartimentation et de virtualisation, décrites dans le Chapitre suivant.

Chapitre II.2

État de l'art des solutions de virtualisation

Sommaire

II.2.1 Étude de la sécurité des solutions de virtualisation existantes **52**

II.2.1.1 Étude de la solution Xen 52

II.2.1.2 Étude de la solution QubesOS 54

II.2.1.3 Étude de la solution LXC 55

II.2.1.4 Étude de la famille L4 56

II.2.1.5 Étude de la solution PikeOS 56

II.2.2 Synthèse **58**

Les solutions de virtualisation sont de bons candidats pour répondre au besoin sécuritaire porté par le MILS (Multi-Independent Level of Security) et le MLS (MultiLevel Security). Ces solutions, pour certaines d'entre elles, fournissent une architecture logicielle compatible avec le principe de Separation Kernel. Le but d'un Separation Kernel est d'assurer un cloisonnement strict entre des environnements logiciels, tout en ayant les propriétés nécessaires pour être intégré dans la Trusted Computing Base (confer Définition I.2.1.11). Dans ce chapitre, je décris les différentes solutions de virtualisation existantes candidates au principe de Separation Kernel. Ne sont considérées que les solutions Open-Source ou Européennes, pour des raisons contractuelles (problématique ITAR [1])

1. International Traffic in Arms Regulation, règlement américain limitant l'usage de produits logiciels ou matériels de haute sécurité (i.e. certifiés EAL4+ ou supérieur)

II.2.1 Étude de la sécurité des solutions de virtualisation existantes

II.2.1.1 Étude de la solution Xen

Principe généraux

Xen [Bar+03] est un hyperviseur s'appuyant sur une architecture de type micro-noyau. Ce dernier fournit les éléments de base pour la gestion et le déploiement des machines virtuelles, sous le contrôle d'une machine virtuelle particulière correspondant à un domaine d'administration, appelé dans le cas de Xen *dom0* [Fra+04].

Ce domaine possède des droits élargis sur l'interface de communication entre l'hyperviseur et les machines virtuelles. Il est en effet apte à créer, configurer, lancer, stopper ou détruire toute autre machine virtuelle, dénommée dans le cadre de Xen *domU*, pour domaine Utilisateur.

Xen fournit une solution de virtualisation de niveau OS. Ainsi, chaque machine virtuelle possède sont propre noyau. Celui-ci peut être géré de plusieurs manières :

1. Via de la paravirtualisation. Dans ce cas, le noyau du domaine utilisateur est modifié pour s'appuyer sur l'API de l'hyperviseur pour tout traitement nécessitant des droits d'administrateur sur la machine.
2. Via de la virtualisation matérielle. Dans ce cas, l'hyperviseur s'appuie sur les mécanismes de virtualisation matérielle tels que Intel-VT ou AMD-V pour gérer la ségrégation des accès aux périphériques et à la mémoire. Cela permet d'exécuter des noyaux virtualisés non modifiés, comme le noyau Windows.

Éligibilité au principe de Separation Kernel

Xen est une solution de virtualisation proposant plusieurs éléments intéressants en terme de cloisonnement. Il peut assurer un cloisonnement strict de la mémoire entre les machines virtuelles. Il est également apte à assurer une charge minimum garantie à chaque machine virtuelle pour un laps de temps donné. C'est le mécanisme présent depuis la version 3 de l'hyperviseur et que les développeurs ont appelé S-EDF [CDV07]. Cet ordonnanceur permet de déterminer, par configuration, une durée d'exécution minimum par fenêtre temporelle ainsi que la capacité à dépasser cette durée. La durée d'exécution est variable en fonction des entrées/sorties des différentes machine virtuelles, détectable par l'hyperviseur car ces dernières transitent par le dom0. C'est une spécificité de l'hyperviseur Xen, seul hyperviseur utilisant un domaine d'administration étant lui-même une machine virtuelle. L'implémentation de l'ordonnanceur S-EDF ne repose malheureusement pas sur des bases théoriques solides. En effet, cette implémentation n'a pas été validée sur la base des principes mathématiques de l'EDF,

mais correspond à une implémentation générique d'un ordonnanceur prenant en compte des échéances pour les machines virtuelles. Les limites comportementales, en terme d'élection de la machine virtuelle ayant la plus petite échéance relative, ont été montrées plusieurs fois [CAA09][CGV07][Gov+07].
De plus, Xen ne permet pas à ce jour de définir des politiques d'ordonnancement strictes de type Time Division Multiplexing. Un tel ordonnancement est nécessaire pour assurer que le comportement d'une machine virtuelle ne vient jamais impacter temporellement les autres machines virtuelles. Cet ordonnancement est donc nécessaire pour garantir un cloisonnement strict en terme de sécurité. Ainsi, la variation de charge CPU liée à une machine virtuelle impliquera une diminution ou une augmentation de charge CPU allouée à une autre machine virtuelle, générant ainsi un canal auxiliaire (confer Définition I.2.1.6).

Les tâches des machines virtuelles sont exécutées et ordonnancées par leur noyau. L'hyperviseur n'a donc pas la visibilité sur la charge ou la manière dont sont ordonnancées les tâches hébergées dans les machines virtuelles. Cependant, il est apte à réagir aux événements matériels asynchrones, en adaptant l'ordonnancement des machines virtuelles pour limiter la latence de traitement. Cependant, un tel ordonnancement est difficilement compatible avec des contraintes de temps réel.

Parmi les solutions open-source connues de virtualisation ou de partitionnement, l'hyperviseur Xen est celui ayant la plus petite empreinte mémoire, du fait de son architecture de type micro-noyau. Cependant, parmi les micro-noyaux, il reste très volumineux avec ses 250.000 lignes de codes, en comparaison avec les micro-noyaux de la famille L4 [Kle+09], décrits dans la Section II.2.1.4 de ce chapitre.

La gestion du réseau dans la solution Xen s'appuie exclusivement sur la pile réseau du domaine d'administration. Ainsi, la définition du réseau virtuel se fait via l'usage de bridges dans ce domaine, générant un surcout non négligeable dans le chemin de données réseau d'une machine virtuelle utilisateur.
Ainsi, lorsque deux machines virtuelles communiquent, l'ordonnancement du domaine d'administration est nécessaire pour permettre au flux d'être reçu de l'émetteur puis renvoyé vers le destinataire [AMN06].

D'un point de vue sécurité, le domaine d'administration est d'ailleurs un élément dangereux car ayant des droits très étendus. La solution QubesOS, qui a été maquettée sur l'hyperviseur Xen, s'appuie sur un mécanisme de chiffrement/déchiffrement des données entre les machines virtuelles utilisateur afin de limiter l'impact d'une corruption du domaine d'administration. Les systèmes de fichiers des machines virtuelles utilisateurs sont également chiffrés afin de limiter la capacité du domaine d'administration à modifier son contenu.

L'usage d'un tel domaine permet de pouvoir simplifier la gestion de la dynamicité des machines virtuelles, ainsi que l'interaction avec le matériel, mais au dépend de la sécurité générale du système [HHD11]. Il est donc d'usage de sortir le domaine d'administration de l'architecture logique, le rendant accessible exclusivement via une console locale d'administration.
Enfin, cette solution n'est à ce jour pas certifiable à un coût raisonnable, du fait de la volumétrie de l'hyperviseur en terme de lignes de code. De plus, il n'existe pas à ce jour de précédents connus d'implémentations natives de gestionnaires de tâches au dessus de l'hyperviseur Xen, autre qu'au travers d'une machine virtuelle hébergeant un système d'exploitation complexe.

Synthèse de la solution Xen

La solution Xen est intéressante pour ses propriétés de partitionnement spacial strict. Cependant, elle net permet pas de manière aisé un partitionnement temporel strict de type TDM. La restriction principale à son usage est son manque de portabilité sur les architectures embarquées. A ce jour, seules les architectures ARM et x86/x86_64 sont supportées, ce qui pose problème dans les domaines de l'avionique et des télécoms, où l'architecture PowerPC est prédominante. De plus, la taille du micro-noyau Xen ne lui permet pas d'initier une procédure de certification sécuritaire et le rend incompatible de l'Exigence II.1.2.2. On peut donc conclure que la solution Xen, bien que attractive au premier abord, ne peut pas être utilisé comme Separation Kernel.

II.2.1.2 Étude de la solution QubesOS

Principe

QubesOS [RW10] est une solution logicielle ciblant les postes de travail. Les exigences initiales de la solution QubesOS sont de pouvoir assurer la confidentialité des données personnelles et professionnelles. Afin de pouvoir répondre à ces exigences, QubesOS s'appuie sur une architecture compartimentée exécutant de manière autonome des environnements d'exécution pour chaque type de donnée [Gad+12].
Les accès aux périphériques sont déportés dans un domaine de confiance afin de limiter l'accès au matériel pour les environnements compartimentés.

Synthèse de la solution QubesOS

La solution QubesOS s'appuie sur l'hyperviseur Xen et un mécanisme d'agrégation de la sortie graphique afin de permettre un accès unifié aux applications des différents domaines de confidentialité. De plus, elle intègre un outillage complexe et des mécanismes de communication entre les compartiments chiffrant les données émises. Cependant, la solution complète QubesOS est lourde et cible exclusivement les postes de travail x86 récents. Elle n'est pas compatible avec les besoins de l'embarqué, mais le principe de compartimentation et les mécanisme

internes de chiffrement sont intéressant pour le besoin de confidentialité des données. L'utilisation de Xen comme base de cloisonnement rend également la solution non évaluable et donc incompatible de l'Exigence II.1.2.2. Bien que ne pouvant être éligible au principe de Separation Kernel, elle utilise des mécanismes de communication inter-domaines intéressant en terme de sécurité.

II.2.1.3 Étude de la solution LXC

Principe

La solution LXC [Lxc] est une solution de virtualisation applicative. Elle s'appuie donc sur un noyau Linux hôte pour ordonnancer les tâches des différents compartiments, avec une gestion de la politique d'ordonnancement potentiellement spécifique à chaque compartiment.

Pour les solutions de virtualisation applicative, la capacité à respecter des exigences temps réel dépend directement des propriétés de l'ordonnanceur de l'hôte. Cependant, une tâche temps réel d'un compartiment sera ordonnancée en concurrence avec les autres tâches temps réel du système, ce qui implique de maîtriser l'ensemble des tâches du système pour pouvoir répondre au besoin temps réel. La solution LXC couplée à une solution de type Linux-RT peut permettre de répondre à des besoins temps réel souple tout en assurant le cloisonnement des différentes tâches. Dans les noyaux Linux récents, il est également possible de définir un certain nombre de propriétés intéressantes en terme de temps réel et de sécurité :

- L'affinité, spécifiant sur quel(s) cœur(s) de(s) processeur(s) le compartiment doit s'exécuter
- La charge associée aux tâches temps réel du compartiment, définissant la charge maximum que peuvent consommer ces dernières
- La charge associée aux tâches non temps réel du compartiment, avec les mêmes propriétés
- La mémoire physique allouée de manière stricte, assurant un cloisonnement spatial entre les compartiments

Ces différentes propriétés rendent la solution intéressante pour des besoins de sécurité limités et des besoins de temps réel souple, mais la solution reste incompatible avec les besoins de temps réel dur et de haut niveau de sécurité, de par la présence d'un noyau Linux comme environnement d'exécution des compartiments.

Synthèse de la solution LXC

La solution LXC est intéressante pour les besoins de maquettage, mais ne permet pas de répondre à des exigences de type MILS, car cette solution est incompatible avec les Propriétés II.1.2.1, II.1.2.2, II.1.2.3 et II.1.2.4 exigées par les architectures MILS et MLS.
Pour des besoins de temps réel souple, elle reste envisageable si les exigences de réactivités sont faibles (supérieur à la milliseconde, comme je le montre dans le Chapitre III.4.2).

II.2.1.4 Étude de la famille L4

L4 représente une famille de micro-noyaux qui partagent un certain nombre de principe d'architecture. Le niveau de sécurité ou la compatibilité avec les exigences temps réel varient selon la solution L4 choisie. Ces noyaux dérivent du micro-noyau L4, créé par Jochen Liedtke [Lie95][Lie96].

Principe

La famille L4 est composée de différents micro-noyaux écrits en C++, comme *Fiasco*, *Pistachio* ou encore *seL4* [Kle+09]. Fiasco est une solution qui n'a été porté que sur une architecture x86, limitant fortement son usage. Cependant, Pistachio a été porté sur un grand nombre d'architectures, dont ARM, PowerPC ou MIPS. Seules les architectures Sparc ne sont pas supportées à ce jour. Le micro-noyau seL4 cible la sécurité, mais est limité à deux architectures matérielles : x86 et ARM. SeL4 a fait vérifier formellement son API et sa gestion mémoire en 2009 [Kle+09].

Synthèse de la famille L4

Parmi les différents micro-noyaux de la famille L4, seL4 est celui qui semble le plus proche du besoin sécuritaire. Son faible volume de code (7500 lignes) et ses premiers travaux en terme de preuve formelle font de lui un bon candidat au besoin MILS. Malheureusement, ce dernier n'est porté que sur deux architectures matérielles et ne propose à ce jour aucune considération pour des exigences de temps réel.

II.2.1.5 Étude de la solution PikeOS

Principe

L'hyperviseur PikeOS [KW07] est un micro-noyau avec des propriétés temps réel (certifié DO-178B-DALB). L'architecture de la solution PikeOS est séparable en trois couches distinctes.

- Le micro-noyau : Ce dernier gère les accès mémoires le mapping des périphériques (lignes d'interruption, espace mémoire d'entrée/sortie) dans les différents compartiments et bien sûr l'ordonnancement.
- Les compartiments : Ces derniers peuvent être des tâches certifiables ordonnançable directement par le micro-noyau ou un système d'exploitation complet, impliquant un ordonnancement de type hiérarchique, comme le décrit la Figure II.2.1.
- Le *System Software* : Il gère les canaux de communications entre les compartiments, mais également les droits d'accès sur l'état de la machine de la part des différents compartiments (capacité à être informé de l'état des autres compartiments, de redémarrer le compartiment ou de redémarrer le système, etc).

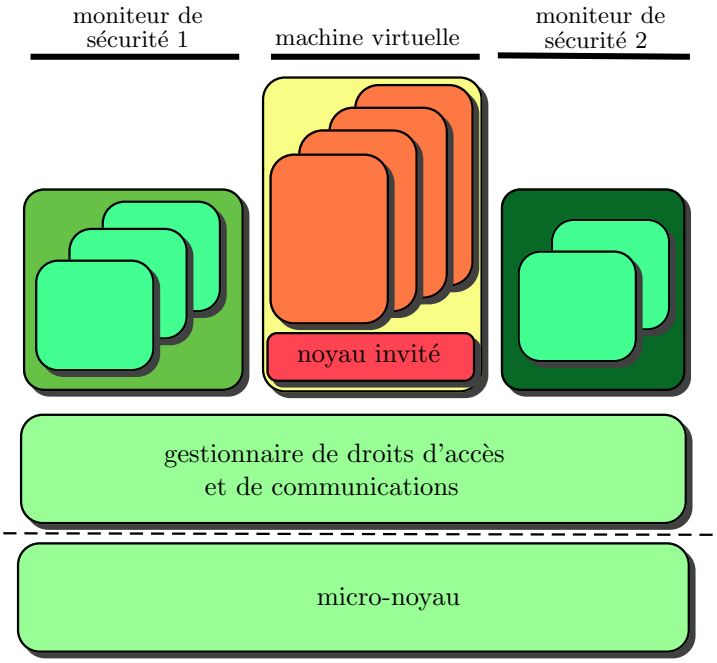

FIGURE II.2.1 – Architecture logicielle de la solution PikeOS

PikeOS s'appuie sur deux principes :

- Le principe de compartimentation spatiale, définissant un espace de mémoire allouée de manière stricte, associé à une politique d'accès aux périphériques définie par configuration
- Le principe de compartimentation temporelle, définissant une durée et une période d'activation. Cette compartimentation permet d'assurer une charge processeur et une réactivité minimum assurée pour la où les tâches exécutée(s) dans le compartiment temporel.

La solution PikeOS permet de gérer les politiques d'ordonnancement de type Rate Monotonic (RM) avec 255 niveaux de priorités. Il permet de plus de définir une séparation temporelle de type TDM pour les différents compartiments temporels, venant s'ajouter à la politique d'ordonnancement RM initiale, afin de déterminer des fenêtres d'ordonnancement strictes et périodiques.

En terme de séparation spatiale, PikeOS permet d'assurer une séparation spatiale stricte en définissant une répartition des compartiments en mémoire par configuration, et assure la ségrégation des accès aux périphériques via la MMU pour les espaces mémoire d'entrée/sortie et au travers d'une gestion fine du contrôleur d'interruption pour associer chaque ligne d'interruption à un compartiment spatial donné. C'est par l'assignation d'une compartimentation spatiale et temporelle à chaque entité devant être ordonnancée que la solution PikeOS permet à la fois de supporter des exigences en terme de temps réel et de sécurité.

En terme de portabilité, la solution PikeOS a déjà été portée sur un grand nombre d'architectures processeur (ARM, PowerPC, MIPS, x86, etc.), et est conçu pour simplifier (et donc limiter le coût financier) d'un portage vers une nouvelle architecture, selon le besoin.

Synthèse de la solution PikeOS

La capacité à ordonnancer à la fois des OS complexes et des tâches certifiables fait de cette solution logicielle un bon candidat à l'intégration des moniteurs de sécurité tels que nécessités par une architecture de type MLS ou MILS [BB09]. Il est en effet compatible avec la Propriété II.1.2.2 d'un système MLS. Les trois autres propriétés (II.1.2.1, II.1.2.3 et II.1.2.4) peuvent être supportées via l'intégration d'éléments complémentaires exécutées au dessus du micro-noyau. De plus, le micro-noyau PikeOS est compatible avec le besoin de certification, y compris à haut niveau, du fait de son très faible volume de code (environ 5000 lignes de codes dans le micro-noyau et 5000 lignes de code dans le gestionnaire de communications s'exécutant au dessus). Un tel volume de code implique également une consommation mémoire extrêmement faible de la part du micro-noyau. Ce dernier est également portable sur un grand nombre d'architectures (ARM, PowerPC, x86, MIPS, SH4, etc.), évitant ainsi toute limitation suite à un choix stratégique de la solution dans un ensemble de produits aux usages multiples.

II.2.2 Synthèse

L'usage d'une solution logicielle de virtualisation à faible empreinte mémoire est un bon candidat au besoin MILS et MLS. Une telle solution permet ainsi d'exécuter divers environnements dont les niveaux de sécurité et de sureté de fonctionnement peuvent être hétérogènes. Cependant, cela implique une compatibilité de la solution de virtualisation logicielle avec à la fois des exigences de temps réel et de sécurité. Le choix d'une telle solution est malheureusement limitée par les différents accord internationaux, comme par exemple ITAR (International Traffic in Arms Regulation), limitant les accès et l'usage des solutions Américaines ayant atteint des niveaux d'assurance élevés, comme par exemples les solutions VXWorks de WindRiver et Integrity de GreenHiils Softwares.
Il est de plus nécessaire, afin de permettre l'intégration de sécurité, d'avoir la possibilité de faire s'exécuter sur l'hyperviseur d'autres éléments logiciels que des systèmes virtualisés, comme je le propose pour PikeOS dans la Section II.2.1.5. Ainsi, l'hyperviseur doit non seulement permettre l'exécution de systèmes d'exploitations virtualisés, mais également des threads directement ordonnancés par ce dernier, tout en assurant une compartimentation spatiale et temporelle forte. Parmi les différentes solutions de virtualisation étudiées, seule la solution PikeOS de Sysgo permet de répondre à l'ensemble de ces besoins tout en prenant en compte les contraintes commerciales liées à ITAR. C'est cette solution que je considère dans ma thèse afin de répondre à la problématique d'architecture de sécurité pour répondre à la fois aux exigences de temps réel

et de sécurité. Dans le cadre de ma thèse, je définis toute une architecture de modules logiciels complémentaires permettant à la fois de répondre aux besoins des systronique et aux propriétés que doivent respecter les systèmes MLS et décrites dans le Chapitre II.1.2.1.

Chapitre II.3

Focus sur l'ordonnancement hiérarchique et la criticité mixte

Sommaire

II.3.1 Solutions pour l'ordonnancement hiérarchique de tâches **61**

II.3.1.1 Principes généraux de l'ordonnancement hiérarchique 61

II.3.1.2 Synthèse 64

II.3.2 État de l'art de l'ordonnancement de tâches à criticité mixte **65**

II.3.2.1 Principe de la criticité mixte 65

II.3.2.2 Synthèse 66

L'usage d'un Separation Kernel est un des moyens proposées pour répondre aux besoins des architectures MILS et MLS. Cette architecture, impliquant potentiellement de la virtualisation de plusieurs environnements autonomes, s'appuie sur un ordonnancement hiérarchique de tâches virtualisées. Dans le cadre de ma thèse, il est également nécessaire de considérer la problématique de criticité mixte, du fait des contraintes fortes des systèmes systroniques en terme de puissance processeur. Ce chapitre décrit un état de l'art sur ces deux thèmes.

II.3.1 Solutions pour l'ordonnancement hiérarchique de tâches

II.3.1.1 Principes généraux de l'ordonnancement hiérarchique

On parle d'ordonnancement hiérarchique lorsqu'une tâche est ordonnancée au travers de deux politiques d'ordonnancement distinctes hiérarchisées. Dans les systèmes temps réels, on considère alors un ensemble de jeux Γ_i de tâches exécutés en concurrence. Pour chaque jeu de tâches, on considère un ensemble de tâches τ_j^i qui lui sont associées. On définit alors

1. Une politique d'ordonnancement des jeux de tâches, nommé ordonnancement global
2. Une politique d'ordonnancement des tâches dans un jeu donné, nommé ordonnancement local

L'ordonnancement hiérarchique a été beaucoup étudié ces dernières années. Ainsi, la capacité à ordonnancer en concurrence des jeux de tâches sur un même cœur d'un processeur permet de pouvoir prendre en considération une classification de ces jeux. La prise en compte d'une classification des jeux de tâche permet de différencier leur ordonnancement en fonction des besoins. On peut ainsi différencier des traitements temps réel de traitement non temps réel. C'est typiquement le cas de Linux qui ordonnance des tâches temps réel et non temps réel en concurrence, en garantissant que les tâches temps réel sont prioritaires sur les autres. Il est également possible de pouvoir ordonnancer en concurrence deux jeux de tâches en utilisant deux politiques d'ordonnancement différentes, pouvant ainsi optimiser les performances selon les propriétés de chaque jeu de tâches.
Aujourd'hui, la plupart des travaux [LB05] ont pour hypothèse une connaissance initiale de l'ensemble des tâches.

La classification des jeux de tâches peut par exemple être faite en fonction des exigences de réactivité associées à ceux-ci. Ainsi, dans [LB05], les auteurs différencient les tâches de traitements des interruptions matérielles, exigeant une forte réactivité, des autres tâches. La Figure II.3.1 décrit une répartition implémentée dans les noyaux Linux jusqu'à la version 2.6.32. Le but de cette classification de jeux de tâches et de pouvoir différencier l'ordonnancement :

- de certains traitements consécutifs à des évènements asynchrones, comme les traitements des IRQ
- des tâches temps réel ordonnancées par l'ordonnanceur FIFO
- des tâches temps réel ordonnancées par l'ordonnanceur RR

La figure II.3.1 décrit le cas de certaines tâches parmi l'ensemble des tâches que l'on peut trouver sous Linux, comme la gestion des interruptions timer, souris et clavier, ou encore les *bottom halves* [Wil03], c'est à dire les traitements déportés dans le temps de certaines fonctions.

Ordonnancement hiérarchique et principe de virtualisation

Lorsque la hiérarchisation de l'ordonnancement est la conséquence de l'emploi de la virtualisation, les deux politiques d'ordonnancement (locale et globale) s'ignorent l'une l'autre. Ainsi, l'état de l'ordonnancement des tâches locales, et leur besoin en terme de charge n'est pas connu de l'ordonnanceur global. Le cas de l'hyperviseur Xen est particulier, car il est capable de déterminer une variation de charge d'une machine virtuelle lorsque cette variation est la conséquence de traitements de type entrées/sorties. Celles-ci transitant par le domaine de contrôle, ce dernier peut alors informer l'hyperviseur afin de modifier l'ordonnancement des machines virtuelles

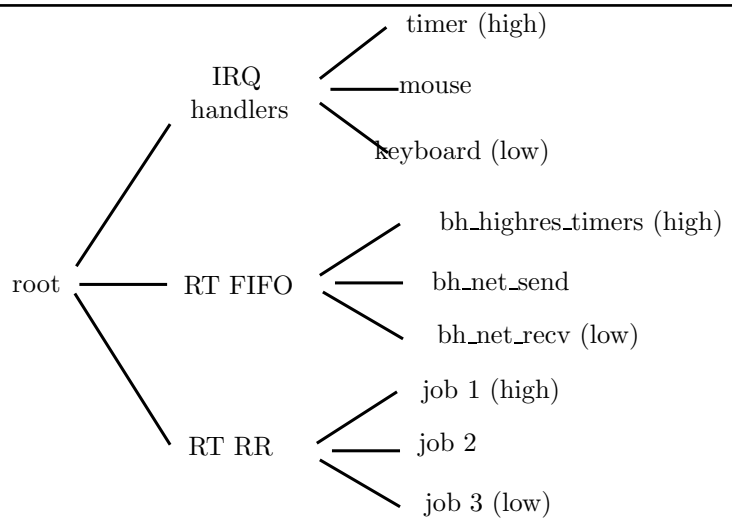

FIGURE II.3.1 – Répartition des tâches du noyau Linux par jeux autonomes, en fonction des exigences de réactivité

en conséquence. Les solutions de virtualisation ne s'appuyant pas sur un domaine d'administration, comme la famille L4 ou PikeOS de Sysgo, ne peuvent pas détecter une variation de charge dans un jeu de tâche d'une machine virtuelle donnée. Dans ce dernier cas, on ne peut plus utiliser un ordonnanceur en-ligne pour impacter l'ordonnancement des machines virtuelles en fonction de la variation de besoin de chaque jeu de tâche.
Dans ce cas, l'ordonnanceur global est souvent de type Fixed Priority (FP). Ce type de restriction est décrit dans [CAA09] où les auteurs expliquent l'impact de l'ordonnanceur global sur la capacité des tâches virtualisées à rester ordonnançable. Différents travaux sur ce sujet ont été publiés, sur la base de solution open-source comme Xen [Bar+03] ou KVM [Kiv+07] ou encore dans un cadre plus générique [CAA09][CAA08].

Du fait de la hiérarchisation de l'ordonnancement des tâches, on parle alors d'un ordonnancement de type X/Y, ou X spécifie l'ordonnancement des machines virtuelles $\Gamma_i, i \in a, b, c...$, Y définissant l'ordonnancement des tâches τ_j^i de la machines virtuelle Γ_i.
Un exemple d'ordonnancement hiérarchique est le RM/EDF, où les tâches sont ordonnancées suivant une politique de type EDF et les machines virtuelles sont ordonnancés suivant une politique de type RM.

Ainsi, dans la solution Xen, une implémentation d'un ordonnanceur global qui se veut temps réel, que les développeurs ont appelé S-EDF, a été faite afin d'assurer une réservation de ressource CPU minimum dans une fenêtre temporelle aux différentes machines virtuelles. Cependant, cette solution a ses limitations, principalement lorsque les tâches temps réel impliquent un grand nombre d'entrées/sorties, impactant le respect de l'ordonnancement des autres machines virtuelles [CAA09].

KVM utilise une architecture différente, puisqu'il implique un hôte pouvant ordonnancer des tâches potentiellement temps réel à côté de machines virtuelles. Dans ce cas, les tâches natives (non virtualisées) peuvent impacter l'ordonnancement des machines virtuelles et des jeux de tâches associés, comme le décrivent T. Cucinotta, G. Anastasi et L. Abeni dans [CAA08]. Les auteurs proposent un nouvel ordonnanceur global, qu'ils dénotent Constant Bandwidth Server [AB98] (CBS) afin d'assurer une réservation stricte de charge processeur pour les différentes machines virtuelles, limitant fortement l'impact des tâches natives. La gestion concurrente de jeux de tâches temps réel virtualisés et de jeux de tâches natifs (ordonnancés directement par l'hôte) est donc un problème complexe à résoudre. Dans le cadre des architectures MLS, ce type d'architecture est nécessaire afin de pouvoir traiter les moniteurs de sécurité comme des tâches du *separation kernel*, ces dernières ne pouvant être intégrées à un OS virtualisé sans perdre leur certifiabilité.

II.3.1.2 Synthèse

L'ordonnancement hiérarchique est un mécanisme nécessaire à l'exécution de compartiments autonomes tels que le nécessitent les architectures MLS. Lorsque l'on s'appuie sur un Separation Kernel sur lequel on doit positionner des fonctions de sécurité ordonnancées directement par l'ordonnanceur global, il est alors nécessaire de vérifier que l'exécution de ces dernières ne vient pas impacter l'ordonnançabilité des tâches temps réel exécutées dans les machines virtuelles. De la même manière, si certaines des fonctions de sécurité, exécutées nativement à coté de machines virtuelles, ont des contraintes de temps réel, il est alors nécessaire d'en assurer également l'ordonnançabilité. Je propose dans le Chapitre III.3.1 une solution pour répondre à cette problématique, où je m'appuie sur les propriétés du micro-noyau PikeOS afin de compartimenter les tâches exécutées directement par l'ordonnanceur de l'hyperviseur dans des slots temporels disjoints. Une telle compartimentation implique cependant de considérer l'impact du slot temporel et de la période de l'ordonnancement TDM dans l'ordonnançabilité de ces tâches, ces dernières s'exécutant alors selon un ordonnancement hierarchique sur un ordonnancement global de type TDM, bien que ces dernières n'appartiennent pas à une machine virtuelle et soient gérées par l'hyperviseur directement.

II.3.2 État de l'art de l'ordonnancement de tâches à criticité mixte

II.3.2.1 Principe de la criticité mixte

Propriétés des tâches d'un ensemble à criticité mixte

En 2007, S. Vestal[Ves07] a décrit le principe d'une tâche dont la criticité peut être mixte. On définit une tâche à criticité mixte en définissant plusieurs pire cas d'exécution (WCET), chaque WCET étant associé à un niveau de criticité. La multiplicité des coûts d'exécution pire cas vient du niveau d'assurance exigé par chaque niveau de criticité. Plus ce dernier est élevé, plus l'assurance demandée par la certification est forte. En conséquence, la mesure d'exécution pire cas est de plus en plus pessimiste afin de garantir que la tâche respecte son WCET. Ainsi, une tâche de faible niveau de criticité peut s'appuyer sur une mesure empirique, dont le WCET est forcément plus faible qu'un calcul formel du WCET.

On définit pour chaque tâche un niveau de criticité en fonction de son importance. Un système s'exécute par défaut dans le niveau de criticité le plus faible. Si une tâche de criticité supérieure au niveau courant dépasse le WCET qui lui est alloué pour ce niveau, il faut augmenter la criticité. On accroît alors la criticité du système et on désactive les tâches dont le niveau de criticité est inférieur au nouveau niveau de criticité. Ceci est fait afin de maintenir l'ordonançabilité des tâches critiques.

Ordonnancement d'un ensemble de tâches à criticité mixte

L'ordonnancement à criticité mixte est un domaine de recherche émergeant gagnant un intérêt croissant. Vestal [Ves07] a initialement introduit le modèle de tâche à criticité mixte. Dans son travail, il décrivit la difficulté à calculer le coût d'exécution pire cas exact, et observa que dans la pratique, plus le niveau d'assurance requis est élevé, plus pessimiste était l'approximation de la valeur du coût d'exécution pire cas. Ce niveau d'assurance est caractérisé par le niveau de criticité. Il suggéra également une stratégie basée sur l'ordonnancement à priorité fixe basée sur le principe d'attribution de priorité de Audsley [Aud91]. Dorin et al. [Dor+10] ont démontré que cet algorithme d'attribution de priorité était optimal pour les tâches à criticité mixte. Les auteurs ont également démontré que l'algorithme de Vestal visant à optimiser les exigences de ressources nécessaires à l'ordonnancement de tâches à criticité mixte, était optimal dans le cas restreint de tâches sporadiques ou périodiques indépendantes à échéances contraintes.

Aujourd'hui, la problématique d'ordonnancement à criticité mixe (*MC-scheduling*) est communément connue pour arriver dans deux contextes différents.

- Le premier correspond à des exigences de certifications multiples. Dans ce contexte, différentes autorités de certification ont besoin de valider les fonctionnalités de l'application. Néanmoins, plus la fonctionnalité est critique, plus pessimiste sera l'approximation du WCET. Baruah et al. [BLS10b] ont étudié les systèmes à criticité mixte dans ce contexte, mais ont restreint leur travail à un ensemble de jobs à criticité mixte. En particulier, Baruah [Bar+10] a montré l'insolubilité de l'ordonnancement à criticté mixte. Ils ont alors suggéré deux conditions d'ordonnançabilité suffisantes, dénotées condition de *WCR-scheduling* et condition d'*OCBP-scheduling*. Plus tard, Baruah et Li [LB10] ont étendu leur travaux initiaux et ont proposé une stratégie d'ordonnancement à priorité fixe basée sur la condition d'*OCBP-scheduling*. Barual et al. ont également adapté l'algorithme EDF (Earliest Deadline First) pour les ensemble de tâches à criticité mixte. Cette approche est connue sous le nom EDF-VD. Plus récemment, Quan et al. [Gua+11] ont présenté une nouvelle approche pour ordonnancer des systèmes à criticité mixte, qui s'appuie sur un calcul hors ligne de la priorité des jobs, qui est ensuite utilisée par un ordonnanceur en ligne. Au même moment, Baruah et al. [BBD11] ont formalisé l'analyse de temps de réponse de tâches à criticité mixte.
- Le second contexte dans lequel la criticité mixte est définie considère que parmi toutes les fonctionnalités déployées sur une unique plateforme de calcul, certaines peuvent être plus critiques que d'autres en terme d'importance pour le bon fonctionnement général de la solution. C'est le cas que je considère dans ma thèse, dans le cadre de la définition d'une passerelle d'interconnexion systronique. Ainsi, certains traitements de flux sont plus critiques, comme le traitement du système d'arme que d'autres, comme les caméras extérieures. Dans ce contexte, de Niz et al. [NLR09] ont observé que l'approche basée sur la réservation de charge, afin d'isoler les fonctionnalités et prévenir les interférences, peuvent avoir pour conséquence un problème d'inversion de criticité, où une tâche moins critique est favorisée par rapport à une tâche de plus haute criticité lorsque cette dernière dépasse le slot temporel lui ayant été alloué. Ils suggérèrent une nouvelle approche, dénommée *zero-slack scheduling*, pour éviter ce problème. Néanmoins, Lakshmanan et al. [LNR11] ont démontré que l'approche du *zero-slack scheduling* pose problème lorsque des tâches de niveaux de criticité différents partagent des ressources. Les auteurs ont alors proposé une modification du protocole PCP (Priority Ceiling Protocol) afin de coordonner le changement de niveau de criticité de l'algorithme d'ordonnancement zero-slack afin d'éviter tout deadlock. Ils définissent ainsi un nouveau protocole de synchronisation dérivé de PCP, dénoté PCCP (Priority and Criticality Ceiling Protocol).

II.3.2.2 Synthèse

Les principes de la criticité mixte permettent de réagir à une infaisabilité temporaire de l'ordonnancement de l'ensemble des tâches. On s'appuie alors sur le niveau de criticité prédéter-

miné pour chaque tâche afin de modifier le jeu de tâches afin d'assurer son ordonnançabilité dans cet état à criticité augmentée. Néanmoins, dans le cadre d'un ordonnancement de tâches périodiques sans dépendances entre les tâches de niveaux de criticité différents sur un seul processeur, les systèmes à criticité mixte ne permettent pas de diminuer le niveau de criticité du jeu de tâches tout en assurant le maintien de l'ordonnançabilité de l'ensemble de tâches.

Dans le cadre de ma thèse, je m'appuie sur les principes de criticité mixte afin de pouvoir répondre à l'impossibilité initiale de pouvoir ordonnancer l'ensemble des traitements sur l'ensemble des flux si ces derniers impliquent un coût d'exécution de chacun des traitements atteignant tous leur WCET. Cette impossibilité est la conséquence des contraintes liées au matériel et à la performance maximum de ce dernier.

Afin de respecter les contraintes liées à la criticité mixte décrites dans ce chapitre, je considère que l'ensemble des tâches périodiques à criticité mixte de la passerelle est ordonnancé sur un seul cœur processeur et qu'il n'y a pas de partage de ressources entre des tâches de niveaux de criticité différents. Dans le Chapitre III.3.3, je propose néanmoins une solution, dans laquelle nous avons cherché tout d'abord à retarder l'augmentation du niveau de criticité du scénario, puis à déterminer l'instant où il est possible de réinitialiser le niveau de criticité d'un jeu de tâches. Il devient alors possible de redescendre dans un état de criticité plus faible, et donc de relancer l'exécution de tâches qui avaient été suspendues.

Troisième partie

Proposition de solution logicielle de passerelle systronique

Chapitre III.1

Rappel des contraintes et hypothèses générales

Sommaire

III.1.1 Matrice de conformité de la solution proposée **71**

III.1.2 Définition des hypothèses générales **74**

III.1.2.1 Hypothèses sur l'architecture matérielle 74

Avant de présenter la solution technique en réponse au besoin initial, je formalise tout d'abord l'ensemble des contraintes décrites dans la problématique générale sous forme d'exigences numérotées. Pour chacune d'entre elle, je précise si la solution que je propose permet d'y répondre. Je définit ensuite les hypothèses générales sur le matériel, nécessaires au bon fonctionnement de la solution.

III.1.1 Matrice de conformité de la solution proposée

Dans les chapitres suivants, je décris la solution que je propose pour répondre aux contraintes de la problématique générale. Ces contraintes se classes en plusieurs familles :

- embarquabilité : la solution doit pouvoir être intégrable dans un environnement contraint (poids, volume et consommation) et doit pouvoir évoluer en terme de matériel supporté.
- maintenabilité : la solution doit pouvoir évoluer et permettre une paramétrisation (e.g. mise à niveau d'une politique de sécurité).
- sécurité : la solution doit pouvoir répondre aux contraintes sécuritaires des systèmes systroniques.
- sûreté : la solution doit pouvoir répondre aux contraintes d'interaction de systèmes dont le niveau de sûreté est hétérogène. C'est par exemple le cas lors de l'interconnexion du système de bord et du système multimédia.

– temps réel : la solution doit pouvoir assurer un traitement des flux borné et garanti, malgré les contraintes d'embarquabilité.

La matrice suivante synthétise les différentes exigences déduites des différentes contraintes de la problématique générale. Pour chaque exigence, je précise si la solution que je propose est conforme ou non. Je dénote chaque exigence de la manière suivante : REQ_<préfixe famille>_<numéro> :

– REQ_EMBAR_xxx correspond à une exigence d'embarcabilité
– REQ_MAINT_xxx correspond à une exigence de maintenabilité
– REQ_SECUR_xxx correspond à une exigence de sécurité
– REQ_SURET_xxx correspond à une exigence de sûreté de fonctionnement
– REQ_TEMPS_xxx correspond à une exigence de temps réel

Pour chaque exigence présente dans cette matrice, je propose une réponse dans le cadre de la solution décrite dans les chapitres suivants. Si la solution est non conforme d'une des exigences présentée ici, cette non-conformité est précisée dans la matrice ci-dessous.

Matrice de traçabilité

Exigence	Besoin	Famille	Conformité
REQ_EMBAR_001	La solution doit pouvoir s'exécuter sur un matériel de faible puissance	Embarquabilité	✓
REQ_EMBAR_002	La solution doit être portable à faible coût	Embarquabilité	✓
REQ_MAINT_001	Les éléments de sécurisation de la solution doivent pouvoir supporter un mécanisme de configuration et de mise à jour de la configuration	Maintenabilité	✓
REQ_MAINT_002	Les éléments de sécurisation de la solution doivent pouvoir supporter un mécanisme d'audit et de remontée d'informations statistiques	Maintenabilité	✓

Page suivante . . .

continuation de la page suivante

Exigence	Besoin	Famille	Conformité
REQ_SECUR_001	La passerelle doit permettre le transfert d'information entre deux domaines de sécurité (conformité avec les quatre exigences des systèmes MLS)	Sécurité	✓
REQ_SECUR_002	La passerelle doit assurer la confidentialité des données qu'elle transmet	Sécurité	✓
REQ_SECUR_003	La passerelle doit assurer l'intégrité des données qu'elle transmet	Sécurité	✓
REQ_SECUR_004	La passerelle doit assurer la disponibilité des données qu'elle transmet	Sécurité	
REQ_SECUR_005	La passerelle doit permettre l'hébergement de fonctions indépendantes de plusieurs niveaux domaines de sécurité disjoints (conformité avec les quatre exigences des systèmes MILS)	Sécurité	✓
REQ_SECUR_006	La passerelle doit permettre le transfert d'informations désensibilisées entre un premier domaine de sécurité et un autre domaine de sécurité de plus faible niveau de sécurité	Sécurité	✓
REQ_SURET_001	La passerelle doit assurer un cloisonnement suffisamment fort pour interconnecter des systèmes dont le niveau de sûreté (et la certification associée) est différent	Sûreté	✓

Page suivante . . .

continuation de la page suivante

Exigence	Besoin	Famille	Conformité
REQ_TEMPS_001	La passerelle possède une borne maximum pour la latence de traversée d'un flux	Temps réel	✓
REQ_TEMPS_002	La passerelle doit être capable de réagir au mieux à un dépassement du WCET d'une ou de plusieurs de ses tâches de traitement des flux réseaux en transit	Temps réel	✓

TABLE III.1.0 – Liste des exigences générales induites par la problématique

III.1.2 Définition des hypothèses générales

III.1.2.1 Hypothèses sur l'architecture matérielle

La définition d'un système sécurisé implique de prendre en compte l'impact sécuritaire du matériel. Dans le cadre de ma thèse, j'ai ciblé principalement les propriétés du logiciel, et considère en conséquence des hypothèses strictes sur le matériel.
Bien que ces hypothèses soient peu réalistes sur du matériel généraliste (i.e. du marché civil), elles doivent être supportées pour assurer les propriétés de sécurité, à défaut d'intégrer un design matériel spécifique. Ces hypothèses sont les suivantes :

HYPOTHÈSE III.1.2.1
conformité *: L'ensemble des éléments matériels sont conformes de leur documentation. Ils assurent le respect des exigences documentaires et ne fournissent aucune propriété non documentée*

HYPOTHÈSE III.1.2.2
compatibilité avec le temps réel *: Les éléments d'interconnexion matériels (bus d'interconnexion mémoire, etc.) sont supposés déterministes. Ces éléments ne sont pas considérés dans le cadre de ma thèse comme impactant.*

HYPOTHÈSE III.1.2.3
sûreté de fonctionnement *: Les éléments matériels sont fiable. Ils ne sont pas source d'erreur et n'impactent pas le logiciel par un quelconque dysfonctionnement.*

HYPOTHÈSE III.1.2.4
sécurité *: Les éléments matériels sont réputés sécurisés. Ils ne génèrent pas de faille de sécurité du fait d'une erreur d'architecture ou de comportement.*

Bien que ces hypothèses soient considérées valides dans le cadre du matériel envisagé dans la solution que je propose, l'usage volontairement non conforme de ce matériel par le logiciel doit être considéré pour les problématiques de sécurité. L'usage du matériel comme base pour des canaux cachés ou auxiliaires est considéré dans le cadre de ma thèse et des solutions sont proposées pour répondre à cette problématique.

Chapitre III.2

Proposition d'une architecture logicielle sécurisée pour le MLS orientée traitement de flux

Sommaire

III.2.1 Proposition d'architecture et choix structuraux **78**
III.2.1.1 Principe de passerelle de traitement de flux MultiLevel Security . 78
III.2.1.2 Présence d'éléments non certifiables 79
III.2.1.3 Moniteurs de sécurité 80
III.2.1.4 Synthèse des choix structuraux 81
III.2.2 Caractérisation du risque sécuritaire **81**
III.2.2.1 Définition des données à protéger 81
III.2.2.2 Intégration des problématiques réseaux 84
III.2.2.3 Définition des profils d'attaque 85
III.2.2.4 Synthèse du risque sécuritaire 87
III.2.3 Définition des moniteurs de sécurité **88**
III.2.3.1 Protection de l'intégrité des données 88
III.2.3.2 Garantie de transmission de flux sans canal auxiliaire inverse . . . 89
III.2.3.3 Conformité des données transmises à la politique de sécurité . . . 92
III.2.3.4 Compartimentation garantie 94
III.2.4 Problématique d'exploitation des canaux auxiliaires **94**
III.2.4.1 Caches processeurs et canaux cachés 94
III.2.4.2 Hypothèses sur le comportement du contrôleur de cache proposé . 95
III.2.4.3 Formalisme de l'interface de contrôle du contrôleur de cache . . . 95
III.2.4.4 Description de l'interface de contrôle du partitionnement du contrôleur de cache 96
III.2.4.5 Principe et automate 97

Dans ce chapitre, je décrit l'architecture générale de la solution logicielle. Les choix que j'effectue ont pour but d'assurer une compatibilité avec les contraintes de sécurité de la problématique, tout en évitant de devenir incompatible des contraintes temps réel, comme je le décrirai dans le chapitre suivant.

III.2.1 Proposition d'architecture et choix structuraux

III.2.1.1 Principe de passerelle de traitement de flux MultiLevel Security

Dans le cadre de la problématique des systèmes systroniques, il est nécessaire de considérer la colocalisation de domaines de sûreté et de sécurité disjoints sur un même matériel. Cette exigence correspond à l'Exigence REQ_SECUR_005 de la matrice de conformité du Chapitre III.1. Pour répondre à cette problématique, je propose de définir une solution logicielle répondant aux besoins de Multiple Independent Levels of Security (MILS). En plus des contraintes de cloisonnements assurant une étanchéité suffisante pour répondre aux exigences de sûreté et de sécurité, il est également nécessaire de considérer des mécanismes logiciels permettant de transférer de l'information entre les différents domaines de sécurité. Cette contrainte correspond à l'Exigence REQ_SECUR_001 du Chapitre III.1. Ce transfert ne doit pas impacter les contraintes de sécurité de chacun des domaines. Pour cela :

- Les données transitant du niveau de sécurité le plus élevé au niveau de sécurité le plus bas doivent être complètement contrôlées, afin de valider que ces dernières peuvent être désensibilisées.
- Le flux transitant du niveau de sécurité le plus bas au niveau de sécurité le plus élevé n'exige pas toujours de filtrage en terme de contenu (en fonction du risque sécuritaire mesuré, lié au contenu du flux). Cependant, l'action de transmission du flux ne doit pas fournir d'information sur le domaine de sécurité le plus haut aux éléments émetteurs ou tout autre élément du niveau de sécurité le plus bas.

Afin d'être compatible avec l'exigence de colocalisation, je m'appuie sur le principe des systèmes MLS et MILS. Les mécanismes définis pour répondre aux besoins de type MLS et MILS impliquent le respect des exigences définies dans le Chapitre II.1.2, et formalisées initialement en 2006 [AF+06]. Je m'appuie, dans le cadre de ma thèse, sur le principe de Separation Kernel, compatible avec les systèmes MILS et MLS, auquel j'ajoute des modules de sécurité complémentaires. L'ensemble est alors appelé Extended Secure Separation Kernel (ESSK) :

DÉFINITION III.2.1.1
J'appelle Extended Secure Separation Kernel une architecture logicielle associant un socle logiciel assurant une compartimentation forte et des compartiments certifiables répondant à des exigences de sécurité. L'ensemble est intégrable comme une TCB (Trusted Computing Base) et ses principes de modularité lui permet de répondre à diverses exigences de sécurité.

Au dessus du Separation Kernel, je propose l'implémentation de modules logiciels en charge de répondre aux mêmes besoins que les éléments de sécurité matériels définis dans le cadre des systèmes d'information. Ces modules sont en charge de répondre aux exigences fonctionnelles et sécuritaires décrite dans la problématique. L'architecture logicielle que j'ai choisie est décrite dans ce chapitre.

III.2.1.2 Présence d'éléments non certifiables

La solution de passerelle systronique que je propose connecte des domaines de sécurité disjoints. Elle doit donc être compatible avec les propriétés II.1.2.1, II.1.2.2, II.1.2.3 et II.1.2.4 des architectures MLS.

Les solutions logicielles pour du traitement télécom, incluant entre autres une implémentation d'une pile TCP/IP, sont trop volumineuses pour être compatibles avec une certification de sécurité à haut niveau. De manière générale, seul l'usage d'éléments logiciels à faible volumétrie de code, de l'ordre de quelques milliers de lignes, permet de supporter une telle certification à coût raisonnable.
Parce que l'implémentation logicielle d'une passerelle de type télécom ne peut être certifiée en terme de sécurité, les éléments d'implémentation de celle-ci sont donc considérés comme ne pouvant pas être de confiance. Il devient alors nécessaire de définir :

- deux familles logicielles Sec et $Unsec$, l'une étant de confiance, l'autre non
- des mécanismes permettant de pouvoir intégrer des éléments non certifiables tout en assurant la sécurité du système dans son ensemble

Dans un premier temps, je définis pour les familles Sec et $Unsec$ les propriétés suivantes afin d'assurer un cloisonnement strict entre ce qui est certifiable et ce qui ne l'est pas :

PROPRIÉTÉ III.2.1.1
Sec définit l'ensemble des éléments logiciels certifiables. Ces derniers sont aptes à contrôler des éléments matériels ou auditer d'autres éléments logiciels de la famille Sec et $Unsec$. Ces éléments font partie de la TCB (Trusted Computing Base). Ces éléments logiciels doivent respecter un certain nombre de contraintes :

- *faible volume de code (quelques milliers)*
- *faible complexité cyclomatique [Han78] (de l'ordre de 8 ou 9 au maximum)*
- *taux de couverture en tests-unitaires élevée*

PROPRIÉTÉ III.2.1.2
$Unsec$ définit l'ensemble des éléments logiciels non-certifiables. Ces derniers ne doivent pas accéder directement à un quelconque élément matériel, en dehors de la mémoire principale, dont l'accès est vérifié par la MMU.

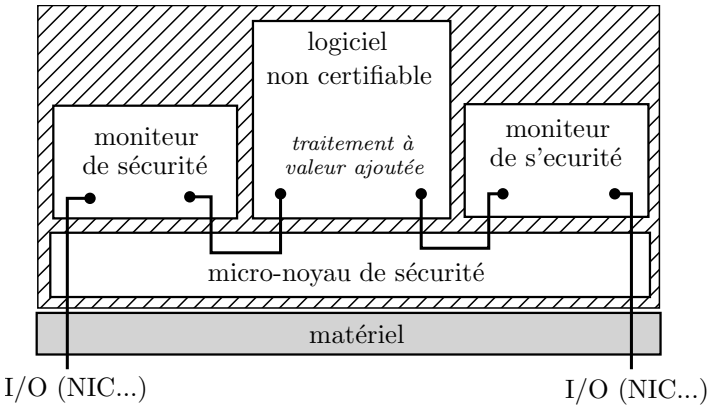

FIGURE III.2.1 – Filtrage des entrées/sorties par un élément logiciel de confiance

PROPRIÉTÉ III.2.1.3
Aucun élément logiciel ne peut être à la fois sûr et non sûr :

$$Sec \cap Unsec = \emptyset$$

Les éléments logiciels non certifiables doivent donc être entourés d'éléments logiciels de confiance (appartenant au groupe Sec) afin de valider l'ensemble de leurs entrées/sorties. Ainsi, la propriété II.1.2.3 des architectures MLS est respectée, comme le montre la Figure III.2.1.

III.2.1.3 Moniteurs de sécurité

Il est montré dans la Section précédente que la présence d'éléments logiciels non certifiables est un mal nécessaire pour permettre un certain nombre de traitements à valeur ajoutée sur des flux télécoms. En effet, une nouvelle implémentation d'une pile réseau (e.g. pile TCP/IP, pile de communication sur réseaux CAN) certifiable étant trop couteuse, j'ai fait le choix d'intégrer de tels logiciels, en prenant en compte, pour certains d'entre eux, le problème de non-certification associée dans l'architecture.

Dans le Chapitre III.2.1.2, je montre que l'usage de tels logiciels peut être envisagé si des éléments certifiables extérieurs sont présents, afin de rester compatible avec les quatre exigences initiales des architecture MLS. Dans le cadre de ma solution, je définis donc des éléments logiciels autonomes et certifiables portant une fonction de sécurité. Par la suite, j'appelle ces éléments logiciels des **moniteurs de sécurité** :

DÉFINITION III.2.1.2
J'appelle moniteur de sécurité un élément logiciel autonome et certifiable portant une fonction de sécurité. Il a pour charge de répondre à un besoin sécuritaire particulier.

Les moniteurs de sécurité permettent de répondre aux différentes exigences du MLS car :

- Ils valident les entrées/sorties des éléments non certifiables.
- Ils détectent et remontent les évènements invalides auprès d'un tiers de confiance.
- Ils assurent le cloisonnement entre les différents éléments logiciels et le matériel, pour assurer l'inviolabilité de l'ensemble de la TCB (confer Définition I.2.1.11).

III.2.1.4 Synthèse des choix structuraux

Dans ce chapitre, je décris les choix structuraux que j'ai fait dans le cadre de la définition de la solution. J'ai caractérisé deux familles d'entités logicielles répondant à des exigences disjointes, et j'ai proposé le principe de moniteur de sécurité, autonome, permettant de respecter les exigences des architectures MLS.
A partir de ces éléments, je vais définir l'architecture logicielle complète permettant de répondre aux besoins d'une passerelle multi-domaines de sécurité, telle que décrite dans la problématique. Une validation de la compatibilité de ces choix structuraux a été faite via un maquettage chez Thales donnant lieu à un premier brevet [TLC13].

III.2.2 Caractérisation du risque sécuritaire

III.2.2.1 Définition des données à protéger

J'ai identifié dans les Sections III.2.1.1 et III.1.2 les principes généraux qui doivent être considérés dans le cadre de la définition d'une architecture logicielle pour le traitement de données entre domaines de sécurité disjoints (i.e. MultiLevel Security). En complément de ces principes généraux, permettant d'être compatible avec le besoin MLS, il faut décrire plus en détail ceux nécessaires pour répondre aux besoins de la problématique générale. Pour cela, je m'appuie sur les principes de base d'une méthode qui a été proposée initialement par l'ANSSI [1], nommée EBIOS [HAT07]. La méthode EBIOS permet de pouvoir définir un niveau de sécurisation nécessaire et suffisant pour répondre au besoin sécuritaire du système cible. Sans entrer dans la rédaction d'un document EBIOS, trop volumineux dans le cadre de ma thèse, je m'appuie néanmoins sur les principes présents derrière le formalisme de ce type de document.

La définition de l'architecture logicielle pour le besoin d'une passerelle multi-domaines de sécurité implique de répondre aux questions suivantes :

QUESTION III.2.2.1
Quelles sont les données à protéger ?

1. Agence Nationale pour la Securité des Systèmes d'Information

QUESTION III.2.2.2
Quel(s) type(s) de protection la passerelle souhaite fournir sur ces données (confidentialité, intégrité, disponibilité) ?

QUESTION III.2.2.3
Quels sont les attaques impactant ces protections dans le cadre de la solution ?

QUESTION III.2.2.4
Pour chaque attaque, quel est le niveau de risque associé (probabilité de l'attaque, complexité technique, etc.)

QUESTION III.2.2.5
Quels sont les fonctions de sécurité nécessaires pour se prémunir contre chacune de ces attaques ?

Les sections suivantes mettent en pratique les principes de la méthode EBIOS pour mettre en adéquation une solution technique avec les besoins sécuritaires liés à la fonction passerelle multi-domaines. Ainsi, l'usage d'une fonction de sécurité doit être mis en adéquation avec le risque sécuritaire auquel elle répond. Lorsque la présence d'une fonction de sécurité dépasse le coût (financier ou managérial) associé au risque sécuritaire, sa présence n'est plus requise.
Néanmoins, dans le cadre de ma thèse, je cherche à pouvoir répondre à divers niveaux de sécurité en fonction du positionnement de la passerelle et du type de véhicule. Je dois donc répondre à des besoins sécuritaires plus ou moins élevés. Je considère donc l'ensemble des attaques identifiées en réponse à la Question III.2.2.4 comme probables et je cherche à répondre à chacune d'entre elles.

Définition des données à protéger

Dans le cadre de ma thèse, je cherche à répondre au besoin d'une passerelle multi-domaines de sécurité et multi-domaines de criticité. La fonction de cette passerelle est de permettre le transfert de flux réseau entre deux domaines dont le niveau de sécurité et le niveau de criticité sont différents. Aux propriétés du MLS décrites dans le Chapitre II.1.2.1, que le logiciel doit supporter, il faut également intégrer des exigences complémentaires liées au flux de données transmis entre deux domaines de sécurité. En effet, la capacité à transférer des données entre deux domaines de sécurité ne doit pas impacter le cloisonnement général de ceux-ci. Ces exigences sont les suivantes :

ÉXIGENCE III.2.2.1
L'action de transmission des flux ne doit pas révéler d'information sur le domaine de plus haute sécurité

La transmission de flux vers le domaine de plus haute sécurité ne doit pas être dépendant de l'état d'exécution de ce dernier. Dans le cas contraire, le domaine de sécurité le plus bas peut

déterminer l'état d'exécution du domaine de sécurité le plus haut, et peut récupérer de l'information comme une clef de chiffrement, si le domaine de plus haute sécurité fait varier son état d'exécution à son escient. Cette exigence est nécessaire pour assurer la compatibilité de la solution avec les contraintes des systèmes MLS.

ÉXIGENCE III.2.2.2
Si les flux sont transmis du domaine de plus haute sécurité au domaine de plus basse sécurité, ces derniers doivent contenir exclusivement des données autorisées à être désensibilisées.

La désensibilisation d'information, qui correspond à autoriser le passage de données d'un domaine de sécurité à un autre domaine de niveau inférieur, implique la gestion d'une politique de filtrage permettant d'autoriser ou non une telle désensibilisation.

ÉXIGENCE III.2.2.3
Les flux transmis doivent être garantis en terme d'intégrité

En effet, si les données transmises sont modifiées, elles ne permettent plus d'assurer une action valide de la part du domaine de plus haut niveau de sécurité. Cette exigence correspond à l'Exigence REQ_SECUR_003 de la matrice d'exigence de la solution, décrite dans le Chapitre III.1.

ÉXIGENCE III.2.2.4
Les flux doivent transiter exclusivement dans la direction initialement choisie, au travers des canaux logiques et physiques initialement choisis

La présence de canaux logiques ou physiques non prévus entraine une faille dans la mécanique de validation des flux transmis. Il devient possible de transmettre de l'information sans contrôle par un moniteur de sécurité certifiable, et rend la solution incompatible avec la propriété II.1.2.1 des systèmes MLS, et donc avec l'Exigence REQ_SECUR_001.

À partir de ces exigences, je détermine la répartition des responsabilités sécuritaires en terme de confidentialité, intégrité et disponibilité.

- La confidentialité des données, correspondant à l'Exigence REQ_SECUR_002, est partagée entre :
  - La configuration de la passerelle, qui est une donnée d'entrée de celle-ci donc en dehors du cadre de ma thèse. C'est cette configuration qui détermine entre autre quel type de flux est autorisé à transiter, et vers quelle destination. La configuration de la passerelle peut potentiellement intégrer des éléments nécessaires au bon fonctionnement de l'interconnexion réseau
  - Les propriétés de cloisonnement de la passerelle, qui répondent également à l'exigence III.2.2.4
- L'intégrité des données est assurée par la passerelle durant le temps de leur transit (en réponse à l'Exigence REQ_SECUR_003).

– Comme précisé dans le cadre de la matrice de conformité aux contraintes de la problématiques générale, la disponibilité n'est pas considéré dans le cadre de ma thèse. A ce jour, la solution que je propose est donc incompatible avec l'Exigence REQ_SECUR_004 du Chapitre III.1

III.2.2.2 Intégration des problématiques réseaux

Problématique générale

La solution consiste à permettre l'interconnexion sécurisée de deux domaines indépendants, avec des exigences temps réel et sécuritaires. Dans le cadre de la définition d'une telle architecture, la gestion des plans d'adressage IP est problématique pour le respect de l'Éxigence III.2.2.1. Il est en effet nécessaire de permettre l'interconnexion de deux services sur deux réseaux disjoints tout en assurant que ces derniers ne soient pas capables de déterminer tout ou partie des informations réseaux associées à l'autre service. Une telle problématique impacte fortement le comportement de la passerelle.

Intégration des proxys

Il devient alors nécessaire de prévoir un mécanisme de séparation protocolaire. Ce mécanisme doit répondre à plusieurs besoins :

– En entrée, les en-têtes de niveau réseau et transport (couches OSI) doivent être retirées, afin d'empêcher toute information dérivant de ces en-têtes, comme le plan d'adressage IP
– En sortie, de nouvelles en-têtes appartenant au plan d'adressage de sortie doivent être construite pour se substituer aux précédantes.

On parle alors d'un mécanisme de séparation protocolaire [BEB02]. Ces mécanismes impliquent l'usage d'applications de type proxy, se substituant d'une part à la destination et d'autre part à la source afin de permettre une interconnexion entre deux services de part et d'autre de la passerelle.

Une telle architecture est décrite dans la Figure III.2.2. Ainsi l'ensemble des communications internes à la passerelle se font indépendamment des protocoles couches 1 à 4, s'appuyant directement sur des buffers. L'usage de proxys n'est pas forcément un mécanisme aisé. Les applications proxys doivent être aptes à gérer des contextes de communications de manière synchrone. Une telle synchronisation peut ainsi impliquer un mécanisme de communication plus ou moins complexe entre les deux proxy corolaires.

Dans le cadre de ma thèse, les proxys sont considérés pour des protocoles simples. Ainsi, on considère comme service un proxy spécialisé dans un protocole de niveau application. Ce proxy est à l'écoute sur un port donné, et interagit avec son corolaire dans l'autre domaine de sécurité, utilisant comme source le même numéro de port. Il est considéré les restrictions suivantes :

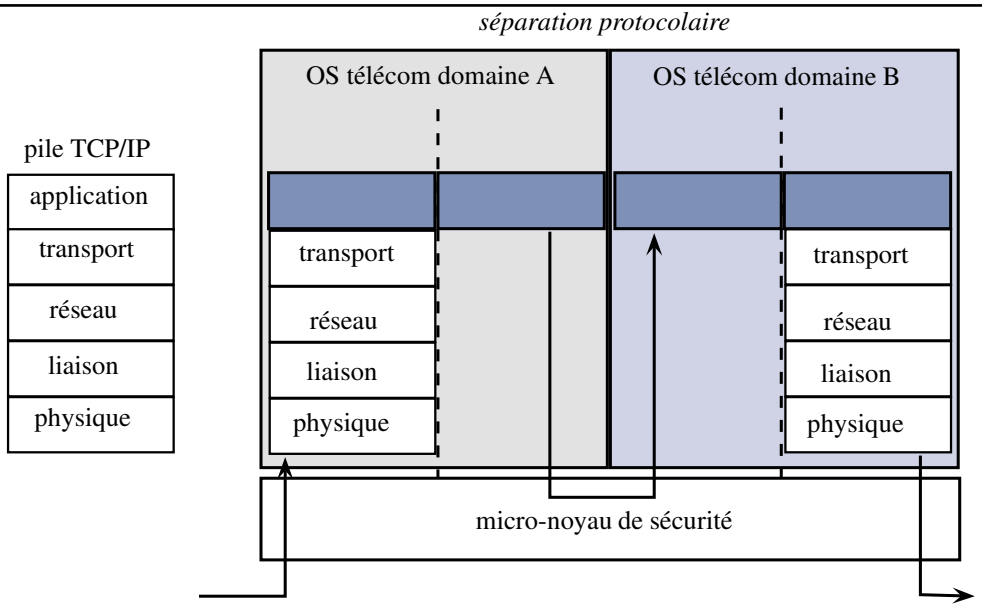

FIGURE III.2.2 – Exemple de séparation protocolaire couche transport

RESTRICTION III.2.2.1
Pour un service donné, une et une seule destination (couple IP et port destination) est considéré par configuration.

RESTRICTION III.2.2.2
Pour un service donné, un seul contexte est considéré à la fois, ne nécessitant ainsi pas de mécanisme de synchronisation.

Un véritable mécanisme de passerelle doit être apte à traiter un ensemble de flux reçus de manière concurrente par un même service. Cela implique alors une capacité de synchronisation des contextes de flux entre le service entrant et le service sortant afin de permettre la différentiation des flux en sortie, tout en respectant l'exigence III.2.2.1.

III.2.2.3 Définition des profils d'attaque

Généralités sur les profils d'attaques

De la définition des Exigences III.2.2.3, III.2.2.1, III.2.2.2 et III.2.2.4, conséquentes à la conformité à l'Exigence REQ_SECUR_001 du Chapitre III.1, il faut dériver des exigences techniques. Pour cela, je détermine d'abord la surface d'attaque. On appelle surface d'attaque les éléments d'interfaces à partir desquels on considère une attaque possible. La surface d'attaque est décomposée en trois environnements disjoints :

1. Les attaques externes initiées du coté du domaine de plus haute sécurité. Je considère alors qu'un élément de ce domaine cherche à faire sortir de l'information non autorisée

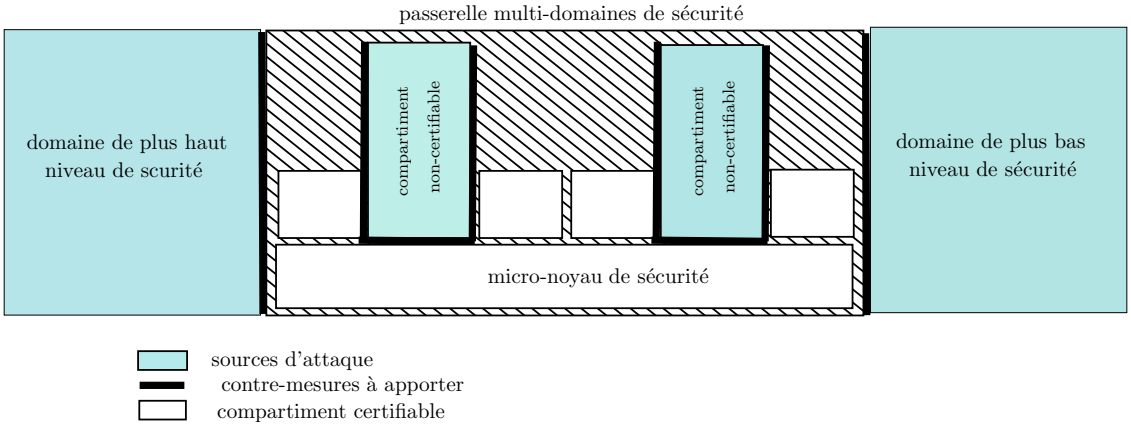

FIGURE III.2.3 – Répartition des surfaces d'attaque sur la passerelle de sécurité

vers le domaine de plus faible sécurité, avec ou sans l'aide d'un élément de ce dernier. Il y a alors un risque de confidentialité du domaine Haut vers le domaine Bas.

2. Les attaques externes initiées du coté du domaine de plus faible sécurité. Je considère qu'un élément du domaine de plus faible sécurité cherche à récupérer de l'information sur le domaine de plus haute sécurité, sans l'aide d'un élément de ce dernier. Il y a alors un risque de confidentialité du domaine Bas vers le domaine Haut.
3. Les attaques internes, venant des éléments de la passerelle appartenant à la famille $Unsec$, telle que décrite dans la Section III.2.1.2, impactantes pour l'exigence III.2.2.4. Il y a lors un risque en terme de confidentialité, d'intégrité et de disponibilité.

Les éléments de la TCB (appartement à la famille Sec) sont considérés comme de confiance. En conséquence, ils ne sont pas considérés comme source d'attaque potentielle.
La Figure III.2.3 schématise la répartition de la surface d'attaque sur le système englobant la passerelle multi-domaines de sécurité.

À ces trois surfaces d'attaque, on associe un niveau de faisabilité de l'attaque, qui permet de déterminer le niveau de risque, et donc l'exigence de protection associée. Dans le cadre de ma thèse, je cherche à définir une solution modulaire associant des éléments non certifiables à des éléments certifiables. Les domaines extérieurs à la solution ne sont de plus pas connus initialement. Il n'est alors pas possible d'en déterminer le risque sécuritaire associé. Pour répondre à un maximum de besoin, je considère alors le risque élevé sur ces trois surfaces et je répond donc en terme de contre-mesures de sécurité à l'ensemble de ces environnements.

Définition des attaques et impact sur les exigences sécuritaires de la solution

Cette Section décrit un certain nombre d'attaques connues impactant les exigences de sécurité de la passerelle.
Le Tableau III.2.1 liste les différents types d'attaques et leur impact sur les exigences de sécurité.

Source	Attaque	Exigence			
		III.2.2.1	III.2.2.2	III.2.2.3	III.2.2.4
Domaine inférieur	Étude temporelle de la vitesse de transit du flux vers le domaine supérieur	X			
Domaine supérieur	Émission de flux non conforme de la configuration de sécurité		X		
Domaine supérieur	Transfert de donnée par canal caché via variation du profil de flux		X		
Interne	Retransmission du flux au travers d'un canal non autorisé				X
Interne	Modification du contenu du flux avant retransmission			X	
Interne	Écoute par canal auxiliaire d'un flux transitant par un autre compartiment pour retransmission	X			X
Interne	Communication entre deux compartiments au travers d'un canal caché	X			X

TABLE III.2.1 – Profils d'attaques et impact sur les exigences sécuritaires

Ces différents comportements permettent de mettre à mal le principe de cloisonnement associé à la passerelle inter-domaines, et doivent donc être bloqués ou à défaut tracés.

III.2.2.4 Synthèse du risque sécuritaire

On peut constater que la surface d'attaque de la solution est grande et nécessite un ensemble d'élements logiciels en charge de se prémunir contre les différents types d'attaque recensés dans cette section.
Dans le cadre de ma thèse, je propose un certain nombre de solutions (contre-mesures) permettant de réagir à ces attaques, ainsi qu'un certain nombre d'éléments de protection, pour les rendre inefficaces ou à défaut réduire leur portée et accroître leur complexité. Ces différents éléments de sécurité sont construits sous forme de *moniteurs de sécurité*, portant chacun une fonction de sécurité particulière. C'est au travers de l'association de tous ces moniteurs qu'il est possible de construire une solution répondant à l'ensemble des exigences de sécurité de la problématique générale de la thèse (REQ_SECUR_001 à REQ_SECUR_006).

III.2.3 Définition des moniteurs de sécurité

III.2.3.1 Protection de l'intégrité des données

Problématique générale

Les moniteurs de sécurité décrits ici ont pour but d'assurer le respect de l'Exigence REQ_SECUR_003. L'intégrité des données transmises est une exigence dont le but est d'assurer que le flux de données qui transite au travers de la passerelle multi-domaines de sécurité n'est jamais modifié sans autorisation préalable. Cela se fait usuellement en définissant une liste de droits d'accès associé au flux, définissant quel élément logiciel (sujet) est autorisé à accéder au contenu du flux (objet). Seuls les sujets explicitement autorisés sont aptes à accéder à ces données et potentiellement les modifier. Dans le cadre de l'architecture choisie, il est nécessaire de faire transiter le flux télécom par des compartiments ne pouvant être considérés comme de confiance. Il s'agit de compartiment appartenant à la famille $Unsec$, qu'il est difficile de réimplémenter sans accroître fortement le coût financier de la passerelle. Il devient alors nécessaire de prévoir des éléments de contre-mesure, permettant de détecter toute modification non autorisée et de réagir à celle-ci.

A propos des moniteurs d'intégrité

Dans le cadre de ma thèse, je propose l'implémentation d'un moniteur de sécurité particulier, afin d'assurer l'intégrité des données transitant par la passerelle. En effet, comme le montre la Figure III.2.3, certains composants de la passerelle ne peuvent être certifiés et sont donc potentiellement attaquables ou corruptibles. Il devient donc nécessaire de vérifier par un moyen certifiable que les données transitant par la passerelle ne sont pas modifiées.

J'appelle moniteur d'intégrité un couple de deux compartiments logiciels, l'un en entrée et l'autre en sortie, interagissant pour valider la signature des données transitant dans la passerelle. Du fait du mécanisme de séparation protocolaire, seule les données de la couches applicative sont vérifiées. Ces deux compartiments s'intègrent en amont et en aval des compartiments télécoms et plus généralement des compartiments de l'ensemble $Unsec$.
Le principe général est de calculer en amont et en aval la somme de contrôle (checksum) de l'ensemble des données de la couche applicative transitant par la passerelle. Les deux moniteurs se partagent alors cette information au travers d'un canal de communication sécurisé, afin que le second moniteur vérifie l'existence de la somme de contrôle qu'il a calculé dans la liste fournie par le premier moniteur. Une telle architecture logicielle implique plusieurs exigences :

ÉXIGENCE III.2.3.1
L'intégration de la signature dans la mémoire partagée doit être garantie avant l'arrivé du paquet dans le second moniteur.

ÉXIGENCE III.2.3.2

Les données de la couche applicatives ne doivent pas être sectionnées. C'est une exigence dure car cela implique que la MTU en aval doit être suffisante pour tout type de paquet reçu en amont.

ÉXIGENCE III.2.3.3

Des paquets peuvent être perdus ou droppés par un autre moniteur de sécurité (e.g. un filtre DPI). En conséquence, les sommes de contrôles doivent avoir une durée de présence dans la mémoire partagée maximum, pour éviter tout débordement.

Dans ma thèse, je considère la problématique temps réel et cherche à déterminer la durée maximum de traversée d'un paquet. Ceci permet de déterminer la durée de présence pire cas de la somme de contrôle (e.g. md5, sha1, sha2) dans la mémoire partagée. En conséquence, si les flux validés en terme d'intégrité sont dimensionnés (taille de trame, débit, burst), on peut alors dimensionner la mémoire partagée nécessaire au bon fonctionnement du moniteur d'intégrité. On peut alors respecter l'exigence suivante :

ÉXIGENCE III.2.3.4

La mémoire partagée est suffisamment grande pour permettre au moniteur amont d'y ajouter les nouvelles sommes de contrôle au fur et à mesure de la réception des paquets.

Bien que ce moniteur de sécurité n'aie pas été validé au travers d'une certification dans le cadre de ma thèse, les exigences nécessaire à son implémentation sont supportée au travers de l'usage d'un moniteur de sécurité décrit ci-après, dans la Section III.2.3.2. Bien que ce moniteur de sécurité ne cible pas la maîtrise du flux entrant pour des raisons de temps réel, il répond néanmoins aux exigences ci-dessus.

III.2.3.2 Garantie de transmission de flux sans canal auxiliaire inverse

Le moniteur de sécurité décrit ici répond à l'Exigence REQ_SECUR_001. Sa présence doit assurer que tout flux transmis entre deux domaines de sécurité puisse se faire sans que les propriétés de confidentialité du domaine de plus haute sécurité ne soit impacté. Le but est d'assurer que le domaine de plus faible niveau ne puisse pas récupérer d'information sur le domaine de plus haute sécurité.

Il existe plusieurs cas qui doivent être considérés dans le cadre du besoin associé à la Problématique générale :

1. Le transfert sens unique de données d'un domaine de plus faible niveau à un domaine de plus haut niveau (Exigence REQ_SECUR_001)
2. Le transfert de flux désensibilisé d'un domaine de plus haut niveau de sécurité vers un domaine de plus faible niveau de sécurité (Exigence REQ_SECUR_006)

A propos d'une architecture de diode logicielle pour le transfert de données

Il existe dans l'état de l'art des solutions permettant d'assurer le transfert de flux unidirectionnel entre deux domaines. C'est le cas des différentes diodes matérielles [Lag06]. Ces solutions impliquent cependant de déployer plusieurs équipements distincts dont la consommation et la volumétrie ne sont pas toujours compatibles avec les exigences de l'embarqué.

Pour répondre à ce besoin, j'ai contribué à la rédaction de deux brevets [Thi+13b][Thi+13a] et à l'implémentation d'un moniteur de sécurité permettant de répondre aux Exigences REQ_SECUR_006 et REQ_SECUR_006, sans nécessiter l'usage de plusieurs matériels distinct, comme cela se fait dans l'état de l'art [Lag06 ; Sec12].
Pour répondre à ces exigences, il est nécessaire d'implémenter une fonction logicielle dont le comportement est prévu pour assurer une opacité forte lors du transit des données entre son entrée et sa sortie, quel que soit le comportement des éléments source et destination du flux.
Ce moniteur de sécurité est accompagné d'une fonction d'audit, permettant de remonter des informations statistiques sur son comportement, le rendant compatible avec l'exigence REQ_MAINT_002. Ce dernier ne nécessite par contre pas de mécanisme de configuration, son comportement ne nécessitant pas de traitement spécifique au flux qu'il fait transiter. Il est donc compatible avec l'exigence REQ_MAINT_001.

Ce moniteur de sécurité a été évalué avec succès dans le cadre d'une Cible de Sécurité de Premier Niveau (CSPN), suivi d'une pré-évaluation du micro-noyau PikeOS sur lequel il s'exécute, au sens des Critères Communs, de niveau EAL5+. De plus, ce moniteur de sécurité est capable de garantir une latence de traversée maximum ainsi qu'un débit garanti par construction, le rendant compatible avec l'exigence REQ_TEMPS_001.
Ces travaux ont donné lieu à deux brevets [Thi+13a][Thi+13b], mais la majorité des éléments techniques est aujourd'hui maintenue secrète par Thales. En conséquence, bien que ce moniteur de sécurité soit aujourd'hui une réalité, je ne peux pas donner dans le cadre de ce document plus d'informations sur son implémentation et son fonctionnement.

Réponse à la problématique des canaux cachés

La transmission de données entre deux domaines de sécurité nécessite plusieurs vérifications. Il y a tout d'abord le filtrage des données selon une politique de sécurité donnée. Ce point est décrit dans la Section III.2.3.3. Néanmoins, la vérification des données à transmettre n'est pas suffisante pour garantir un cloisonnement efficace des deux domaines. En effet, indépendamment de la donnée à transmettre, il est possible de construire un mécanisme permettant de faire fuir de l'information au travers d'un canal caché : la période inter-trame. Le principe est le suivant :

- L'émetteur et le récepteur connaissent le protocole de transmission d'information cachées. Dans le cadre de ma thèse, je considère qu'il s'agit du suivant :
  - Une durée inter-trame de 500 millisecondes correspond à la valeur *0*.
  - Une durée inter-trame de 1000 millisecondes correspond à la valeur *1*.
- L'émetteur émet des paquets dont le contenu est compatible avec la politique de sécurité avec une inter-arrivée variable, afin d'émettre de l'information binaire.
- Le récepteur reçoit les paquets et calcule la durée inter-trame. Il détermine si la période est proche des 500 millisecondes (la donnée est alors 0) ou proche de la seconde (la donnée est alors 1).

La durée des inter-arrivées reste élevée dans les deux cas pour que la présence des filtres entre l'émetteur et le récepteur ne vienne pas impacter la mesure. En s'appuyant sur ce mécanisme, il est possible de transférer une clef symétrique de 128 bits dont la confidentialité est restreinte au domaine de plus haut niveau de sécurité en moins de 120 secondes.

Afin d'empêcher l'usage de ce canal caché, je propose dans le Chapitre III.4 l'implémentation d'un moniteur de sécurité appelé émetteur périodique. Ce moniteur de sécurité s'appuie sur des buffers de type FIFO (First In First Out) de profondeur fixe. Chaque case de ces FIFO est dénoté par la suite *cellule*. L'automate à état de ce moniteur est le suivant :

Ce moniteur de sécurité est en charge de bufferiser les données à émettre pour les transmettre ensuite de manière périodique, à l'unité ou par burst. Si aucun paquet n'est reçu entre deux vidanges du buffer, le dernier paquet est réémis en lieu et place. Le but d'un tel comportement est d'empécher l'utilisation d'un canal caché basé sur l'émission ou la non-émission de paquet, comme décrit plus haut.
Etant donné que la profondeur du buffer de réception du moniteur est fixe, et que l'inter-émission est un paramètre d'entrée, l'émetteur périodique ne peut dépasser un débit de paquet maximum. En conséquence, il peut être amené à dropper des paquets si le buffer de réception est plein, comme le montre la Figure III.2.4. Ce moniteur de sécurité doit donc gérer des compteurs de drop, auditable par un élément d'audit externe. La mécanique d'audit utilisable peut être similaire à celle décrite dans un des brevets que j'ai co-écrit [Thi+13a], car les données audités sont identiques.

L'usage d'un tel moniteur implique des hypothèses sur le flux réordonnancé. Ce dernier doit en effet être dimensionné afin de rester compatible avec la capacité d'émission de l'émetteur périodique. A défaut, l'émetteur périodique peut être dans l'obligation de supprimer certains paquets en cas de surcharge en entrée. Les services en charge de la transmission de ce flux doivent donc gérer la perte de paquets.
Ces services doivent de plus supporter la possibilité de recevoir des données dupliquées, lors

Soit s le numéro courant de cellule du buffer interne
Soit S le nombre total de cellules du buffer interne
Soit t la durée depuis la dernière émission de trame
Soit T la période inter émission

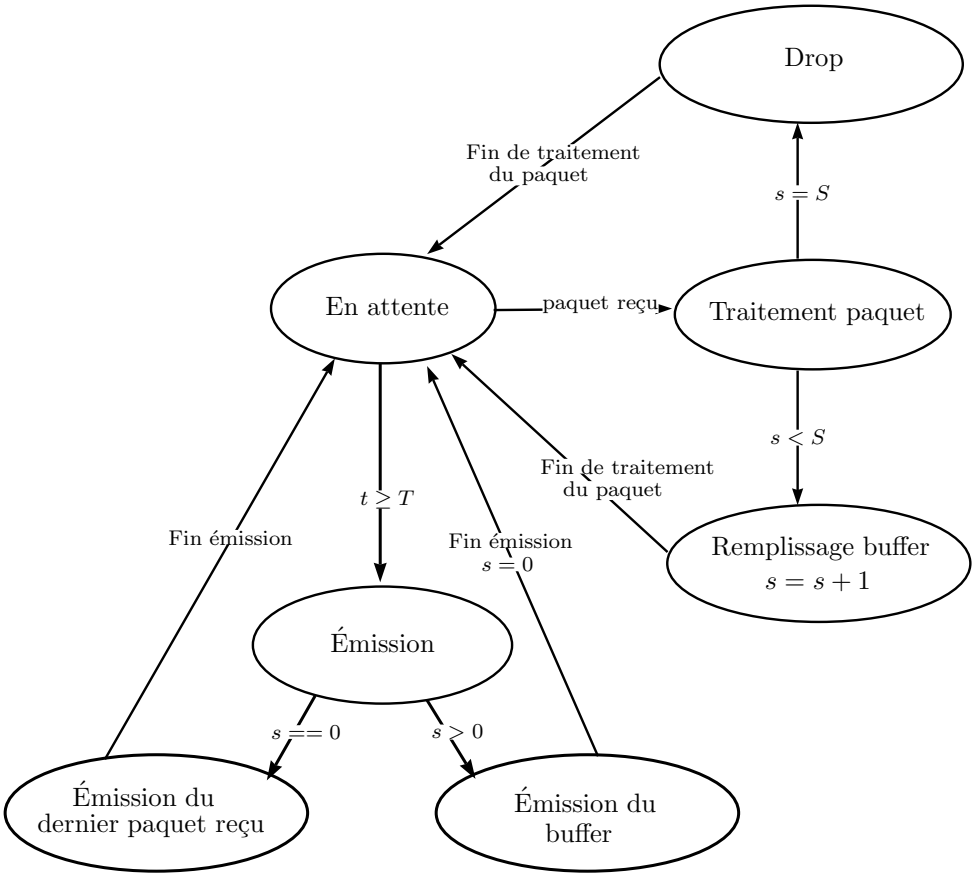

FIGURE III.2.4 – Automate du moniteur de sécurité émetteur périodique

de la réémission de paquet en cas de buffer vide. Bien que ces hypothèses soient fortes, elles correspondent à un certain nombre de flux de signalisation à faible débit utilisés dans les systèmes systroniques entre plusieurs domaines de sécurité. C'est le cas des diverses remontées de sondes comme le positionnement GPS, les données de température ou encore les éléments de positionnement géographique de plusieurs véhicules dans le cadre d'un système de cartographie. Ainsi, ce moniteur peut limiter l'usage de canaux cachés entre deux domaines de sécurité pour un certain nombre de flux spécifiques à la systronique.

III.2.3.3 Conformité des données transmises à la politique de sécurité

Les moniteurs de sécurité de cette section répondent à l'exigence III.2.2.2. Il en existe plusieurs car cette exigence implique plusieurs fonctions, à la fois sur le contenu transmis et sur le comportement du flux, afin d'empêcher l'usage d'un canal auxiliaire de type temporel.

Moniteurs de sécurité pour la vérification des données

Il est habituellement demandé, dans le cadre d'un transfert de données entre deux domaines de sécurité, de prévoir un mécanisme garantissant la validité des contenus à transmettre auprès d'une politique de sécurité prédéfinie. On parle alors d'un filtre dit DPI (Deep Packet Inspection) [Smi+08][Kum+06].

DÉFINITION III.2.3.1
On appelle DPI l'ensemble des fonctions en charge de valider un ou plusieurs flux réseau afin de garantir que son contenu, mais également son comportement (profil, débit, émetteur, destination, etc.) sont conformes d'une politique de sécurité.

Problématique générale du DPI Un tel moniteur de sécurité est plus ou moins complexe selon le besoin. La problématique de cette fonction est la grande difficulté à pouvoir déterminer une borne supérieure au traitement DPI. Une fonction DPI correspondant à un ensemble de traitements unitaires successifs sur un ou plusieurs flux, c'est typiquement une fonction candidate au principe de criticité mixte, afin de pouvoir garantir une borne supérieure à l'exécution de la fonction DPI sur un paquet réseau donné. A ce jour, les travaux de l'état de l'art sur le DPI ne prennent pas en considération la contrainte d'exécution bornée. Dans le cadre de ma thèse je n'ai néanmoins pas étudié de manière détaillée la fonction DPI, cette dernière étant particulièrement complexe est sujet de nombreux travaux qui lui sont dédiés encore aujourd'hui.

Fonction DPI et passerelle multi-niveaux de sécurité Étant donné que la passerelle intègre un mécanisme de séparation protocolaire (décrit dans la Section III.2.2.2), il n'est pas nécessaire, dans le cadre de ce moniteur, de prendre en considération les couches transport et inférieures de la pile TCP/IP. Seule la couche applicative est considérée. Selon le type de protocole de niveau application utilisé dans le cadre du transfert de données, il peut être nécessaire d'attendre la fin de transmission avant de pouvoir valider les données. C'est clairement le cas si l'on fait transiter des données de types XML, trop grosses pour être traitées au travers d'un seul paquet. Il devient nécessaire de dimensionner la mémoire du moniteur de sécurité afin de lui permettre de stocker l'information jusqu'à être apte à la valider auprès d'une politique de sécurité. Un tel stockage implique également un accroissement de la latence de traversée de la passerelle égale au pire cas d'attente de l'ensemble des paquets nécessaire à la validation de l'ensemble du flux.
La complexité du filtre DPI est directement dépendante de la complexité du protocole à traiter et de la politique de sécurité afférente. Je considère donc qu'un filtre DPI est spécialisé pour un protocole donné. Il en va de même pour la politique de sécurité associée pour ce protocole.
Le filtre DPI est directement lié au protocole qu'il traite. Ainsi, une passerelle multi-protocoles implique de considérer un ensemble de filtres DPI. Je considère dans le cadre de ma thèse que le moniteur de sécurité de filtrage DPI gère l'ensemble des filtres DPI associés à l'ensemble

des protocoles devant être transmis. Ces derniers sont regroupés, pour des raisons de performances, dans un seul compartiment, avec une gestion de priorisation et d'ordonnancement qu'il est nécessaire de considérer afin de garantir le bon fonctionnement de l'ensemble des filtres.

Il est également nécessaire de considérer un mécanisme permettant de flasher, dans le cadre d'une préparation de mission, la politique de sécurité du filtre. Ce mécanisme ne doit être actif que durant le cycle de préparation, et débrayé une fois la passerelle en mode de fonctionnement nominal. En général, on s'appuie sur une connexion locale (e.g. via une interface UART) au travers d'un port physique sécurisé une fois placé dans l'enceinte du véhicule. Je n'ai pas pris en compte la problématique de gestion du flashage de politiques de sécurité, cette dernière ayant déjà des réponses dans le cadre des travaux existants dans les radios militaires (sans références dans le cadre de ma thèse car soumis à des exigences de confidentialité fortes).

III.2.3.4 Compartimentation garantie

Il a été décrit dans la Section III.2.1.2 l'usage, dans la passerelle, d'éléments non certifiables (éléments appartenant à l'ensemble $Unsec$). Pour être compatible avec les Exigences REQ_SECUR_001 et REQ_SECUR_005, il est nécessaire d'assurer que l'ensemble des canaux de communications utilisés sont explicitement ceux qui ont été validés dans le cadre de la définition de l'architecture. La compartimentation garantie est faite au niveau du micro-noyau de sécurité (SSK, Secure Separation Kernel), au travers d'un ordonnancement TDM strict avec une séparation spatiale entre chaque compartiment logiciel. Le SSK que j'ai choisi est le micro-noyau PikeOS de la société Sysgo, pour sa compatibilité avec les différentes exigences du MILS, comme décrit dans la Section II.2.1.5. De plus, sa récente certification [Sys13] SIL (Safety Integrity Level - ISO 26262 [Sta12]) niveau 4 fait de lui un bon candidat pour les contraintes véhiculaires.

III.2.4 Problématique d'exploitation des canaux auxiliaires

III.2.4.1 Caches processeurs et canaux cachés

L'usage de canaux auxiliaires basés sur le temps de réponse des accès mémoire au travers des contrôleurs de caches est aujourd'hui connu et efficace pour la transmission de données entre deux compartiments [Per05][Ber05]. Il existe différentes manières de répondre à ce problème. La plus simple consiste à s'appuyer sur des cœurs processeurs différents pour chaque domaine de sécurité, tout en débrayant les mémoires caches partagées entre les cœurs. Selon l'architecture matérielle, la désactivation des mémoires caches partagées est plus ou moins impactante sur les performances du logiciel, au point de rendre ce dernier complètement inefficace. Dans ce cas, l'usage d'une architecture multi-CPU en lieu et place d'un SoC mono-CPU et multi-

cœurs permet d'éviter un partage trop conséquent des contrôleurs de caches. Cependant, dans le cadre de ma thèse, l'usage d'une architecture multi-CPU est incompatible avec les exigences de consommation.
Un autre moyen est de positionner temporellement les domaines de sécurité de manière à ce qu'à un instant t, un et un seul domaine de sécurité s'exécute. Lors du changement de domaine, les mémoires caches sont alors flushées afin de ne pas transmettre d'information. Il est alors nécessaire de prendre en compte l'impact du flush de cache sur l'exécution de la solution. J'ai donc proposé un mécanisme permettant de supporter l'intégration, sur un même cœur du processeur, de plusieurs domaines de sécurité tout en contrecarrant les attaques de type analyse de temps d'accès à la mémoire au travers du contrôleur de cache. La proposition cible exclusivement le contrôleur de cache L1 data. Le cache L1 code et le TLB (Translation Lookaside Buffer), bien que nécessitant également une solution, ne sont pas pris en compte.

III.2.4.2 Hypothèses sur le comportement du contrôleur de cache proposé

Dans le cadre de ma thèse, j'ai travaillé sur un mécanisme de partitionnement du contrôleur de cache L1 data. Je m'appuie sur les principes généraux des caches partitionnés considérés dans le temps réel [CS07], mais pour des besoins sécuritaire.
Ainsi, la solution que je propose a pour but de partitionner le contrôleur de cache en fonction des domaines de sécurité s'exécutant. Le nombre de partition est alors égal au nombre de domaines de sécurité, auquel on ajoute une partition pour le *Secure Separation Kernel*.
Je considère que le domaine de plus faible sécurité est limité à un et un seul compartiment, ordonnancé dans un cœur du processeur où deux domaines de sécurité sont ordonnancés. Le contrôleur de cache n'ayant pas une mémoire extensible, je définit un nombre maximum de slots limité (égal à huit partitions dans mon étude). Chaque slot se comporte de manière autonome avec un nombre de ligne de cache configurable, ceci afin d'avoir une plus grande liberté dans la découpe du contrôleur de cache. Le principe de partitionnement est décrit plus loin. Je prends comme hypothèse que la politique de substitution des lignes de caches de chaque slot reste identique à l'état de l'art. Le nombre de ligne de cache de chacun d'entre eux doit donc être compatible de l'algorithme de substitution considéré. Par exemple, dans le cas d'un algorithme pseudo-LRU de type 4-ways, chaque compartiment doit alors pouvoir fournir un nombre de lignes de caches multiple de quatre.

III.2.4.3 Formalisme de l'interface de contrôle du contrôleur de cache

Le contrôleur de cache fournit jusqu'à huit slots dans lesquels les contextes de caches de chaque domaine de sécurité sont maintenus. Il reste nécessaire de pouvoir configurer le contrôleur de cache afin de pouvoir passer d'un domaine de cache à un autre. Cette configuration doit être faite par un élément de confiance en charge du changement de domaine de sécurité, qui

correspond, dans le cadre de l'architecture logicielle que j'ai choisie, au *Secure Separation Kernel*. Cela implique de la part de la routine de configuration du contrôleur certaines propriétés :

- La routine est positionnée dans un espace mémoire non-cachable, car ne doit pas impacter l'état du contrôleur de cache
- La routine doit être la plus petite possible (pour des raisons de performances)
- La routine doit être exécutée en mode superviseur exclusivement

Une fois la mécanique de partitionnement active, les canaux auxiliaires de type mesure de temps ne doivent plus être utilisables. Les exigences suivantes doivent alors être respectées par le contrôleur de cache :

ÉXIGENCE III.2.4.1
L'usage du contrôleur de cache par une partition logicielle donnée ne doit pas avoir d'impact sur l'état du cache d'une autre partition.

ÉXIGENCE III.2.4.2
Une partition logicielle donnée ne doit pas pouvoir accéder à l'état du cache d'une autre partition logicielle.

ÉXIGENCE III.2.4.3
L'hyperviseur doit être capable de configurer les slots du contrôleur de cache. Cette configuration, qui initie le mode partitionné du cache, doit avoir pour conséquence un flush du cache afin de retirer toute ligne de cache préexistante.

ÉXIGENCE III.2.4.4
L'hyperviseur doit être capable de changer le slot de cache actif. Ce changement doit être rapide et ne doit pas impacter le contenu du slot de cache précédemment actif, ni celui sur le point d'être activé.

ÉXIGENCE III.2.4.5
Les évènements externes asynchrones (type interruption matérielle) doivent être supportés. La routine de gestion de l'interruption ne doit pas impacter la partition de cache en cours d'exécution lors de l'interruption, mais s'exécuter dans la partition de cache dédiée à l'hyperviseur.

ÉXIGENCE III.2.4.6
Le contrôleur de cache doit supporter un mode générique non partitionné, actif par défaut à l'initialisation du contrôleur sans aucune configuration par le logiciel.

III.2.4.4 Description de l'interface de contrôle du partitionnement du contrôleur de cache

Dans cette Section, je décris une proposition d'interface matérielle permettant au logiciel de gérer une décomposition du contrôleur de cache en plusieurs slots indépendants. Cette interface

cherche à être simple d'usage pour l'implémentation logicielle tout en supportant un comportement générique non partitionné par défaut.

III.2.4.5 Principe et automate

Afin de limiter au maximum la taille du code traitant de la configuration du contrôleur de cache, cette dernière a été définie sur une base de quelques registres spécifiques.

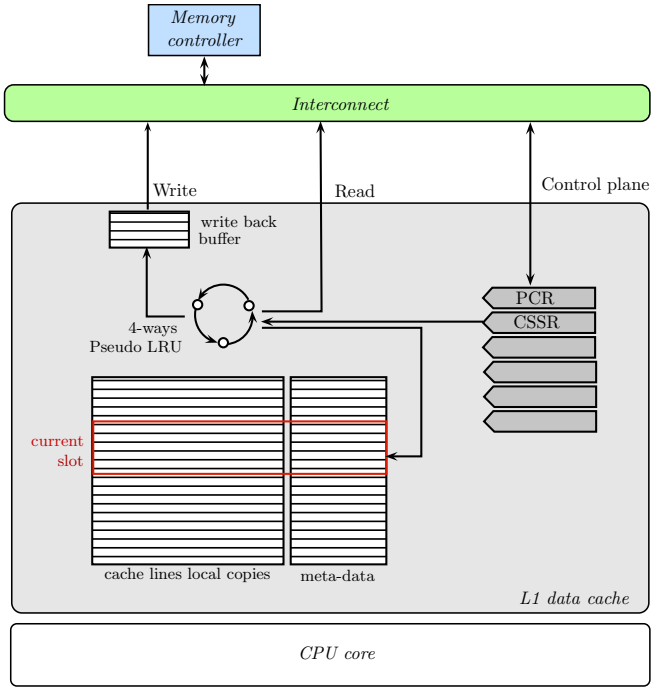

FIGURE III.2.5 – Design d'architecture macroscopique du Contrôleur de cache L1 data proposé

Le contrôleur de cache possède plusieurs états :

- Désactivé. Le contrôleur est alors complètement désactivé et ne fait que transférer les requêtes du processeur vers la mémoire centrale
- configuration générique, le contrôleur se comporte alors comme un contrôleur de cache classique. C'est sa configuration par défaut
- Partitionné. Le contrôleur a été configuré et activé. La partition active est celle qui a été sélectionnée lors de la configuration

Définition des registres de contrôle

J'ai défini un premier registre, que je dénote PCR (*Partition Control Registrer*). Ce dernier est décrit dans la Figure III.2.7

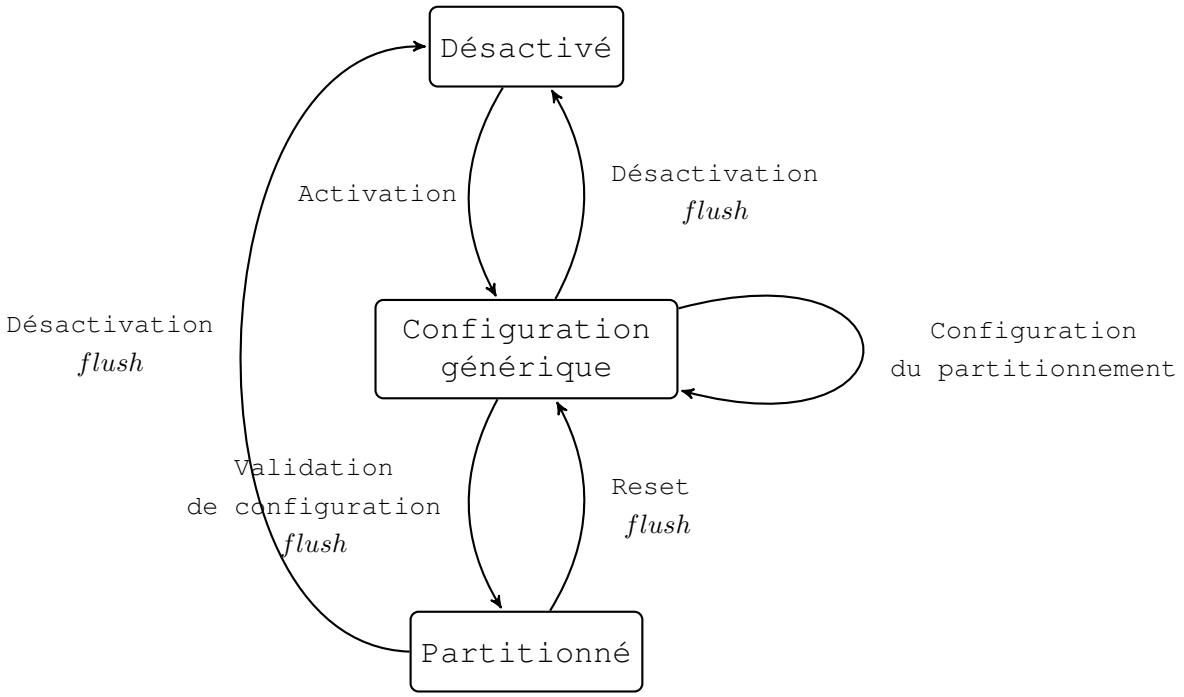

FIGURE III.2.6 – Automate à état du contrôleur de cache

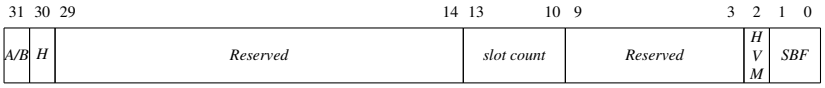

FIGURE III.2.7 – Structure du registre PCR

La description des différents champs du registre PCR est faite dans le Tableau III.2.2.

Le registre PCR permet d'activer et de valider la configuration globale du mode partitionné du cache. Néanmoins, il ne spécifie pas la taille des différents slots. Dans le cadre de ma thèse, je souhaite pouvoir supporter des tailles de slots configurables. En effet, afin de ne pas sous consommer le cache, ou de pouvoir, dans le cadre d'un compartiment logiciel particulier, accroître la taille du slot associé pour des raisons de performances, j'ai rendu la taille de chaque slot configurable. Néanmoins, il existe des restrictions :

- La taille d'un slot (en nombre de lignes de cache) doit être compatible avec l'algorithme de substitution du contrôleur de cache (e.g. pseudo-LRU 4-ways).
- La somme des tailles de slots actifs doit être inférieure ou égale à la taille totale de l'espace mémoire du contrôleur de cache.

La configuration des slots est faite dans les différents registre SCR (Slot Configuration Register). Il en existe 8, pour un nombre de slots maximum de 8.

Chaque registre SCR définit la configuration d'un slot. Sa structure est définie dans le Tableau III.2.4. Par défaut, l'ensemble de ces registres ont une valeur mise à 0x0.

CHAMP	SIGNIFICATION	DROITS
bit A/B	Bit d'activation. Positionné à 0 au boot. Doit être positionné à 1 afin d'activer le mécanisme de partitionnement du contrôleur de cache. L'ensemble de la configuration du contrôleur doit être fait au moment où ce bit est mis à 1. La mise à 1 de ce bit implique un changement d'état du contrôleur de cache de l'état générique à l'état partitionné. L'ensemble des données en cache à cet instant sont flushées.	RW
bit H	Fige la configuration du contrôleur de cache. Si positionné à 1, plus aucune configuration des slots du contrôleur de cache ne peut être effectuée, y compris par l'hyperviseur. Seul le retour en mode générique peut être effectuée, via la mise à 0 du bit A/B. Ce retour implique un flush complet des lignes de cache.	RW
Slot count	Spécifie le nombre de slots qui seront utilisés. L'hyperviseur doit spécifier le nombre de slots configurés dans ce champ, avant de rendre la configuration active. Le nombre de slot doit être inférieur au nombre de slots maximum (définit à huit dans l'étude).	RW
HVM	Active le support HVM (Hardware Virtual Machine). Le but est de permettre une gestion par le mécanisme de gestion de la virtualisation plutôt que par l'hyperviseur. Il s'agit d'une prise en compte pour un usage futur, et n'est pas considéré dans le cadre de ma thèse.	RW
SBF	Ce bit (Status Bit Field) est renseigné par le contrôleur de cache lors du passage en mode partitionné. Une relecture du registre permet de récupérer la valeur de ce champ. Si SBF vaut 00, la configuration est valide et le contrôleur est passé en mode partitionné. Dans le cas contraire, le cache est resté en mode générique. Le status de la configuration est alors accessible dans ce même registre, avec le tableau de correspondance ci-dessous : – `00` : configuration active – `01` : l'espace total consommé est trop grand. Il faut vérifier la taille des slots et leur nombre. – `10` : conflit sur la taille d'un ou plusieurs slots. Le nombre de lignes de caches ne permet pas l'usage de l'algorithme Pseudo-LRU 4-ways (le nombre de lignes de cache doit être un multiple de 4). – `11` : *usage futur*	RO

TABLE III.2.2 – Description des champs du registre PCR

Les deux registres précédents permettent l'activation du mode partitionné et la configuration des différents slots de cache. Néanmoins, il reste à déterminer quel slot est le slot courant, et en informer le contrôleur de cache pour permettre le changement de slot. Afin de permettre ce mécanisme, je propose un registre supplémentaire dédié à ces deux fonctions. Ce registre est nommé CSSR (*Current Slot Selector Register*). Le registre est schématisé dans la Figure III.2.9.

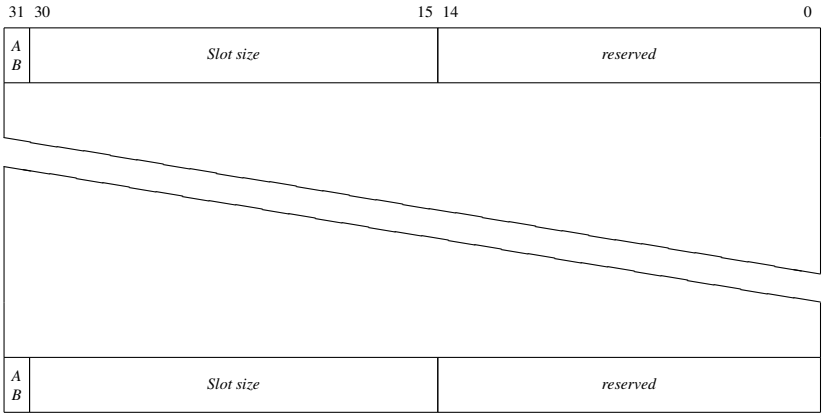

FIGURE III.2.8 – Slot configuration table

CHAMP	SIGNIFICATION	DROITS
AB	Bit d'activation. Sa valeur par défaut est 0. Si l'hyperviseur change sa valeur pour 1, le contrôleur de cache, au moment de la validation de la configuration, considère le slot actif, et utilise alors les valeurs des autres champs pour configurer le slot.	RW
Slot size	Définit le nombre de lignes de caches du slot. Ce nombre doit être un multiple de 4.	RW

TABLE III.2.3 – Description des registre de configuration des slots de cache

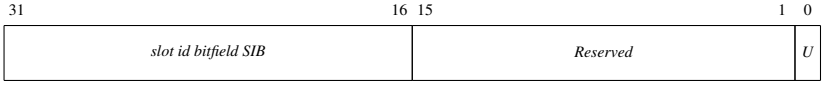

FIGURE III.2.9 – Structure du registre CSSR

La Table III.2.4 décrit les différents champs du registre CSSR.

La configuration initiale du contrôleur de cache doit se faire de la manière suivante :

1. Configuration de chaque SCR
2. Configuration du CSSR, afin de spécifier le slot courant. Le bit *update* doit être mis à 1
3. Configuration du PCR, pour activer le mode partitionné. Si le passage en mode partitionné s'est bien déroulée, le cache est complètement flushé et le slot courant déterminé dans le registre CSSR commence à être utilisé
4. Vérification du champ SBF du registre PCR, afin de valider le passage en mode partitionné

CHAMP	SIGNIFICATION	DROITS
SIB	définit l'identifiant du slot courant. Il s'agit d'un masque où le numéro du slot est calculé par rapport au numéro du bit positionné à 1. Ainsi, soit S_{id} le numéro du slot, la valeur du champ vaut alors : $$SIB = 2^{S_{id}}$$	RW
U	Champ *update*. Quand ce champ est positionné à 1 par l'hyperviseur, le contrôleur de cache relit le registre et change de slot en cours. Ainsi, dans le cadre du fonctionnement nominal du système avec le contrôleur de cache en mode partitionné, seul ce registre nécessite une modification pour changer de slot.	RW

TABLE III.2.4 – Description des champs du registre CSSR

Une fois cette configuration effectuée, le contrôleur de cache utilise donc le slot courant défini dans le registre CSSR. Je considère dans ma thèse trois slots utilisés dans un cas classique de sécurisation :

1. Un slot *hyperviseur*, utilisé par l'hyperviseur et la gestion des pieds d'interruption. Si les interruptions sont ensuite remontées au niveau des machines virtuelles, l'hyperviseur change de slot avant de donner la main à la machine virtuelle. Cela implique de démarrer le pied d'interruption par un changement de slot vers le slot hyperviseur et un changement de slot juste avant l'ordonnancement de la machine virtuelle pour le traitement de la routine d'interruption associée
2. Un slot *domaine de sécurité bas*, pour la ou les machines virtuelles de plus bas niveau de sécurité
3. Un slot *domaine de sécurité haut*, pour la ou les machines virtuelles de plus haut niveau de sécurité

Le changement de slot se fait donc facilement via l'écriture dans un seul registre. Dans le cadre de l'architecture PowerPC, l'écriture pourrait se faire via l'usage de l'instruction `mtspr` [Sem05], dédiée à l'écriture dans les registres de type *Special Purpose Registers*.
Le segment de code en charge de cette écriture ne doit pas être mis en cache, ce dernier impactant le slot courant. Il doit donc être positionnable dans un segment mémoire non-cachable ou placé dans un espace mémoire dédié type scratchpad, dont l'accès est plus rapide est local au cœur processeur.

Dans le cas d'usage que je propose, je n'utilise que trois slots sur les huit présents. Il reste néanmoins possible d'utiliser jusqu'à huit slots dans le contrôleur de cache, afin de supporter une plus grande séparation, en cas d'usage de plus de deux domaines de sécurité ou pour des contraintes temps réel.

Cette proposition a pour but d'éviter le flush de caches en réponse à la problématique d'utilisation de canaux auxiliaires basées sur les contrôleurs de cache. Néanmoins, pour qu'une telle solution soit efficace, il faudrait prévoir un partitionnement pour les caches L1 data, L1 code, mais également le TLB. De plus, le dimensionnement des partitions doit être fait en s'assurant que la consommation en terme de lignes de caches par chacun des domaines de sécurité soit telle que le nombre de lignes de caches qui lui est allouée est suffisante dans le cadre de son usage. Dans le cas inverse, un tel partitionnement augmenterait fortement le nombre de cache-miss, et aurait un impact plus négatif que positif. Ce dimensionnement n'est pas étudié dans le cadre de ma thèse. De plus, si la taille de chaque partition du cache, nécessaire pour éviter un accroissement des cache-miss, est telle qu'il n'est pas possible de déterminer une découpe valide du contrôleur de cache L1, la solution ne peut plus être choisie pour répondre au besoin initial de cloisonnement du contrôleur sans flush explicite. Pour terminer, la problématique des caches L2 et plus globalement des caches partagés reste entière.

Chapitre III.3

Compatibilité d'une architecture sécurisée avec des contraintes de temps réel

Sommaire

III.3.1 Architectures compartimentées et temps réel **103**
III.3.1.1 Rappel des exigences de temps réel 104
III.3.1.2 Hiérarchie et Time Division Multiplexing 104
III.3.2 Moniteurs de sécurité, fonctions temps réel et environnements non sécurisables **107**
III.3.2.1 Compatibilité des moniteurs de sécurité avec le temps réel 107
III.3.2.2 Compatibilité des environnements non certifiables avec le temps réel 107
III.3.3 Intégration du support des flux à criticité mixte **109**
III.3.3.1 Rappels et enjeux 109
III.3.3.2 Modèle et Définitions 110
III.3.3.3 Ordonnançabilité d'un jeu de tâches à criticité mixte 114
III.3.3.4 Retarder l'accroissement de criticité 116
III.3.3.5 Marge de tolérance 117
III.3.3.6 Implémentation de la marge de tolérance 123
III.3.3.7 Réinitialiser la Criticité du système 126
III.3.3.8 Simulations 127
III.3.3.9 Conclusion 130

III.3.1 Architectures compartimentées et temps réel

Afin de répondre aux exigences sécuritaires de la solution, l'usage d'architectures logicielles compartimentées strictement dans le temps est nécessaire. Cette compartimentation, de type TDM (Time Divion Multiplexing) réduit en effet les interactions dues aux variations de comportement (charge processeur, accès mémoire, etc) des différents compartiments.

Lorsqu'il est nécessaire d'intégrer dans le même temps des exigences temps réel sur les flux traité par la passerelle, il devient nécessaire de considérer la bonne adéquation de la solution de compartimentation avec les contraintes d'exécution (WCET, période, échance) des services de traitement de flux.

III.3.1.1 Rappel des exigences de temps réel

En plus des différentes exigences de sécurité décrites dans le chapitre précédent, plusieurs exigences de temps réel doivent être respectées dans le cadre d'une passerelle d'interconnexion. Elle est en effet en charge de faire transiter des flux plus ou moins critiques, sur lesquels elle va exécuter un ensemble de traitements. Ces traitements, selon leur nature, ont également des niveau de criticité variables, impliquant la prise en compte du principe de criticité mixte afin de pouvoir limiter le nombre de traitements effectué sur chaque flux en fonction de la capacité qu'a la passerelle à les exécuter, comme le précise l'Exigence REQ_TEMPS_002.
Dans le même temps, la passerelle doit assurer une latence de traversée bornée, pour que les données transmises soient encore valides (confer Exigence REQ_TEMPS_001).

III.3.1.2 Hiérarchie et Time Division Multiplexing

Les architectures logicielles compartimentées sont ordonnancées de façon similaire aux systèmes temps réel à ordonnancement hiérarchique [Reg+03]. On définit alors un ordonnancement dit *global*, correspondant à l'ordonnancement des compartiments, et un ordonnancement dit *local* correspondant à l'ordonnancement des tâches dans un compartiment donné.
Un *slot* est associé à chaque compartiment. Ce slot est une fraction temporelle périodique pendant laquelle le compartiment est autorisé à être exécuté. L'exécution successive de l'ensemble des slots du système forme un motif temporel strict et prédictible définissant la période TDM.

Dans le cadre de ma thèse, j'ai étudié la problématique d'ordonnancement temps réel des fonctions logicielles de la passerelle sur une base TDM, pour les raisons de sécurité évoquées dans le Chapitre III.2. Les fonctions de traitement des différents compartiments devant respecter des exigences de temps réel s'appuient sur une politique d'ordonnancement de type EDF (Earliest Deadline First). La solution s'appuie donc sur un ordonnancement global TDM avec un ordonnancement local EDF, qui sera noté dans la suite TDM/EDF.

Dans le cadre de cette architecture, il est cependant nécessaire de déterminer, pour l'ensemble des compartiments ayant des tâches devant respecter des exigences de temps réel, l'inter-arrivée et la durée nécessaire du slot permettant d'assurer leurs échéances. La particularité de la solution de compartimentation pour la sécurité sur base TDM est que l'ordonnanceur global, géré au niveau de l'hyperviseur, n'est pas informé en-ligne des propriétés temps réel des tâches hébergées dans les compartiments. De plus, le respect du partitionnement TDM strict interdit

Variables de niveau global	*Variables de niveau local*
Considérant m compartiments, on définit : U_i : la charge associée au compartiment C_i $\tau = \{\tau_1, \ldots, \tau_m\}$ l'ensemble des ensembles tâches associés à C_i ST_i La période du slot S_i SC_i La durée du slot S_i	Considering n tâches, on définit : τ_i l'ensemble de tâches du compartiment C_i, $\tau_i = \{\tau_i^1, \ldots, \tau_i^n\}$ d_i^j l'échéance de la tâche τ_i^j de C_i c_i^j le WCET de la tâche τ_i^j de C_i t_i^j la période de la tâche τ_i^j de C_i

TABLE III.3.1 – Définition des variables de niveau compartiment et de niveau tâches

toute variation du motif d'ordonnancement des compartiments. En conséquence, bien que l'ordonnancement local des tâches d'un compartiment donné puisse être considéré en ligne (c'est le cas de la politique d'ordonnancement EDF), l'ordonnancement global est calculé hors ligne. C'est ce calcul qui est considéré ici.

Plusieurs variables sont introduites dans la Table III.3.1 afin de formaliser l'ordonnançabilité d'une architecture logicelle compartimentée sur une base TDM (Time Division Multiplexing) stricte.

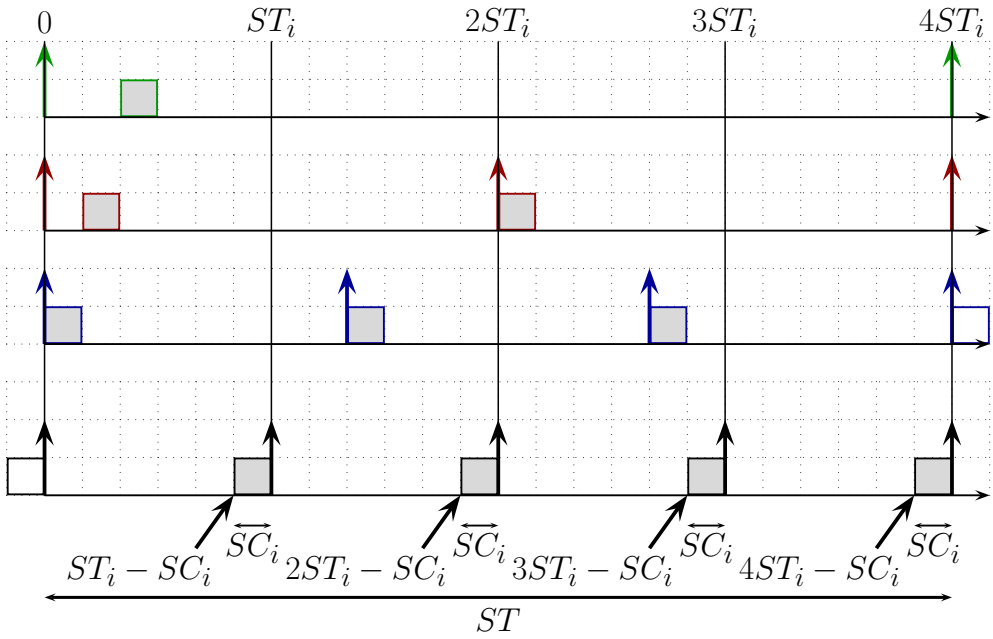

FIGURE III.3.1 – Ordonnancement hiérarchique sur base TDM

Afin de garantir l'ordonnançabilité d'un tel système, plusieurs contraintes doivent être satisfaites. Certaines sont des contraintes *globales* (au niveau de l'ordonnanceur de l'hyperviseur)

Considérant la table III.3.1, soit le problème : *Maximiser* $min_{i=1...,n} SC_i(1-U_i)$ **sous les contraintes :**

contraintes de niveau global

$$\sum_{i=1}^{m} U_i \leq 1 \quad \text{(III.3.1)}$$

$$\forall i \in \{1,\ldots,m\}, \frac{ST}{ST_i} \in \mathbb{N}^* \quad \text{(III.3.2)}$$

$$\sum_{i=1}^{m} SC_i \leq gcd(ST_1,\ldots,ST_m) \quad \text{(III.3.3)}$$

contraintes de niveau local

$$\forall i \in \{1,\ldots,m\}, \forall \tau_i^j \in S_i, d_i^j > ST_i - SC_i + c_i^j \quad \text{(III.3.4)}$$

$$\forall k \in \mathbb{N}^*, \forall t \in [kST_i - SC_i, kST_i],$$

$$h(t) \leq t - k(ST_i - SC_i) \quad \text{(III.3.5)}$$

$$avec\ h(t) = \sum_{j=1}^{n} max(0, 1 + \lfloor \frac{t-d_i^j}{t_i^j} \rfloor) c_i^j,$$

TABLE III.3.2 – Formalisation des contraintes sur l'ordonnancement hiérarchique de tâches en mode TDM/EDF

d'autres *locales* (au niveau de l'ordonnanceur local au compartiment). Ces contraintes sont définies dans la Table III.3.2.

Les contraintes globales sont les suivantes :
L'Equation III.3.1 garantit que la charge processeur est inférieur à 1. L'Equation III.3.2 est une condition nécessaire pour assurer l'usage de slots temporels périodiques. L'Equation III.3.3 est une condition suffisante pour assurer que les slots ne se chevauchent pas, comme définit dans [MS+10].
La Figure III.3.1 décrit un exemple de répartition temporelle entre différents slots.
Il est également nécessaire de considérer une contrainte locale. L'Equation III.3.4 est une condition nécessaire pour garantir que les tâches des compartiments exécutés dans le cadre de slots temporels ont toujours une échéance supérieure à la période d'inactivité associées à ce compartiment. On s'appuie, dans le cadre d'une hiérarchie d'ordonnancement de type TDM/EDF, sur l'Equation III.3.5, qui est une condition nécessaire et suffisante de faisabilité dans le cadre de l'ordonnancement EDF. Cette Équation est démontrée dans la Preuve III.3.1.1.

PREUVE III.3.1.1
Pour démontrer l'Équation III.3.5, considérons le slot i et l'ensemble de tâches exécuté localement avec une politique d'ordonnancement EDF dans ce slot.
Le Slot i est exécuté de manière périodique par l'hyperviseur, avec une période de ST_i unités de temps. Les tâches exécutées dans ce slot le sont pour une durée contigüe de SC_i unité de temps. On commence par déterminer le scénario pire cas en terme de faisabilité pour cet ensemble de tâches. Ce dernier arrive si :

- *Toutes les tâches sont relâchées simultanément (en t_0)*
- *Le slot i a déjà été exécuté entre t_{-SC_i} et t_0, de telle sorte que les tâches ont raté l'échéance du slot et ne pourront être exécutées que lors du prochain slot, soit $ST_i - SC_i$ unités de temps plus tard*

Ce scénario maximise clairement à la fois la charge associée à l'ensemble de tâches du slot et le délai avant l'exécution de cet ensemble de tâches. Le scénario pire cas étant déterminé, il reste ensuite à définir la condition de faisabilié d'un ensemble de tâche ordonnancé par une politique de type EDF dans le slot i dans le cadre de ce scénario.
Toutes les tâches relâchées dans l'intervalle de temps $[0, t]$ avec une échéance absolue inférieure ou égale à t sont ordonnançables et respectent leur échéances si et seulement si elles sont exécutées durant au pire $t - k(ST_i - SC_i)$ unités de temps. Cela se traduit en :
Pour tout entier k avec $k \geq 1$, pour tout t dans l'intervalle $[kST_i - SC_i, kST_i]$, la fonction de demande processeur au temps t, $h(t)$ est inférieure ou égale à $t - k(ST_i - SC_i)$ (condition de l'équation III.3.5)

III.3.2 Moniteurs de sécurité, fonctions temps réel et environnements non sécurisables

III.3.2.1 Compatibilité des moniteurs de sécurité avec le temps réel

Les moniteurs de sécurité sont considérés, dans le cadre de ma thèse, comme ordonnancés directement dans l'hyperviseur. Cela découle du choix de la solution PikeOS [KW07] comme hyperviseur. Ce dernier est en effet apte à virtualiser des environnements complexes tout en positionnant à leurs côtés des threads ordonnancés directement par le micro-noyau. Ces derniers sont ordonnancés avec les contraintes du TDM, mais peuvent donc être considérés comme des tâches temps réel. Leur simplicité et leur faible volumétrie de code permet de déterminer leur coût d'exécution pire cas. Chaque thread étant autonome dans son propre slot TDM, la période correspond alors à la période de récurrence de ce slot et l'échéance correspond à la durée de ce dernier. Cette durée est déterminée à partir du WCET de la fonction.

III.3.2.2 Compatibilité des environnements non certifiables avec le temps réel

Dans le cadre de mes travaux, j'utilise dans les compartiments non certifiables le système d'exploitation GNU/Linux. Ce dernier fournit une grande richesse applicative et une pile réseau très complète. Cependant, GNU/Linux n'est pas un système d'exploitation compatible avec des exigences temps réel dur. En effet, le noyau Linux étant très riche il s'appuie :

- sur des threads noyau ordonnancés avec les politiques d'ordonnancement temps réel de POSIX (SCHED_FIFO et SCHED_RR) qui intègrent un mécanisme complexe de vieillissement. La charge associée à ces threads est difficilement bornable car elle correspond pour la plupart à la conséquence de traitements d'entrées/sorties comme la gestion du disque. Ces threads possèdent néanmoins une affinité CPU, ce qui évite de les migrer

en fonction de la charge. Malheureusement, l'assignation d'un traitement à un coeur processeur plutôt qu'à un autre n'est pas définissable sous la forme d'une configuration, mais s'établit en fonction de la charge courante de chacun des cœurs.

– sur des fonctions logicielles qui génère une surcharge non maîtrisée des traitements via du *vol de cycle* : ces fonctions (nommées *softirq*) ont été intégrées sous formes de codes complémentaires exécutés à la suite de divers services comme les appels systèmes ou les interruptions, afin de permettre une bonne réactivité de certaines fonctions (émission/réception de paquets, timers haute résolution, etc.). Cependant, une telle solution ne permet pas de garantir un coût d'exécution pire cas pour les processus ordonnancés car leur propre exécution est impactée par ces fonctions. Ces dernières étant de plus en plus fréquentes avec les nouvelles versions du noyau, et sont difficiles à dimensionner. Néanmoins, des travaux commencent à être faits pour intégrer leur impact dans le cadre d'analyses d'ordonnançabilité [Lew+07].

Malgré tous ces éléments problématiques pour le temps réel, il existe plusieurs patchs permettant de rétablir un peu de comportement temps réel dans le noyau Linux. Dans le cadre de ma thèse, afin d'avoir un maquettage rapide, je me suis donc appuyé sur la solution Linux-RT, qui intègre les travaux du patch PREEMPT_RT d'Ingo Molnar [KE09]. Cette solution permet de modifier le comportement du noyau Linux en divers points :

– Les threads noyaux ne s'exécutent plus avec une haute priorité temps réel, supprimant la collision de ses derniers avec l'ensemble des tâches temps réel.
– Les *softirqs* ne s'exécutent plus via du vol de cycle, mais exclusivement dans un thread kernel spécifique appelé *ksoftirqd*.

Néanmoins, un démonstrateur plus performant aurait également pu être fait sur une base Xenomai, en s'appuyant sur son module rtNet [TRW09]. Néanmoins, afin d'intégrer un comportement temps réel performant dans un environnement Linux virtualisé sur le micro-noyau PikeOS, il aurait été nécessaire de porter la solution Xenomai sur le micro-noyau PikeOS, travail qui n'a pas été effectué dans le cadre de ma thèse.

Dans le cadre de mes travaux, j'ai dimensionné l'impact de ce patch sur la variation de latence à l'initialisation d'un travail de priorité maximum. Cette mesure permet de déterminer si il est effectivement possible de définir une borne supérieure à cette initialisation.
Pour cela, je me suis appuyé sur de l'outillage de test [Abe+02] pour simuler une forte charge à la fois en terme d'entrées/sorties, d'accès mémoire et de charge processeur. Je décris plus précisément ces différentes mesure dans le Chapitre III.4, dans le cadre d'un maquettage de la solution sur une architecture logicielle basée sur GNU/Linux. Les mesures ont montré qu'il est possible de garantir une borne supérieure à l'instanciation d'une tâche temps réel y compris en cas de forte surcharge de la solution logicielle.

Il est donc possible, selon le besoin, de positionner des tâches temps réel souple dans des compartiments GNU/Linux. Les tâches à contraintes temps réel fortes ne sont cependant pas intégrables dans les compartiments basés sur des systèmes d'exploitation sans propriétés temps réel fortes.

III.3.3 Intégration du support des flux à criticité mixte

III.3.3.1 Rappels et enjeux

Les systèmes systroniques possèdent des exigences impliquant de considérer des niveaux de criticités hétérogènes dans un même ensemble de tâches. La première cause est la présence d'exigences physiques associées à la systronique, qui limite la puissance processeur utilisable dans le cadre de la passerelle. A défaut de pouvoir utiliser un environnement matériel suffisamment puissant pour traiter l'ensemble des tâches temps réel avec des mesures de coût d'exécution pessimistes, je me replie sur les mécanismes de criticité mixte afin d'ordonnancer au mieux l'ensemble des tâches nécessaire aux divers traitements. L'Exigence REQ_TEMP_002, définie dans le Chapitre III.1 est alors respectée.

Ainsi, dans le cadre d'une passerelle multi-niveaux en charge de traitements réseaux, il est possible de classer ces différents traitements en fonction de leur impact fonctionnel, et donc de déterminer une classement en terme de criticité mixte. La Figure III.3.2 décrit ainsi un ensemble de tâches τ en charge de traitements divers sur plusieurs flux. Le niveau de criticité n'est pas porté par le flux lui même mais par le type de traitement apporté par chaque tâche. A défaut de pouvoir s'appuyer sur de la qualité de service réseau, on s'appuie alors sur un ordonnancement à criticité mixte, qui se charge d'assurer le traitement des tâches les plus critiques, y compris en cas de dépassement de coût d'exécution pire cas de la part des tâches plus faiblement critiques.

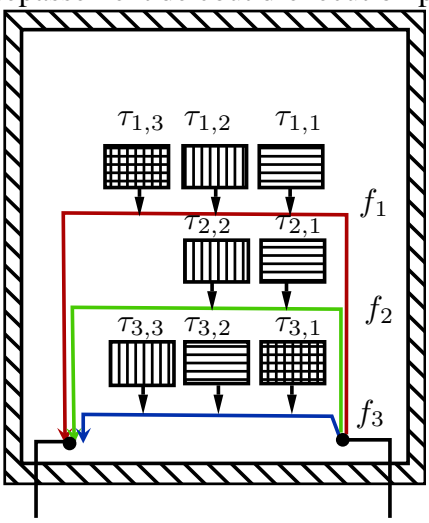

FIGURE III.3.2 – Traitements multi-critiques de flux réseaux

Tache	Niveau de criticité l	flux associé
$\tau_{1,1}$	1	f_1
$\tau_{1,2}$	2	f_1
$\tau_{1,3}$	3	f_1
$\tau_{2,1}$	2	f_2
$\tau_{2,2}$	1	f_2
$\tau_{3,1}$	3	f_3
$\tau_{3,2}$	2	f_3
$\tau_{3,1}$	1	f_3

TABLE III.3.3 – Niveau de criticité des différentes tâches de traitement de flux

Pour rappel, dans le cadre de la criticité mixte, le WCET des tâches de fortes criticité est estimé plusieurs fois, avec des méthodes plus ou moins pessimistes selon le niveau de certification demandé. Plus l'autorité de certification veut garantir le respect du WCET de la tâche, plus la valeur estimée est pessimiste. Cela implique que lors de l'exécution du système, si une autre tâche atteint une estimation de son WCET à un niveau d'assurance qui est supérieur au niveau de criticité de la tâche initiale, cette dernière est alors suspendue, les conditions nécessaires à sa faisabilité n'étant plus respectées [Bar+10]. Néanmoins, cette approche peut être pessimiste, les tâches de plus faible criticité pouvant potentiellement continuer à s'exécuter, ou du moins continuer leur traitement pour quelques temps sans pour autant compromettre les contraintes temporelles des tâches de plus forte criticité.
En conséquence, nous appliquons une stratégie précédemment proposée pour les ensembles de tâches traditionnels [BGM07], habituellement appelée *analyse de sensibilité* et qui permet de définir une marge de tolérance dans le cadre des espaces de tâches à criticité mixte. Nous proposons ensuite une implémentation simple de la marge de tolérance qui s'appuie exclusivement sur les timers, appelée *Latest Execution Time* (LET). Afin d'assurer une bonne continuité de l'exécution de l'ensemble de tâches à criticité mixte, nous montrons également qu'il existe un instant dans l'exécution de l'ensemble des tâches où il est possible d'exécuter à nouveau l'intégralité des tâches.

III.3.3.2 Modèle et Définitions

Dans le cadre de ces travaux, la cible est l'ordonnancement préemptif sur architecture monocœur. La cible de ma thèse associant des mécanismes de compartimentations et d'affinité, je considère alors que les environnements d'exécution nécessitant de la criticité mixte doivent être associés par affinité CPU à un et un seul cœur, afin de rester compatibles avec les hypothèses nécessaires dans le cadre de l'algorithme LET.
Soit un ensemble de tâche $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ où le niveau de criticité maximum d'une tâche est dénoté L. Dans un système à criticité mixte, une tâche à criticité mixte est caractérisé par un 5-uplet $\tau_i = \{r_i, t_i, d_i, \chi_i, C_i\}$ où :

- $r_i \in \mathbb{N}$ est l'instant de réveil du premier travail de la tâche τ_i ;
- $t_i \in \mathbb{N}^*$ est la période de la tâche τ_i ;
- $d_i \in \mathbb{N}^*$ est l'échéance de la tâche τ_i, $d_i \leq t_i$;
- $\chi_i \in \mathbb{N}$ est le niveau de criticité maximum de la tâche τ_i, $\chi_i \leq L$;
- $C_i \in \mathbb{N}^L$ est le vecteur $C_i = \{c_i(1), \ldots, c_i(\chi_i)\}$ de taille L de coûts d'exécution pire cas, où $c_i(\ell)$ est une estimation du WCET de la tâche τ_i au niveau de criticité $\ell \in [1, L]$.

Nous considérons que $c_i(\ell)$ s'accroît de manière monotone pour un accroissement de ℓ. Plus précisément, pour une tâche τ_i :

- $\forall m \in [1, \chi_i[, c_i(m) \leq c_i(m+1)$;
- $\forall m \in [\chi_i, L[, c_i(m) = c_i(\chi_i)$.

Cela suppose qu'aucune tâche ne s'exécute plus longtemps que son WCET pour son niveau courant de criticité. Le k^{n}-ème travail J_k^i instancié par une tâche à criticité multiple τ_i est caractérisé par un 5-uplet $\{r_k^i, d_k^i, \mathcal{X}_k^i, C_k^i, c_k^i\}$ où :

- $r_k^i \in \mathbb{N}$ est l'instant où J_k^i est instancié. Étant donné que l'on considère un ensemble de tâches sporadique, on a $r_k^i \geq r_{k-1}^i + t_i$, et $r_1^i = r_i$;
- $d_k^i \in \mathbb{N}^*$ est l'échéance absolue de J_k^i. Plus précisément : $d_k^i = r_k^i + d_i$;
- $\mathcal{X}_k^i \in \mathbb{N}$ est la criticité de J_k^i, héritée de la tâche τ_i : $\mathcal{X}_k^i = \mathcal{X}_i$;
- $C_k^i \in \mathbb{N}^L$ est la taille L du vecteur de WCETs pour J_k^i, hérité de la tâche τ_i : $c_k^i = c_i$;
- $c_k^i \in \mathbb{N}^*$ est l'exact coût d'exécution du travail J_k^i. Du fait des spécifications de τ_i, on peut dire que $c_k^i \leq c_i(\mathcal{X}_i)$, mais la valeur exacte de c_k^i n'est connue que lorsque J_k^i termine son exécution.

DÉFINITION III.3.3.1

Une tâche τ_i de criticité minimum $\ell + 1$ est sujette à un dépassement d'exécution de niveau ℓ, *si un travail instancié par τ_i dépasse son coût d'exécution pire cas de niveau ℓ.*

A tout instant t, on appelle le $j^{\text{ème}}$ travail J_j^i instancié par la tâche τ_i *disponible* si $t \geq r_j^i$ et J_j^i n'a pas encore terminé son exécution. Le coût d'exécution effectif du travail J_j^i n'est pas mesurable à partir des spécifications de τ_i, mais ne sera connu qu'une fois que J_j^i aura terminé son exécution. A tout instant t on appelle le *scénario* s_i^t de la tâche τ_i comme l'ensemble des coûts d'exécution exacts $\{c_1^i, c_2^i, ..., c_k^i\}$ pour chacun des k travaux instanciés par τ_i qui ont déjà terminé leur exécution à l'instant t. Le scénario de l'ensemble de tâches à criticité mixte τ à l'instant t est défini comme $s^t = \{s_1^t, s_2^t, ..., s_n^t\}$

Le *niveau de criticité* du scénario s^t est défini comme le plus petit entier ℓ pour lequel, pour chaque $s_i^t \in s^t$, $c_k^i \leq c_i(\ell)\, \forall k$ Intuitivement, il représente le plus petit niveau de criticité dans lequel aucune tâche ne dépasse son coût d'exécution pire cas de niveau de criticité correspondant. Si cet entier ℓ n'existe pas, alors le scénario est dit *erroné*, car au moins une tâche dépasse son coût d'exécution pire cas de son niveau de criticité propre ($c_j^i > c_i(\chi_i)$).

DÉFINITION III.3.3.2

Un ordonnancement pour un scénario s^t de niveau de criticité ℓ est faisable si tous les travaux $J_j^i \in s_i^t$, $\forall i$, avec $\chi_i \geq \ell$, terminent leur temps d'exécution c_j^i entre son instanciation et son échéance.

Cette définition implique que l'ordonnancement à criticité mixte n'a pour but que de garantir les contraintes temporelles des tâches dont le niveau de criticité est supérieur ou égal au niveau de criticité du scénario.

DÉFINITION III.3.3.3

Une politique d'ordonnancement en ligne est correcte pour un ensemble de tâches τ si, pour tout scénario non-erroné de τ, la politique génère un ordonnancement faisable.

Du fait qu'une politique d'ordonnancement en ligne ne découvre le coût d'exécution exact des travaux qu'une fois que ces derniers ont terminé leur exécution, le niveau de criticité du scénario ne peut être connu qu'après coup. Les travaux instanciés sont alors ordonnancés. Dès qu'un travail dépasse son coût d'exécution pire cas de niveau $\ell - 1$, la criticité du scénario est accrue au niveau ℓ, les travaux instanciés de criticité inférieure à ℓ sont abandonnés, et les requêtes futures des tâches de criticité inférieure à ℓ ne sont plus considérées.

DÉFINITION III.3.3.4
Un ensemble de tâche de criticité mixte τ est dit ordonnançable si il admet au moins une politique d'ordonnancement en ligne correcte.

La Figure III.3.3 étend le modèle traditionnel des états des tâches [Ose] avec des transitions complémentaires, afin de prendre en considération le mécanisme de suspension de tâche.

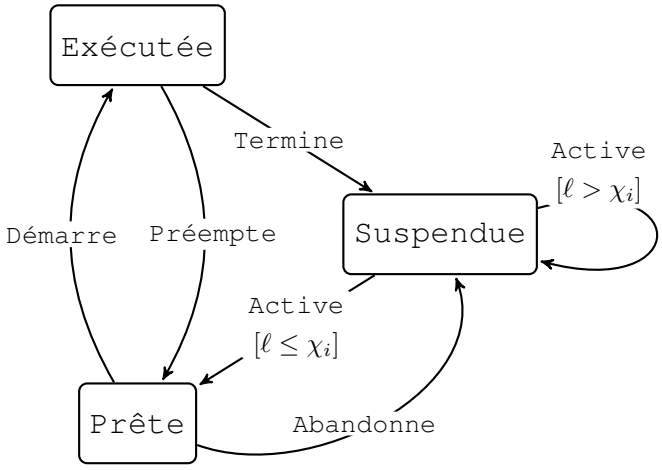

FIGURE III.3.3 – Modèle étendu des états de tâches

Une tâche peut être dans un des états suivants :
- Exécutée : une tâche τ_i est exécutée lorsque l'ordonnanceur choisit τ_i pour l'exécuter. Dans cet état, les instructions de τ_i sont traitées par le CPU. A tout moment, au plus une tâche est dans un état *Exécutée*.
- Prête : une tâche τ_i atteint l'état prêt si tous les prérequis fonctionnels pour une transition vers l'état *Exécutée* sont présents, et τ_i est en attente uniquement d'une élection de la part de l'ordonnanceur.
- Suspendue : une tâche τ_i atteint l'état suspendu si elle n'est actuellement pas candidate à une élection par l'ordonnanceur.

Les transitions ont les significations suivantes :
- Démarre : La tâche est sélectionnée par l'ordonnanceur pour être exécutée.
- Préempte : La tâche est préemptée par une autre tâche.
- Termine : La tâche a complété correctement sont exécution.

- Abandonne : La criticité du système atteint un niveau supérieur à celui de la tâche, qui est en conséquence abandonnée. Cela n'arrive que lorsque τ_i est préemptée par une tâche de priorité et de criticité supérieure.
- Active [$\ell > \chi_i$] : La tâche souhaite instancier un nouveau travail, mais le niveau courant de criticité du système est supérieur au niveau de criticité de la tâche.
- Active [$\ell \leq \chi_i$] : La tâche souhaite instancier un nouveau travail, et le niveau de criticité du système est inférieur ou égal à celui de la tâche.

EXEMPLE III.3.3.1

Considérons un ensemble de tâches à triple criticité ($L = 3$) $\tau = \{\tau_1, \tau_2, \tau_3\}$, avec :

tâche	r_i	t_i	d_i	χ_i	C_i
τ_1	*0*	*5*	*5*	*1*	*(1,1,1)*
τ_2	*0*	*7*	*7*	*2*	*(1,3,3)*
τ_3	*0*	*8*	*8*	*3*	*(1,3,7)*

Tout scénario pour lesquels aucun travail ne dépasse un coût d'exécution de 1 est de criticité 1, tandis que tout scénario pour lequel les tâches τ_2 et τ_3 s'exécutent pour au moins 2 unités de temps et au plus 3 unités de temps est de criticité 2. Si tout travail de la tâche τ_3 s'exécute entre 4 et 7 unités de temps, alors cela définit un scénario de criticité 3. Pour finir, tout autre scénario est erroné.

DÉFINITION III.3.3.5

Le temps de réponse pire cas ϕ_i de la tâche τ_i est la durée maximum pour exécuter la tâche τ_i, en prenant en compte l'exécution de toutes les tâches de priorités supérieures à τ_i. On a $\phi_i \geq c_i(\chi_i)$.

DÉFINITION III.3.3.6

Un instant critique *pour une tâche τ_i est défini comme étant un instant où τ_i instancie un travail dont le temps de réponse sera le plus grand parmi tous les travaux instanciés par τ_i.*

Liu [Liu00] a montré que, lorsque l'on considère un ensemble de tâches traditionnelles ordonnancées en priorité fixe (FP), un instant critique pour une tâche τ_i se présente à chaque fois que τ_i instancie un travail de manière simultanée avec toutes les tâches τ_j de priorité supérieure. Comme montré dans le théorème III.3.3.1, ce principe peut être étendu aux ensembles de tâches à criticité mixte.

THÉORÈME III.3.3.1

Soit $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ un ensemble de tâches sporadiques à criticité mixte ordonnancé avec une politique d'ordonnancement à priorité fixe. L'instant critique d'une tâche τ_i se présente dès lors que τ_i instancie un travail simultanément avec toutes les tâches τ_j de priorité supérieure.

PREUVE III.3.3.1
Sachant que chaque tâche τ_i est certifiée pour un niveau d'assurance qui est égal à son niveau de criticité $\mathcal{X}_i$, on peut associer chaque tâche $\tau_j \in \tau$ à une tâche traditionnelle τ'_j, pour laquelle le coût d'exécution pire cas de τ'_j est égal à $c_j(\mathcal{X}_j)$. En revenant à un ensemble de tâches traditionnel, les résultats démontrés par Liu [Liu00] sont réutilisables, ce qui prouve notre théorème.

III.3.3.3 Ordonnançabilité d'un jeu de tâches à criticité mixte

Baruah et al. [Bar+10] ont démontré que le problème de MC-scheduling, appliqué à un ensemble fini de travaux, est NP-complet. Plutôt que de s'appuyer sur les conditions exactes du *MC-scheduling*, on cible une condition *suffisante* qui peut être vérifiée dans un temps polynomial.
L'algorithme utilisé pour vérifier cette condition, introduit par Vestal [Ves07], détermine hors ligne un ordre total des tâches dans τ. Chaque tâche est assignée à une priorité fixe distincte dont ses travaux héritent. L'assignation des priorités est faite en utilisant l'approche d'Audsley [Aud91], basée sur la définition suivante (la priorité n est la plus basse priorité, la priorité 1 la plus élevée).

DÉFINITION III.3.3.7
Une tâche τ_i dans un ensemble de tâches à criticité mixte τ est dite viable au plus bas niveau de priorité *si toutes les conditions suivantes sont satisfaites :*

1. *la plus basse priorité est assignée à τ_i ;*
2. *à toutes les autres tâches de τ peut être assignée n'importe quelle priorité, du moment que cette priorité est supérieure à celle assignée à τ_i ;*
3. *tout travail instancié par τ_i respecte son échéance lorsqu'il est exécuté pour au plus $c_i(\chi_i)$ unité de temps et toutes les autres tâches $\tau_j \in \tau$ instancient des travaux qui s'exécutent pour au plus $c_j(\chi_i)$ unités de temps.*

L'attribution de priorité est alors construite de manière itérative en utilisant l'algorithme 1. Si un tel ordre de priorité existe, i.e. si une tâche viable est trouvée à chaque niveau de priorité, alors chaque tâche τ_i est garantie de respecter son échéance si elle s'exécute pour au plus $c_i(\chi_i)$ unités de temps et qu'aucune tâche τ_j de priorité supérieure à τ_i s'exécute pendant plus de $c_j(\chi_i)$ unité de temps.

Parce que la priorité d'une tâche est basée sur son niveau de criticité, on dit qu'un ensemble de tâches ordonnançable selon une priorité basée sur la criticité propre des tâches (*Own Criticality Based Priority* (ou *OCPB-schedulability*)) si on peut attribuer à chaque tâche sa priorité fixe en utilisant l'algorithme 1.

Données : τ
pr $\leftarrow |\tau|$
tant que $\tau \neq \varnothing$ **faire**
  si *aucune tâche* $\tau_i \in \tau$ *n'est viable à la priorité la plus basse* **alors**
    retourner *erreur*;
  sinon
    Soit τ_i une tâche viable à la priorité la plus basse;
    assigne τ_i la priorité la plus basse pr;
    $\tau \leftarrow \tau \setminus \{\tau_i\}$;
    pr $\leftarrow$ pr $- 1$;
  finsi
fintq

Algorithme 1: Algorithme d'attribution d'un ordre total aux tâches d'un ensemble à criticité mixe

THÉORÈME III.3.3.2

Si un ensemble de tâches à criticité mixte τ est OCBP-ordonnançable sur un processeur donné, alors τ est ordonnançable sur ce même processeur.

PREUVE III.3.3.2

Cette preuve est principalement inspirée par celle présenté par Baruah et al. [BLS10a]. Soit τ OCBP-ordonnançable, et soit, après renommage des tâches, $\tau_1 \rhd \tau_2 \rhd ... \rhd \tau_n$ représentant un ordre complet des tâches de l'ensemble. Soit τ_i une tâche dans cet ordre de priorité. Afin de démontrer l'ordonnançabilité au sens de la criticité mixte, on doit pouver que chaque travail instancié par la tâche τ_i peut avoir $c_i(\chi_i)$ unités de temps d'exécution entre son instanciation et son échéance dans tout scénario de niveau de criticité χ_i ou inférieur. Mais dans n'importe lequel de ces scénarios, chaque travail instancié par la tâche τ_j s'exécute pendant pas plus de $c_j(\chi_i)$ unités de temps. L'OCBP-ordonnançabilité de τ avec un ordre de priorité $\tau_1 \rhd \tau_2 \rhd ... \rhd \tau_n$ implique que chaque travail instancié par la tâche τ_i aura $c_i(\chi_i)$ unités de temps d'exécution si aucun travail instancié par une tâche $\tau_j \in \{\tau_1, \tau_2, ..., \tau_{i-1}\}$ s'exécute plus de $c_j(\chi_i)$ unités de temps. En conséquence, τ_i respecte effectivement son échéance dans tout scénario de criticité χ_i ou inférieure.

EXEMPLE III.3.3.2

Considérons de nouveau l'ensemble de tâche τ présenté dans l'exemple III.3.3.1. Une assignation de priorité qui rend τ ordonnançable est $\tau_3 \rhd \tau_2 \rhd \tau_1$. Dans ce cas, τ étant un ensemble de tâches périodique à instanciation simultanées, $t = 0$ est un instant critique pour toutes les tâches, ce qui implique qu'il est nécessaire de considérer uniquement le temps de réponse du premier travail de chaque tâche pour déterminer l'ordonnançabilité (dans le cas préemptif et pour des tâches à échéances contraintes). À la tâche τ_1 peut être assignée la plus basse priorité car son temps de réponse ne dépasse pas $3 < d_1$ unités de temps quand aucune des autres tâches ne dépasse son coût d'exécution pire cas de niveau de criticité 1. À la tâche τ_2 peut être assignée la priorité intermédiaire parce que son temps de réponse ne dépasse pas $6 < d_2$ unités

de temps quand la tâche τ_3 ne dépasse pas son coût d'exécution pire cas de niveau de criticité 2. Pour finir, à la tâche τ_3 peut être assignée la plus haute priorité car son temps de réponse ne dépasse pas $7 < d_3$ unités de temps. τ étant OCBP-ordonnançable, on peut alors dire, selon le théorème III.3.3.2, que τ est également ordonnançable au sens de la criticité mixte.

Dans la section suivante, on considère les ensembles de tâches OCBP-ordonnançables. De plus, dans la Section III.3.3.5, les priorités fixes sont considérées comme assignées selon l'algorithme 1.

III.3.3.4 Retarder l'accroissement de criticité

Les politiques d'ordonnancement à criticité mixte traditionnelles arrêtent un travail dès lors que la criticité du scénario dépasse celle du travail. En particulier, dès que le niveau de criticité du scénario atteint ℓ, toutes les tâches τ_i avec $\chi_i < \ell$ sont suspendues, même si le processeur a encore suffisamment de ressources pour pouvoir continuer à exécuter sans impacter les contraintes temporelles des travaux de criticité d'au moins ℓ. Cela est dû à la condition de faisabilité lié à l'OCBP-ordonnançabilité, présentée dans la Section III.3.3.3 qui ne garantit plus les contraintes temporelles.
Ce principe est trop restrictif et peut être amélioré sous certaintes conditions. Dans le cadre d'un ensemble de tâches OCBP-ordonnançables, le lemme suivant nous permet déjà de réduire l'ensemble des tâches à suspendre.

LEMME III.3.3.1
Soit τ un ensemble de tâches à criticité mixte OCBP-ordonnançables, et τ_i n'importe quelle tâche dans l'ordre de priorité définit par l'algorithme 1. Quand un travail instancié par τ_i atteint son coût d'exécution pire cas de niveau de criticité $\ell \leq \mathcal{X}_i$, il n'est jamais nécessaire de suspendre les tâches τ_j avec $\chi_j < \ell$ qui ont une priorité supérieure à τ_i.

PREUVE III.3.3.3
Soit $\tau_i \in \tau$ une tâche qui atteint sont coût d'exécution pire cas de niveau de criticité ℓ. τ étant OCBP-ordonnançable, chaque travail instancié par τ_i peut s'exécuter jusqu'à $c_i(\chi_i)$ unités de temps et respecter son échéance si aucune tâche de priorité supérieure τ_j s'exécute plus que $c_j(\chi_i)$. Soit τ_p une tâche de priorité supérieure, dont le niveau de criticité χ_p est inférieur à χ_i. Étant donné que $\chi_i > \chi_p$, $c_p(\chi_i) = c_p(\chi_p)$, et la condition d'OCBP-ordonnançabilité garantie que chaque travail instancié par τ_i peut s'exécuter jusqu'à son coût d'exécution pire cas de son propre niveau de criticité. En conséquence, seules les tâches de plus basse criticité qui ont une priorité inférieure à τ_i doivent être suspendues.

Le Lemme III.3.3.1 évite la désactivation des tâches de plus basse criticité pour lesquelles les ressources CPU disponibles sont encore suffisantes pour permettre la terminaison de ces

tâches. Nous pourrions néanmoins aller encore plus loin et autoriser l'exécution de tâches de criticité inférieure tant que les travaux que ces dernières sont aptes à respecter leur échéance sans remettre en cause celles des tâches plus critiques. Cela revient à déterminer combien de temps les tâches τ_i de criticité d'au moins $\ell + 1$ peuvent supporter la suspension de tâches de niveau de criticité inférieur en cas de dépassement d'exécution de niveau ℓ. On appelle cette durée la *marge de tolérance de niveau* ℓ de la tâche τ_i, dénotée $A_i(\ell)$ En se basant sur cette marge de tolérance, on peut déterminer le dernier instant au delà duquel la criticité doit être augmentée et les tâches de criticité inférieure ne peuvent plus être assurées d'être exécutées à temps pour respecter leur échéance.

III.3.3.5 Marge de tolérance

La marge de tolérance dérive du concept d'analyse de sensibilité. L'analyse de sensibilité a été étudié par Bini et al. [BB04 ; BDNB06]. Ils discutèrent du concept de domaine de faisabilité qui nous permet de déterminer l'espace des valeurs possibles pour un paramètre spécifique d'une tâche. Le principe de marge de tolérance a été initialement introduit par Bougueroua et al. [BGM07] dans le contexte des ensembles de tâches traditionnels sujets au risque de dépassement de WCET. Leur travail a ciblé le calcul de la plus grande valeur qui peut être ajoutée au WCET d'une tâche sans pour autant impacter l'ordonnançabilité de l'ensemble de tâches, en considérant un modèle de fautes par fenêtre temporelle glissante.

Dans le contexte de la criticité mixte, la marge de tolérance de niveau ℓ d'une tâche τ_i dont le niveau de criticité est d'au moins $\ell + 1$ se base sur le calcul de la marge de son coût d'exécution pire cas de niveau ℓ. Cela représente le coût d'exécution maximum qui peut être ajouté à $c_i(\ell)$ sans impacter les contraintes temporelles des tâches de criticité inférieure τ_j si elles s'exécutent jusqu'à $c_j(\chi_j)$ unités de temps et τ_i s'exécute jusqu'à $c_i(\ell) + A_i(\ell)$ unité de temps. La marge de tolérance de niveau ℓ des tâches pour lesquelles la criticité est inférieure ou égale à ℓ est alors mise à zéro.

Concepts et Notations

Au travers de cette section, les concepts et notations suivantes sont utilisés :

- $U_\ell = \sum_{\tau_p \,|\, \chi_p \geq \ell} \frac{c_p(\ell)}{t_p}$ est le facteur d'utilisation processeur au niveau de criticité ℓ.
- FP_{MC} dénote l'algorithme préemptif à priorité fixe (*Fixed Priority Highest Priority First*), avec une assignation des priorités déterminée par l'algorithme 1, et qui ordonnance uniquement les tâches de criticité supérieure ou égale à la criticité courant du système.
- Un modèle de faute dénoté $\frac{k}{W}$. Cela correspond, pour chaque niveau de criticité $\ell \in [0, L]$, d'au moins k tâches ($0 \leq k \leq n$), au risque de dépassement d'exécution de niveau de criticité ℓ de k tâches dans une fenêtre temporelle glissante de taille W. On définit

$W \leq \min_i(t_i)$ pour éviter à une tâche d'être sujette à de multiple dépassements de coût d'exécution pire cas de niveau ℓ dans une même fenêtre temporelle. Néanmoins, cela peut être trop pessimiste pour supposer que chaque tâche de criticité au moins égale à $\ell + 1$ sera sujette à un dépassement d'exécution dans la même fenêtre. La valeur de k est associée à un ensemble de tâches et peut être obtenue de manière statistique en observant le véritable nombre de dépassements d'exécution de niveau de criticité ℓ durant l'exécution du système, ou au travers de contraintes de certification.

- Pour toute tâche τ_i ordonnancée avec FP_{MC}, et un modèle de faute $\dfrac{k}{W}$:
  - $\text{hp}(i)$: l'ensemble des tâches τ_j ayant une priorité supérieure à τ_i ;
  - $\text{hp}^R(i)$: l'ensemble de travail disponibles instanciés par les tâches $\tau_j \in \text{hp}(i)$;
  - $\text{hp}_{k-1}(i, \ell)$: l'ensemble d'au plus $k - 1$ tâches τ_j ayant une priorité supérieure à τ_i, et une criticité $\chi_j > \ell$. Si plus de $k - 1$ tâches satisfont ces conditions, on choisit les tâches les plus récurrentes, i.e. les $k - 1$ tâches ayant les périodes les plus petites ;
  - $\text{lp}(i, \ell)$: l'ensemble de tâches τ_j ayant une priorité inférieure à τ_i et une criticité $\chi_j \geq \ell$.
  - $\text{lp}^R(i, \ell)$: l'ensemble des travaux disponibles instanciés par les tâches $\tau_j \in \text{lp}(i, \ell)$;
  - L'équation suivante permet de calculer la borne supérieure du temps de réponse pire cas ϕ_i de τ_i :

$$\phi_i = c_i(\chi_i) + \sum_{\tau_p \in \text{hp}(i)} \left\lceil \frac{\phi_i}{t_p} \right\rceil \times c_p(\chi_i) \tag{III.3.6}$$

Par la suite, on considère une distribution uniforme de la marge de tolérance de niveau ℓ entre toutes les k tâches de criticité au moins $\ell + 1$, i.e. chaque tâche de criticité au moins égale à $\ell + 1$ possède une marge de tolérance de niveau ℓ identique.

PROPOSITION III.3.3.1

Soit τ un ensemble de tâches à criticité mixte ordonnançable. Il en découle que pour chaque niveau de criticité ℓ, la condition suivante est satisfaite :

$$U_\ell \leq 1 \tag{III.3.7}$$

Marge de tolérance en priorité fixe

Le théorème suivant décrit comment calculer la marge de tolérance de niveau ℓ d'une tâche τ_i dans le cadre d'un ordonnancement à priorité fixe. Les priorités assignées aux tâches sont celles qui répondent au conditions d'OCBP-ordonnançabilité. Le calcul de $A_i(\ell)$ est basé sur le temps de réponse pire cas des tâches avec la marge de tolérance intégrée.

THÉORÈME III.3.3.3

Soit $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ un ensemble de n tâches périodiques à criticité mixte ordonnancé en priorité fixe, et soit $\tau_1 \rhd \tau_2 \rhd ... \rhd \tau_n$ l'ordre de priorité qui correspond à une OCBP-ordonnançabilité. La marge de tolérance de niveau ℓ qui peut être garantie pour une tâche τ_i de criticité au moins égale à $\ell + 1$, avec une distribution uniforme de la marge de tolérance,

et un modèle de faute $\frac{k}{W}$, est la valeure positive minimum de $A_i(\ell)$ satisfaisant les équations suivantes :

$$c_i(\ell) + A_i(\ell) \leq c_i(\chi_i) \tag{III.3.8}$$

$$U_\ell + \sum_{\tau_p \in \mathrm{hp}_{k-1}(i,\ell) \cup \tau_i} \frac{A_i(\ell)}{t_p} \leq 1 \tag{III.3.9}$$

$$\begin{aligned} &\forall \tau_j \in \mathrm{lp}(i,\ell) : \\ &\phi_j = c_j(\chi_j) + \sum_{\tau_p \in \mathrm{hp}(j)} \left\lceil \frac{\phi_j}{t_p} \right\rceil \times c_p(\chi_j) \\ &\qquad + \sum_{\tau_p \in \mathrm{hp}_{k-1}(i,\ell) \cup \tau_i} \left\lceil \frac{\phi_j}{t_p} \right\rceil \times A_i(\ell) \leq d_j \end{aligned} \tag{III.3.10}$$

PREUVE III.3.3.4

L'équation III.3.8 garantit que la marge de tolérance n'autorise pas une tâche à s'exécuter pour plus que son coût d'exécution pire cas de niveau de criticité ℓ modifié pour prendre en compte la consommation de la marge de tolérance de niveau ℓ d'au plus k tâches. On veut s'assurer que les tâches de criticité inférieure puissent continuer à respecter leur échéance en cas de dépassement d'exécution d'au plus k tâches de criticités supérieures pour une durée limitée par la marge de tolérance, ce qui nous mène à l'équation III.3.10. Cette équation représente le temps de réponse pire cas d'une tâche $\tau_j \in lp(i,l)$ lorsqu'elle s'exécute à son plus haut niveau de criticité, les tâches de plus haute criticité τ_p incluant τ_i s'exécutant pendant au plus $c_p(\chi_j)$ unités de temps et k tâches de priorité supérieure τ_r incluant τ_i consommant leur marge de tolérance. Cette équation ne doit être vérifiée que pour les tâches ayant une priorité inférieure à τ_i du fait du Lemme III.3.3.1.

Il en suit que les tâches de criticité ℓ ne doivent pas être suspendues tant qu'une tâche de priorité supérieure τ_i de criticité $\chi_i > \ell$ ne s'exécute pas plus longtemps que $c_i(\ell) + A_i(\ell)$ unités de temps.

Marge de tolérance avec EDF

Le principe de la marge de tolérance, initialement introduit par Bougueroua et al. [BGM07], a été appliqué à la fois dans les politiques d'ordonnancement à priorité fixe et dans la politique d'ordonnancement EDF, pour les ensembles de tâches traditionnels. Néanmoins, comme le déclare la proposition suivante, la stratégie EDF (Earliest Deadline First) n'est pas optimale pour les ensembles de tâches à criticité mixte.

PROPOSITION III.3.3.2

Tout ensemble de tâches à criticité mixte τ qui est ordonnançable au sens de la criticité mixte n'admet pas nécessairement EDF comme une politique d'ordonnancement en ligne correcte. En d'autres termes, EDF n'est pas optimal pour les ensembles de tâches à criticité mixte.

PREUVE III.3.3.5

Soit τ un ensemble de tâches à criticité mixte qui a été prouvé comme ordonnançable au sens de la criticité mixte dans l'exemple III.3.3.2. La Figure III.3.4 montre que la criticité du scénario n'est pas accrue suffisamment pour garantir les contraintes temporelles de la tâche τ_3 quand toutes les tâches s'exécutent avec leur coût d'exécution pire cas de leur propre niveau de criticité.

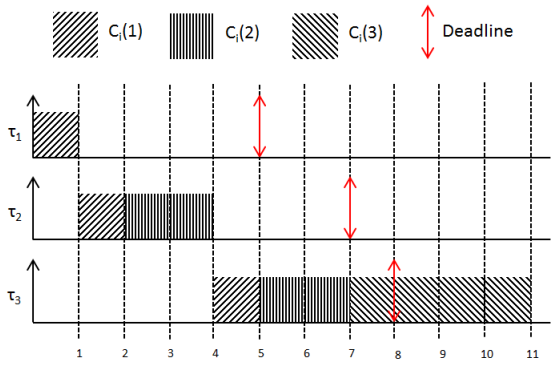

FIGURE III.3.4 – Non Optimalité de l'algorithme EDF

Etant donné que cette étude cible les ensembles de tâches τ qui sont ordonnançables en priorité fixe selon le principe d'*OCBP-schedulability*, et parce que cette propriété ne permet pas de déterminer si Earliest Deadline First est une politique d'ordonnancement correcte pour τ, nous n'avons pas implémenté de principe de marge de tolérance pour l'algorithme EDF.

Domaine de marge sur les WCETs

La Section III.3.3.5 a décrie comment calculer une marge de tolérance de niveau ℓ équitablement partagée pour un ensemble de tâches ordonnancé en priorité fixe. Néanmoins, nous souhaitons utiliser une autre règle d'allocation pour la marge de tolérance de niveau ℓ. Dans cette Section, nous présentons une manière de représenter toutes les valeurs possibles pour la marge de tolérance de niveau ℓ, en caractérisant l'espace des marges de tolérance de niveau ℓ faisables.

DÉFINITION III.3.3.8

Soit $A(\ell) = \{A_1(\ell), A_2(\ell), ..., A_n(\ell)\}$ l'ensemble de marges de tolérances pour un ensemble de tâches à criticité mixte τ. On appelle domaine de faisabilité de l'espace $A(\ell)$ l'ensemble des valeurs de $A(\ell)$ pour lesquelles τ reste faisable si chaque tâche $\tau_i \in \tau$ consomme jusqu'à $A_i(\ell)$ unité de temps supplémentaires au niveau de criticité ℓ.

Bini et al. [BB04] ont introduit une condition d'ordonnançabilité pour les ensembles de tâches périodiques. Nous avons adapté cette condition aux ensembles de tâches à criticité mixte dans le théorème suivant :

THÉORÈME III.3.3.4

Un ensemble de tâches à criticité mixte τ est ordonnançable en priorité fixe si et seulement si :

$\forall i = 1, ..., n,\ \exists t \in \mathrm{schedP}_i$ *tel que*

$$c_i(\chi_i) + \sum_{\tau_p \in \mathrm{hp}(i)} \left\lceil \frac{t}{t_p} \right\rceil c_p(\chi_i) \leq t \quad \text{(III.3.11)}$$

Où schedP_i *est un ensemble de points d'ordonnancement définis comme* $\mathrm{schedP}_i = \mathcal{P}_{i-1}(d_i)$, *et* $\mathcal{P}_i(t)$ *définit comme suit :*

$$\begin{cases} \mathcal{P}_0(t) \\ \mathcal{P}_i(t) = \mathcal{P}_{i-1}\left(\left\lfloor \frac{t}{t_i} \right\rfloor\right) \cup \mathcal{P}_{i-1}(t) \end{cases} \quad \text{(III.3.12)}$$

En utilisant une notation compacte et des opérateurs logiques, l'Equation III.3.11 peut être réécrite :

$$\bigwedge_{i=1,...,n} \bigvee_{t \in \mathrm{schedP}_i} n_i \cdot C_i \leq t \quad \text{(III.3.13)}$$

où $n_i = \left(\left\lceil \frac{t}{t_1} \right\rceil, \left\lceil \frac{t}{t_2} \right\rceil, ..., \left\lceil \frac{t}{t_{i-1}} \right\rceil, 1\right)$ *et* $C_i = (c_1(\chi_i), c_2(\chi_i), ..., c_i(\chi_i))$

Sachant que nous souhaitons représenter le domaine de faisabilité de l'espace $A(\ell)$, nous avons besoin d'introduire les variables de la marge de tolérance. L'Équation III.3.13 doit être étendue afin de considérer une marge de tolérance de niveau ℓ pour chaque tâche de criticité au moins égale à ℓ.

THÉORÈME III.3.3.5

Soit τ un ensemble de tâches à criticité mixte. Le domaine de faisabilité de la marge sur les WCETs de niveau ℓ est calculé comme suit :

$$\bigwedge_{\tau_i \,|\, \chi_i \geq \ell} \bigvee_{t \in \mathrm{schedP}_i} \left(n_i \cdot C_i^\ell \leq t \wedge A_i(\ell) \leq c_i(\chi_i) - c_i(\ell)\right) \quad \text{(III.3.14)}$$

où $C_i^\ell = (c_1(\ell) + A_1(\ell), c_2(\ell) + A_2(\ell), ..., c_i(\ell) + A_i(\ell))$.

PREUVE III.3.3.6

L'Equation III.3.14 étend l'Equation III.3.13 en autorisant chaque tâche à consommer jusqu'à son coût d'exécution pire cas au niveau de criticité ℓ plus sa marge de tolérance, et empêche la marge de tolérance d'autoriser une tâche à dépasser son coût d'exécution pire cas à son propre niveau de criticité.

EXEMPLE III.3.3.3

Calculons maintenant les espaces de tolérance $A(\ell)$ pour $\ell = 1, 2, 3$ de l'ensemble de tâches à criticité mixte présenté dans l'Exemple III.3.3.1. Commençons par renommer les tâches en fonction de l'assignation de priorité. La tâche τ_3 devient la tâche τ_1 et la tâche τ_1 devient la tâche τ_3. La tâche τ_2 reste inchangée.

- *l'espace de tolérance $A(1)$: L'ensemble de points d'ordonnancement à considérer est donné dans la Table III.3.4.*

τ_i	schedP_i
τ_1	$\{8\}$
τ_2	$\{7\}$
τ_3	$\{5\}$

TABLE III.3.4 – Points d'ordonnancement dans l'espace $A(1)$

L'espace de tolérance $A(1)$ est alors représenté par l'ensemble d'équations III.3.15 et est schématisé dans la Figure III.3.5, avec $A_3(1) = 0$.

$$\begin{cases} A_1(1) \leq 7 \wedge A_1(1) \leq 6 \\ A_2(1) + A_1(1) \leq 5 \wedge A_2(1) \leq 2 \\ A_3(1) + A_2(1) + A_1(1) \leq 2 \wedge A_3(1) \leq 0 \end{cases} \tag{III.3.15}$$

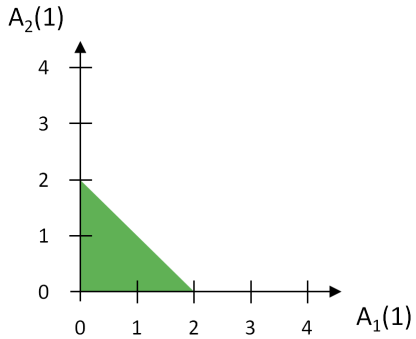

FIGURE III.3.5 – Espace de tolérance $A(1)$

- *Espace de tolérance $A(2)$: L'ensemble de points d'ordonnancement à considérer est donné dans la Table III.3.5.*

τ_i	schedP_i
τ_1	$\{8\}$
τ_2	$\{7\}$

TABLE III.3.5 – Points d'ordonnancement dans l'espace $A(2)$

L'espace de tolérance $A(2)$ est alors représenté par l'ensemble d'équations III.3.16.

$$\begin{cases} A_1(2) \leq 5 \wedge A_1(2) \leq 4 \\ A_2(2) + A_1(2) \leq 1 \wedge A_2(2) \leq 0 \end{cases} \tag{III.3.16}$$

menant à $A_1(2) = 1$ et $A_2(2) = 0$.

– *Espace de tolérance $A(3)$: L'ensemble de points d'ordonnancement à considérer est donné dans la Table III.3.6.*

τ_i	schedP_i
τ_1	$\{8\}$

TABLE III.3.6 – Points d'ordonnancement dans l'espace $A(3)$

L'espace de tolérance $A(3)$ est alors représenté par l'équation III.3.17. Comme on peut le voir, aucune marge de tolérance strictement positive ne satisfait cette équation, entraînant $A_1(3) = 0$.

$$A_1(3) \leq 1 \wedge A_1(3) \leq 0 \tag{III.3.17}$$

III.3.3.6 Implémentation de la marge de tolérance

Nous présentons maintenant un mécanisme en-ligne pour la gestion de la marge de tolérance qui peut être implémenté en utilisant exclusivement des timers. Nous appellerons ce mécanisme LET (*Latest Execution Time*).

DÉFINITION III.3.3.9

Soit t_i l'instant de requête du $k^{ième}$ travail instancié par la tâche τ_i, avec $\chi_i \geq \ell$. $\mathsf{LET}_i^{\ell}(t_i)$ est défini comme le moment d'exécution au plus tard J_k^i en cas de dépassement de coût d'exécution pire cas de niveau ℓ, jusqu'auquel il n'est pas nécessaire de désactiver des tâches de criticité inférieure pour respecter les contraintes temporelles des tâches de criticité supérieure ou égale à ℓ.

Nous montrons maintenant comment calculer le LET d'une tâche τ_i, et ce LET sera utilisé pour déterminer l'instant à partir duquel les tâches de criticité inférieure nécessitent d'être désactivées.

DÉFINITION III.3.3.10

Soit J_k^i le $k^{ième}$ travail instancié à l'instant t_i par la tâche τ_i avec une criticité $\chi_i \geq \ell$. L'instant d'exécution au plus tard (LET) pour J_k^i au niveau de cricité ℓ est calculé à l'instant t_i comme

suit :

$$\forall \ell \in [1, L] :$$
$$\mathsf{LET}_i^\ell(t_i) = \max\left(t_i, \max_{\tau_j \in \mathrm{hp}^R(i)} \left(\mathsf{LET}_j^\ell(t_j)\right)\right) + c_i(\ell) + A_i(\ell) \tag{III.3.18}$$

L'instant d'exécution au plus tard des travaux J de $\mathrm{lp}^R(i, \ell)$ est alors calculé comme suit :

$$\forall J \in \mathrm{lp}^R(i, \ell) : \mathsf{LET}_j^\ell(t_j) = \mathsf{LET}_j^\ell(t_j) + c_i(\ell) + A_i(\ell) \tag{III.3.19}$$

Basé sur le $\mathsf{LET}_i^\ell(t_i)$ d'un travail dont le niveau de criticité est supérieur à ℓ, on peut initialiser le timer en charge de détecter un dépassement d'exécution pire cas de niveau ℓ. Lors de la terminaison de ce dernier, toutes les tâches de criticité inférieure à ℓ et non encore terminées doivent être désactivées. Nous n'initialisons pas de timer pour les travaux dont la criticité est égale à ℓ étant donné que nous ne serons pas sujet à un dépassement de coût d'exécution de niveau ℓ. Nous devons cependant calculer leur LET quand même, pour calculer le LET des autres travaux.

EXEMPLE III.3.3.4

L'implémentation de la marge de tolérance est illustrée en utilisant l'Exemple III.3.3.1, et il est assumé que toutes les tâches de criticité au moins égale à $\ell + 1$ peuvent être sujettes à un dépassement de coût d'exécution de niveau ℓ. La marge de tolérance et l'instant d'excution au plus tard du premier travail de chaque tâche calculés, ces derniers sont représentés dans la Table III.3.7.

Task	$A_i(1)$	$A_i(2)$	$\mathsf{LET}_i^1(0)$	$\mathsf{LET}_i^2(0)$	$\mathsf{LET}_i^3(0)$
τ_3	1	1	2	4	7
τ_2	1	0	4	7	$\times$
τ_1	0	0	5	$\times$	$\times$

TABLE III.3.7 – Exemple de trois tâches ordonnancées en priorité fixe

La Figure III.3.6 illustre un scénario pour lequel à la fois τ_3 et τ_2 dépassent leur coût d'exécution pire cas de niveau de criticité 1, mais sans nécessiter d'interruption de la tâche τ_1, qui possède encore assez de temps pour terminer sa propre exécution. Si le mécanisme de marge de tolérance n'avait pas été utilisé, la criticité du scénario aurait été augmentée à l'instant 1, empêchant la tâche τ_1 de terminer son exécution.

De manière similaire, la Figure III.3.7 illustre un scénario où τ_3 dépasse son coût d'exécution pire cas de niveau de criticité 2, mais termine son exécution suffisamment tôt pour éviter l'interruption de la tâche τ_2, qui possède encore assez de temps pour terminer son exécution

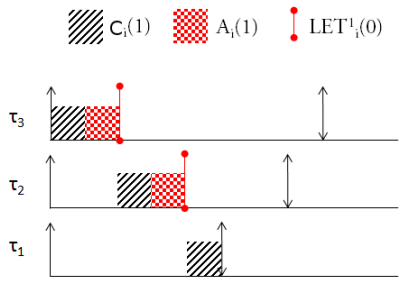

FIGURE III.3.6 – Exemple de $\mathsf{LET}_i^1(0)$

pour une durée égale à son coût d'exécution pire cas de son propre niveau de criticité. Si le principe de marge de tolérance n'avait pas été utilisé, la criticité du scénario aurait été augmentée à l'instant 3, empêchant la tâche τ_2 de terminer son exécution.

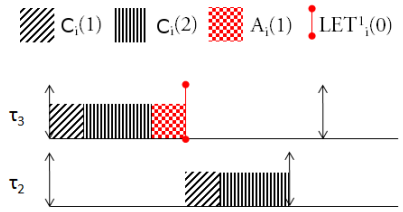

FIGURE III.3.7 – Exemple de $\mathsf{LET}_i^2(0)$

Pour finir, la Figure III.3.8 illustre la récaupération de la marge de tolérance. En effet, la tâche τ_3 avait une marge de tolérance de une unité de temps, mais a pu terminer son exécution avec un coût inférieur à son WCET du niveau de criticité courant. Cela a permis à la tâche τ_2 de profiter de deux unités de temps complémentaires pour terminer son exécution sans pour autant nécessiter l'interruption de la tâche τ_1.

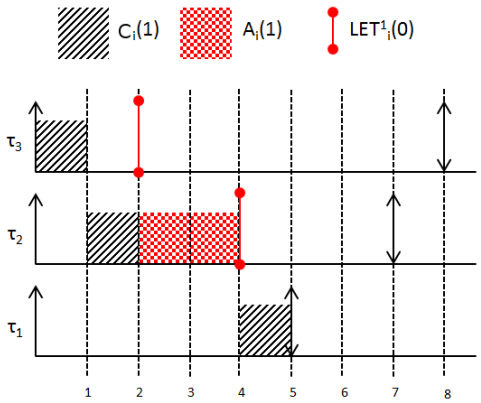

FIGURE III.3.8 – Récupération de la marge de tolérance

III.3.3.7 Réinitialiser la Criticité du système

Le mécanisme présenté en Section III.3.3.6 nous permet d'accroître le délai de suspension d'une tâche de criticité plus faible le plus possible, empêchant de désactiver des travaux qui auraient pu avoir suffisamment de temps pour terminer leur exécution. Néanmoins, dans certains cas et à un niveau de criticité donné, une tâche peut nécessiter plus de temps pour terminer son exécution que la marge de tolérance ne lui permet. Dans ce cas, le niveau de criticité du scénario a besoin d'être augmenté.
Dans cette Section, nous étudions la pertinence de conserver les tâches de priorité inférieure suspendues. Nous pensons que, si il est nécessaire de suspendre des tâches de priorité inférieure pour garantir les contraintes temporelles de tâches de criticité supérieure, cette décision reste réversible, et qu'il existe un instant où toutes les tâches peuvent de nouveau s'exécuter en concurrence, quel que soit leur niveau de criticité.

Au travers de cette Section, nous pensons à l'inverse de ce que le Lemme III.3.3.1 suggère, à savoir que dès que le scénario atteint le niveau de criticité ℓ, toutes les tâches de criticité inférieure nécessitent d'être désactivées. Les définitions et résultats que nous présentons peuvent néanmoins être adaptés pour prendre en compte le Lemme III.3.3.1.

DÉFINITION III.3.3.11
Si le scénario courant est de niveau de criticité ℓ, un instant d'inactivité de niveau ℓ (level-l idle time) t sur le processeur est défini comme un instant durant lequel aucun travail de criticité au moins égale à ℓ n'est disponible, en attente d'ordonnancement à l'instant t.

THÉORÈME III.3.3.6
Soit τ un ensemble de tâches périodiques à criticité mixte ordonnançable. Dès qu'un instant d'inactivité de niveau ℓ se produit dans l'ordonnancement de τ, le niveau de criticité du scénario peut être réinitialisé à sa valeur la plus basse.

PREUVE III.3.3.7
Soit t_{idle} le premier instant d'inactivité de niveau ℓ dans l'ordonnancement de τ (avec $t_{idle} > 0$). Cela signifie que tous les travaux instanciés par une tâche dont le niveau de criticité est au moins aussi élevé que que la criticité du scénario à l'instant t_{idle} ont terminé leur exécution, et qu'aucun travail n'est disponible. Soit τ_i n'importe quelle tâche de τ qui a été désactivée avant t_{idle}. Nous devons montrer qu'à l'instant t_{idle}, tous les travaux instanciés par τ_i peuvent à nouveau recevoir $c_i(\chi_i)$ unités de temps pour leur exécution dans n'importe scénario de criticité d'au plus χ_i. Soit J_i le premier travail instancié par τ_i à l'instant t_i ($t_i \geq t_{idle}$). On distingue deux cas :

1. *t_i est l'instant critique pour τ_i : il s'en suit que le temps de réponse de J_i sera le plus grand parmi tous les travaux instanciés par τ_i. Mais comme τ est ordonnançable au sens de la criticité mixte, nous savons que J_i respecte malgré tout son échéance si il s'exécute*

jusqu'à $c_i(\chi_i)$ unité de temps et que aucune des autres tâches τ_j n'instancie de travail qui s'exécute pour plus de $c_j(\chi_i)$ unités de temps.

2. *t_i n'est pas un instant critique pour τ_i : dans ce cas, nous savons que le temps de réponse de J_i sera inférieur ou égal au coût d'exécution pire cas de la tâche τ_i. Mais nous avons prouvé que J_i peut respecter son échéance si il avait été instancié à un instant critique pour la tâche. Il en suit que J_i respectera également son échéance dans un instant non-critique.*

EXEMPLE III.3.3.5

Considérons une nouvelle fois l'ensemble de tâches τ présenté dans l'Exemple III.3.3.1, et un scénario possible pour cet ensemble de tâche illustré dans la Figure III.3.9. Le premier travail instancié par la tâche τ_3 termine son exécution après 7 unités de temps. Le niveau de criticité du scénario a alors atteint le niveau 2 à l'instant $t = 1$, et le niveau 3 à l'instant $t = 3$. En conséquence, à la fois τ_1 et τ_2 ont été désactivées. A l'instant $t = 7$, un instant d'inactivité de niveau 3 se présente, donc le niveau de criticité peut alors être réinitialisé à 1, et les travaux instanciés par τ_1 et τ_2 peuvent être à nouveau considérés. La tâche τ_2 instancie son travail suivant à l'instant $t = 7$, mais il est préempté par la tâche τ_3 à l'instant $t = 8$. Etant donné que τ_3 a terminé son exécution, après 3 unités de temps, le niveau de criticité du scénario atteint la valeur 2, et la tâche τ_1 doit être désactivée. Cependant, τ_2 est encore apte à respecter son échéance. A l'instant $t = 13$, un instant d'inactivité de niveau 2 se présente, et le niveau de criticité est réinitialisé à nouveau. Cela permet à toutes les tâches d'instancier un travail de nouveau apte à respecter son échéance.

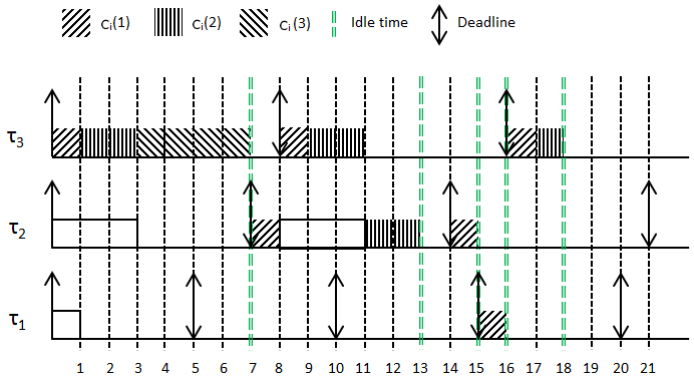

FIGURE III.3.9 – Diminution de criticité

III.3.3.8 Simulations

Dans cette Section, nous comparons les performances de trois solutions possible pour traiter l'accroissement de criticité lors de l'ordonnancement d'un ensemble de tâches à criticité mixte :

1. L'approche traditionnelle (TA) : dès qu'une tâche τ_i dépasse son coût d'exécution pire cas de niveau de criticité ℓ, la criticité du scénario est augmentée, et toutes les tâches τ_j avec $\chi_j = \ell$ ayant une priorité inférieure à τ_i sont désactivées.
2. L'approche traditionnelle, étendue avec un mécanisme de diminution de criticité (CD, pour *Criticality Decrease*) :dès qu'une tâche τ_i dépasse son coût d'exécution pire cas de niveau de criticité ℓ, la criticité du scénario est augmentée et toutes les tâches τ_j avec $\chi_j = \ell$ ayant une priorité inférieure à τ_i sont désactivées. Néanmoins, dès qu'un instant d'inactivité de niveau ℓ se présente, la criticité du système est réinitialisée à sa valeur la plus basse.
3. L'approche traditionnelle, étendue avec un mécanisme de diminution de criticité et avec le concept de marge de tolérance (CD-A) : dès qu'une tâche τ_i dépasse son coût d'exécution pire cas de niveau de criticité ℓ plus sa marge de tolérance, la criticité du scénario est augmentée et toutes les tâches τ_j avec $\chi_j = \ell$ ayant une priorité inférieure à τ_i sont désactivées. Néanmoins, dès qu'un instant d'inactivité de niveau ℓ se présente, la criticité du scénario est réinitialisée à sa valeur la plus basse.

A chaque travail J_k^i instancié par une tâche τ_i est assignée une durée $c_{i,k}^e$ dans l'intervalle $[1, c_i(\chi_i)]$, en utilisant une distribution triangulaire. Sachant qu'un travail ne termine son exécution ni après un très court instant ni après une très longue durée mais plutôt après une durée intermédiaire, l'usage d'une distribution triangulaire est un moyen de représenter cette probabilité. Ainsi, ce qui suit permet de concentrer la plupart des probabilités autour d'une valeur appelée le *mode*. Comme le montre la Figure III.3.10, pour une tâche τ_i, la limite basse de la distribution est mise à 0, et la limite haute à $c_i(\chi_i)$, et le mode de la distribution est égal à $\frac{c_i(\chi_i)}{3}$. Cela signifie qu'un travail va la plupart du temps terminer son exécution après une consommation proche de 33% de son coût d'exécution pire cas de son niveau de criticité propre. Nous générons alors un nombre aléatoire r qui suit cette distribution particulière, et $c_{i,k}^e = \lceil r \rceil$.

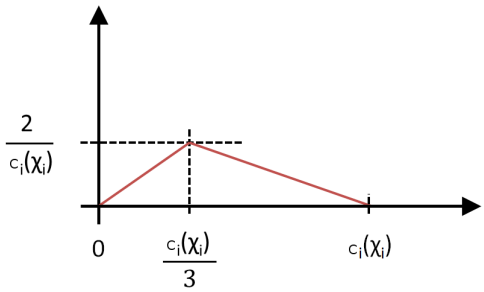

FIGURE III.3.10 – Distribution triangulaire pour la tâche τ_i

La durée exacte de chaque travail est complètement indépendante de la durée exacte des travaux précédents instanciés par la même tâche. Un travail J_k^i complètera son exécution dès que l'ordonnanceur lui aura fourni un total de $c_{i,k}^e$ unités de temps, mais cette durée n'est pas

connue à l'avance par l'ordonnanceur.

Nous analysons les ensembles de tâches τ composés d'un nombre de tâches à criticité mixte $|\tau|$ variant entre 4 et 8, avec une criticité bornée par 4. Pour l'approche CD-A, le nombre de tâches en faute est mis à $k = |\tau|$, pour s'assurer que la criticité est augmentée du fait d'une tâche dépassant son coût d'exécution pire cas à un niveau de criticité donné plus sa marge de tolérance, et non parce que le nombre de tâches en faute dépasse k.

Nous avons comparé le nombre moyen de travaux qui ont dû être supprimés par les trois méthodes. Nous avons alors généré 100 ensembles de tâches à criticité mixte et réitéré les simulations pour de multiples bornes supérieures sur chaque charge de tâche (on note cette borne supérieure en tant que $U_{\max}(\tau_i)$). Le Tableau III.3.8 et la Figure III.3.11 présentent les résultats de ces simulations. L'axe des X représente la charge maximum par tâche $U_{\max}(\tau_i)$, tandis que l'axe des Y représente le pourcentage moyen de travaux ayant été supprimés.

$U_{\max}(\tau_i)$	TA	CD	CD-A
0.1	33.4223%	9.3976%	1.4071%
0.2	32.9436%	11.7511%	4.2982%
0.3	34.0782%	13.3883%	6.2279%
0.4	38.2408%	15.6153%	9.4358%
0.5	42.7098%	19.4652%	13.052%
0.6	42.3251%	20.0583%	14.2854%
0.7	42.9011%	21.8738%	16.7087%
0.8	48.0304%	23.05%	17.1582%
0.9	50.266%	24.0808%	19.1672%
1.0	50.1776%	26.1121%	21.1875%

TABLE III.3.8 – Pourcentage moyen de travail supprimés en fonction de $U_{\max}(\tau_i)$

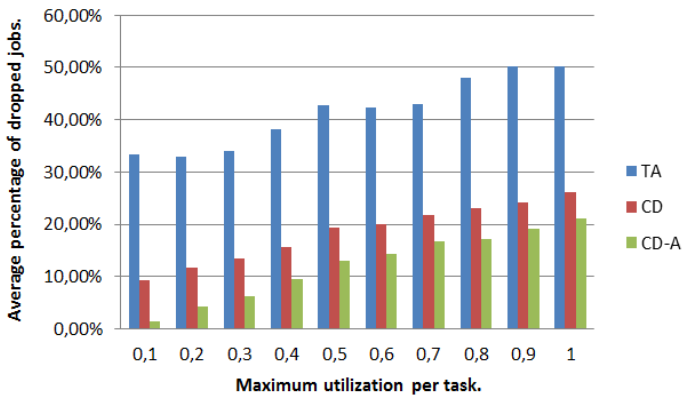

FIGURE III.3.11 – Pourcentage moyen de travaux supprimés en fonction de la charge

Nous observons que le mécanisme de diminution de criticité peut réduire le nombre de travaux supprimés de 25% en pire cas, alors que cette réduction peut atteindre plus de 30%

lorsque la charge de chaque tâche est faible. Nous constatons de plus que ce mécanisme est accrût avec la marge de tolérance, le nombre de travaux supprimés étant réduit de 5% à 9% de plus.

Nous observons que plus la charge de chaque tâche est faible, plus le mécanisme de marge de tolérance est efficace. Cela est du au fait qu'une marge de tolérance plus grande peut être apportée à chaque tâche.

III.3.3.9 Conclusion

Le principe de criticité mixte peut donc être intégré dans la passerelle afin de pouvoir traiter au mieux l'ensemble des tâches nécessaires pour répondre au besoin systronique sur un environnement matériel limité, formalisé par l'Exigence REQ_TEMPS_002.
Les différents ensembles de tâches à criticité mixte, intégrés chacun dans un compartiment spatial et temporel autonome pour des raisons de sécurité, peuvent être ordonnancés de manière efficace en intégrant à la fois le concept de marge de tolérance et le concept de réinitialisation de la criticité. Il reste cependant une limitation en terme d'affinité de l'ensemble de tâche, limité pour le moment à un ordonnancement mono-processeur. Dans le cadre de ma thèse, je considère que de tels ensembles de tâches sont seuls sur le cœur du processeur sur lequel ils sont assignés. Néanmoins, il est possible d'étendre les résultats précédents dans le cadre d'un ordonnancement partitionné de tâches à criticité mixte sur une architecture multi-cœurs.

Chapitre III.4

Vers une passerelle sécurisée et temps réel pour le traitement de flux multi-critiques : cas d'usage véhiculaire

Sommaire

III.4.1 Rappel du besoin vétronique **131**

III.4.1.1 Évolution des systèmes véhiculaires 131

III.4.1.2 Problématiques techniques des architectures vétroniques civiles et militaires 133

III.4.2 Définition d'une architecture logicielle pour une passerelle systronique . . **136**

III.4.2.1 Définition d'une architecture de filtrage multi-fonctions pour les systèmes systroniques 136

III.4.3 Limitations de la solution technique et compléments **147**

III.4.1 Rappel du besoin vétronique

III.4.1.1 Évolution des systèmes véhiculaires

L'intégration des systèmes d'aide à la conduite et de gestion du multimédia a beaucoup évolué ces dernières années. De tels systèmes intègre à la fois des mécaniques de protection de l'usager tels que l'*Anti-Lock Breaking System* (ABS) ou l'*Electronic Stability Control* (ESP). De tels sous-systèmes fonctionnent de manière autonome et n'autorisent aucune interaction utilisateur dans le cadre de leur fonctionnement. Ils restent cependant activables ou desactivables par le conducteur via une interface de contrôle. Ses sous-systèmes sont compatibles avec des exigences temps réel fortes, dûes à la problèmatique de sûreté de fonctionnement qui leur est associée.

Avec l'accroissement des performances des architectures embarquées basse consommation, de

nouveaux sous-systèmes voient le jour dans l'infrastructure véhiculaire. C'est par exemple le cas des radars de reculs, parfois couplés à des caméras, ou encore de nouveaux modes d'interaction avec les terminaux utilisateurs, via du réseau sans-fil courte portée. C'est également le cas de sous-systèmes comme le *stop and go*, permettant de couper le moteur lors de l'arrêt du véhicule afin de réduire la consommation moyenne d'essence.

Les systèmes vétroniques sont aujourd'hui très complexes, intégrant à la fois des éléments avec de fortes exigences temps réel comme l'ABS ou l'ESP, avec des exigences de sûreté élevées comme le stop-and-go ou encore des exigences de sécurité également élevées, comme la gestion du démarrage sans contact. A cela, il faut intégrer un réseau de contrôle et de diagnostic, permettant de simplifier l'audit d'un véhicule en cas de problème électronique.

Dans [Kes09], l'auteur montre à quel point les réseaux d'interconnexion de systèmes vétroniques peuvent être complexes. Ces réseaux sont habituellement basés sur une infrastructure de bus CAN (*Controller Area Network*) interconnectés au travers de gateways. Ces dernières sont parfois à sens unique, afin d'assurer la sûreté de fonctionnement de certains sous-systèmes.
Avec l'intégration de terminaux utilisateurs dans le cadre de la gestion du multimédia, viennent se greffer ces réseaux IP dont le cloisonnement est aujourd'hui de plus en plus complexe à assurer. En effet, la complexité de l'ordinateur de bord est de plus en plus grande, car ce dernier fournit de plus en splus de service. Dans les études en cours, les constructeurs cherchent également à interconnecter le véhicule avec des réseaux distants, et l'utilisateur pourras accéder, via ces derniers, à diverses informations moteurs (e.g. usure des pièces, état général du véhicule, etc.), impliquant un média de remontée d'information. Le niveau de sûreté mais aussi de sécurité des passerelles d'interconnexion devient alors de plus en plus élevé.
L'ordinateur de bord, élément dont l'interface utilisateur est unifié, permet à la fois le paramétrage des élément non-sûrs (multimédia, gestion des périphériques utilisateurs) et des éléments critiques ((des)activation des différents sous-systèmes d'aide à la conduite, etc). Une telle architecture implique un support à la fois des problématiques de sûreté de fonctionnement mais aussi de sécurité, afin de protéger les sous-systèmes critiques de toute attaque ou impact dû à un comportement invalide d'un sous-système non-sûr comme les terminaux utilisateurs.

Comme le décrit l'auteur dans [RO08] la gestion des flux en provenances des différents sous-systèmes se fait au travers d'un ordonnancement de type TDM, assurant un cloisonnement temporel strict entre les traitements à fortes exigences de sûreté et les traitements non sûrs, comme le multimédia. Les réseaux sont également strictement séparés afin d'assurer un cloisonnement spatial strict.

III.4.1.2 Problématiques techniques des architectures vétroniques civiles et militaires

Lorsque systèmes critiques et non-sûrs ne sont plus indépendants et que la notion de réseau IP apparaît dans le cadre de l'interconnexion de systèmes non-sûrs, on parle alors de systèmes *systroniques*. C'est le cas principalement dans les véhicules militaires avec l'intégration des sous-systèmes de gestion des caméras, de la radio véhiculaire courte, moyenne et longue portée mais également au travers des terminaux de gestion intégrés nécessitant un cloisonnement sécuritaire fort. En effet, un véhicule de ce type émet et reçoit des données dont la confidentialité peut être restreinte au pays ou à la coalition. Une telle problématique apparaît également dans les véhicules de police ou encore dans les véhicule médicalisés pouvant interagir avec une base de données patients centralisée, du fait des exigences de confidentialité associées aux données médicales.
Dans le cas des véhicules militaires, des exigences de sûreté de fonctionnement sont également présentes sur la partie caméra, lorsque le véhicule possède un blindage *transparent* (i.e. le conducteur s'appuie sur des caméras et des capteurs plutôt que sur des vitres).

Les exigences de certification, tant en terme de sécurité qu'en terme de sûreté de fonctionnement, exigent que lorsque deux sous-systèmes interagissent directement, ces derniers doivent être compatibles du niveau de certifiabilité de celui ayant la plus forte contrainte.
La conséquence directe est bien entendu le coût associé à la certification. Cela limite en conséquence la complexité de l'ensemble du système et peut interdire certains usages, comme l'intégration de terminaux non-sûrs.

Dans les véhicules militaires, la demande pousse vers l'usage de terminaux fantassins type smartphone/tablette, à l'instar de la poussée du BYOD (Bring Your Own Device) dans les systèmes d'information civils. Bien que maîtrisés par le fournisseur, ces terminaux restent par nature non-sûr, de par la complexité de leur architecture logicielle. Ils peuvent de plus potentiellement être multi-niveaux de sécurité, impliquant une gestion de leur interconnexion avec le *système d'information* véhiculaire complexe.
Cette problématique là apparaît également sur les terminaux GPS, dont la maîtrise par le pays utilisateur n'est pas toujours assuré, la technologie GPS étant une solution militaire américaine.

Dans les systèmes systroniques sont définit trois réseaux d'interconnexion :

- Le réseau moteur, basé sur une architecture complexe de bus CAN, permettant de remonter l'ensemble des informations des différentes sondes, mais aussi de contrôler certains relais et fonctions électroniques ;
- Le réseau critique, ayant de forte exigences de sûreté de fonctionnement, mais pas d'exigence de sécurité lorsqu'il est pris seul. Ce réseau gère les éléments critiques du véhicule

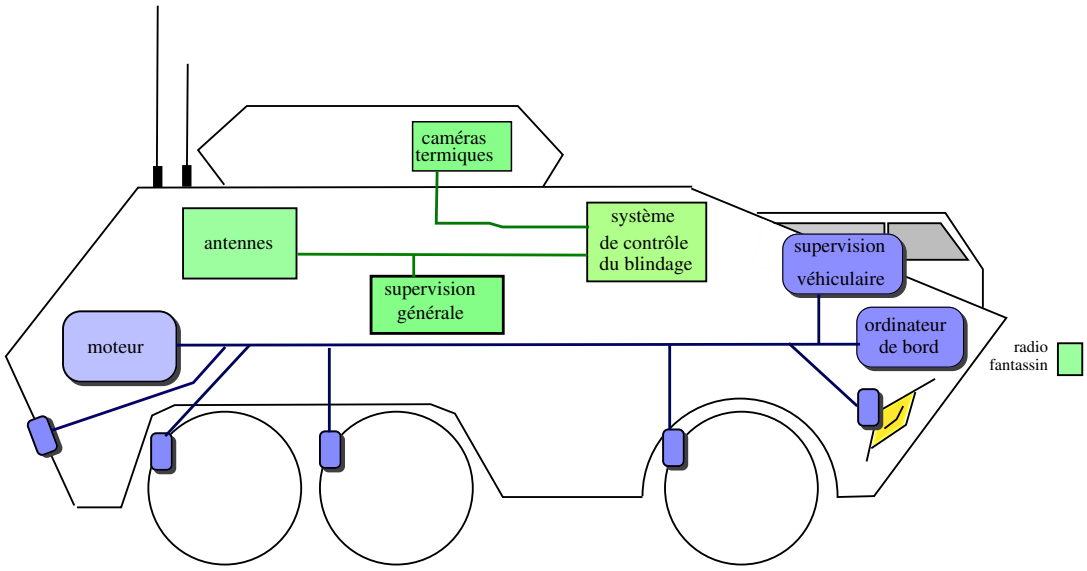

FIGURE III.4.1 – Exemple d'intégration d'un système systronique pour véhicule blindé léger

tels que les caméras ou les systèmes de désactivation des mécanismes défensifs en cas d'ouverture de portes ;

– Le réseau d'interconnexion au système d'information et de commandement, qui gère les antennes, les connexions avec les terminaux fantassin, la gestion de la voix ou encore le GPS

Aujourd'hui, il est demandé dans les systèmes systroniques une solution permettant de faire transiter certaines données entre ces trois réseaux, afin principalement de simplifier la supervision de l'ensemble de ces sous-systèmes. Cela implique d'interconnecter des réseaux aux exigences de sûreté et de sécurité variables. Il devient alors nécessaire de prévoir une passerelle permettant de faire transiter certains flux avec un contrôle strict, à la fois afin de garantir la sûreté et la sécurité du système entier. Pour des raison de poids, de température et de surface, cette passerelle doit être embarcable et de petite taille. Elle doit de plus répondre aux exigences physiques en terme de vibration, de température ou encore de résistance à un air chargé en particules comme le sable.

Le fait d'interconnecter des réseaux de niveaux de sûreté et de sécurité différents implique une architecture de passerelle particulière, permettant de maintenir le niveau de sécurité demandé tout en permettant le transfert de certaines informations. La passerelle doit également permettre d'interagir avec le réseau moteur, tout en assurant un filtrage suffisant pour ne pas impacter la sûreté de fonctionnement associée à ce réseau.

Afin de répondre aux problématiques de sécurité, j'ai proposé plusieurs moniteurs de sécurité dans le Chapitre III.2. L'association de ces différents moniteurs permet d'assurer le cloisonnement des données sensibles. Leur usage sera décrit plus loin dans ce chapitre. Il faut également garantir le maintient du niveau de sûreté des réseaux moteur et caméras. Pour cela, je

propose de m'appuyer sur le moniteur de sécurité diode logicielle, permettant d'assurer une remontée d'information à sens unique entre sonde moteur et console de supervision d'une part et caméras et console conducteur d'autre part. Pour ces deux flux, il est de plus important d'assurer une latence maximum garantie. C'est une exigence particulièrement critique pour la remontée de flux d'information des caméras lorsque le véhicule s'appuie sur un blindage transparent.

Dans le cadre du réseau d'interconnexion du système d'information, le transport de la voix est un élément important. Non seulement il permet aux chef de patrouille de communiquer via la radio avec le système d'information et de commandement, mais il permet également au personnel de communiquer entre eux, le bruit ambiant étant trop élevé pour permettre une discussion directe. Cette problématique induit un usage important de l'infrastructure VoIP de bord, avec une conséquence en terme de multi-criticité. En effet, lorsque la passerelle de traitement de la voix ne permet pas un traitement concurrent de l'ensemble des flux voix, la priorité passe aux plus haut gradé, impliquant une potentielle coupure du flux d'un (ou plusieurs) des subordonnés.

L'interconnexion doit être garantie en terme de bande passante afin de pouvoir répondre aux besoins du domaine de sûreté le plus faible. C'est un besoin important dans le traitement de flux vidéo ou voix émis à partir d'un domaine de sûreté élevé. Dans ces deux cas, un débit et une latence garantis sont nécessaires, comme définit dans l'exigence REQ_TEMPS_001. C'est également le cas de certains flux d'alertes, comme la remonté d'alarme des différentes sondes moteur. Cela implique également que les réseaux interconnectés soient aptes à assurer des délais. Ainsi, sur un réseau systronique de type Ethernet, on s'appuiera sur du réseau TT-Ethernet (Time Triggered Ethernet) [Ste+09] synchronisé temporellement via un protocole type IEEE 1588 [Ste+09].
Néanmoins, les données transmises peuvent avoir à être filtrées afin d'éviter le transfert non maîtrisé de données confidentielles. Parce que les système de sûreté plus faible ne peuvent être certifiés pour le traitement de flux dont le niveau de sûreté est supérieur, ces derniers ne peuvent émettre de requêtes vers un domaine de sûreté supérieur sans validation préalable par un agent de confiance. Il peut parfois être exigé que la communication soit garantie à sens unique entre deux domaines de sûreté hétérogènes. Dans ce dernier cas, un moniteur de sécurité de type diode logicielle doit permettre de garantir que le flux d'interconnexion est à sens unique. Cela peut être le cas pour les sondes moteurs, qui ne devrait pas recevoir de requête de la part d'un élément de domaine de sûreté plus faible, mais plutôt émettre une information de manière périodique, simplifiant dans le même temps leur comportement.

Les systèmes systroniques sont basés sur une interconnexion complexe de systèmes de niveau de sûreté et de sécurité hétérogènes avec un grand nombre d'interactions.
Ces interactions nécessite d'être filtrées, validées voire inspectée afin de garantir à la fois le besoin sécuritaire et le besoin de sûreté des différents sous-systèmes présents. Afin de permettre

ces différents traitement, un équipement d'interconnexion doit être mis en place aux différents points de communications inter-domaines, et doit être compatible des exigences de sûreté et de sécurité des différents domaines qu'il interconnecte.
Dépendant des réseaux qu'il interconnecte, ce dernier doit pouvoir fournir différents services à la fois en terme de sécurité et de temps réel. Ainsi, la remonté d'informations moteur sur un smartphone ne nécessite pas de conformité à des exigences temps réel. A l'inverse, la remonté de flux vidéo à partir d'un système de type blindage transparent doit garantir un comportement temps réel dur afin d'assurer une bonne vision de l'environnement immédiat du véhicule.
En terme de sûreté, l'interconnexion d'une console de gestion du système d'arme doit pouvoir accéder aux informations de status de ces dernières, mais ne doit pas pouvoir émettre d'ordre de mise en route. En terme de sécurité, les données de positionnements auxquelles accède le véhicule ne doivent pas être réémise sur l'ensemble des systèmes radios qui sont interconnectés, mais uniquement à ceux qui sont utilisés dans le cadre de la coalition voire exclusivement du pays utilisateur.
On constate ainsi que les systèmes systroniques, décrit ici dans le cas militaires, impliquent à la fois l'interconnexion de systèmes de sûreté et de sécurité hétérogènes, avec des exigences variables et des données multiples. La définition d'un équipement d'interconnexion modulaire est donc important pour limiter le coût de la solution complète. Cette problématique se retrouve dans différents secteurs, comme le médical ou la police.

III.4.2 Définition d'une architecture logicielle pour une passerelle systronique

III.4.2.1 Définition d'une architecture de filtrage multi-fonctions pour les systèmes systroniques

La conséquence de la problématique décrite en III.4.1.2 est que plusieurs type de connecteurs sont nécessaires, à divers endroits du systèmes systronique, avec des contraintes différentes, tant en terme de service rendu qu'en terme de sécurité et de temps réel.
Définir une solution modulaire, apte à intégrer en entrée une configuration extérieure lors de la production serait un atout en terme de coût dans le cadre de la définition du système.

Comme décrit dans le Chapitre III.2, l'architecture logicielle du connecteur est basée sur un ensemble de conteneurs autonomes communicants. Chaque conteneur est dédié à un traitement unitaire dont je limite la complexité. Ces traitements sont par exemple un filtre DPI, un mécanisme de shaping de flux, une diode logiciel assurant un transfert sens unique ou tout autre traitement nécessaire au service que doit rendre le connecteur. Ces conteneurs ont pour but d'être réutilisables, et ce indépendamment des autres conteneurs déployés, afin de limiter le

coût de réimplémentation des fonctions et de re-certification de ces dernières.
Certains traitements peuvent correspondre à des besoins sécuritaire ou de sûreté de fonctionnement, comme la diode logicielle. Dans ce cas, le conteneur doit pouvoir être certifié afin de garantir la fonction. Dans ce cas, on souhaite considérer cette fonction comme un moniteur de sécurité (ou de sûreté), comme je l'ai décrit dans le Chapitre III.2. Dans d'autre cas, comme par exemple pour le filtre DPI, il peut être nécessaire d'intégrer un ensemble de fonctions comme une pile réseau, impliquant un volume de code plus grand et accroissant fortement le coût de certification associée. L'intérêt des conteneurs est donc de permettre une dissociation de la certification des différentes fonctions, si ces dernières sont garanties disjointes en terme logiciel. C'est par exemple le cas lorsque l'on s'appuie pour la séparation spatiale et temporelle sur des compartiments PikeOS II.2.1.5.
Dans le cadre du besoin systronique, j'ai proposé un premier maquettage de plusieurs de ces fonctions :

- Une fonction diode, assurant le transit en sens unique des données.
- Une fonction DPI, assurant un filtrage des données en fonction d'une politique de filtrage déterminée en avance.
- Un émetteur périodique, assurant un profil de flux en sortie garanti, tout en empêchant l'usage du profil d'entrée pour faire transiter de l'information via un canal caché.

Ces différents conteneurs peuvent correspondre à une fonction de faible empreinte, qui peut être difficile à superviser. En conséquence, ces derniers doivent tous fournir une interface de supervision, pour interagir avec un conteneur de supervision. Ce dernier a pour but de récolter les différentes informations de status voire de débogue associées à ces différents conteneurs, pour les renvoyer vers un environnement de supervision déterminé.

Dans le cadre du besoin systronique, il est souvent demandé que le connecteur soit capable de fournir une garantie de coût de traversée maximum (WCTT - Worst Case Traversal Time) pour les flux qu'il fait transiter. Comme décrit dans la Figure III.4.2, j'ai implémenté un mécanisme de mesure du coût de traversée, qui débute lorsqu'un paquet est reçu par le conteneur *VE filter* et se termine lorsqu'un paquet retransmettant cette même donnée est émis par le conteneur *VE sender*.
Le respect du WCTT implique que chaque tâche en charge à un instant donné du traitement de la donnée à faire transiter doit être considérée comme une tâche temps réel, impactant en conséquence la politique d'ordonnancement en charge de cette dernière. Dans le cadre du maquettage que j'ai effectué, je me suis basé sur la technologie des Cgroups Linux, associée aux politique d'ordonnancement temps réel POSIX de type SCHED_FIFO. L'usage des Cgroups dans la maquette impacte l'architecture générale. Ainsi, dans cette dernière, il ne s'agit pas d'un ordonnancement hiérarchique mais d'un ordonnancement unique par l'ordonnanceur SCHED_FIFO du noyau Linux.

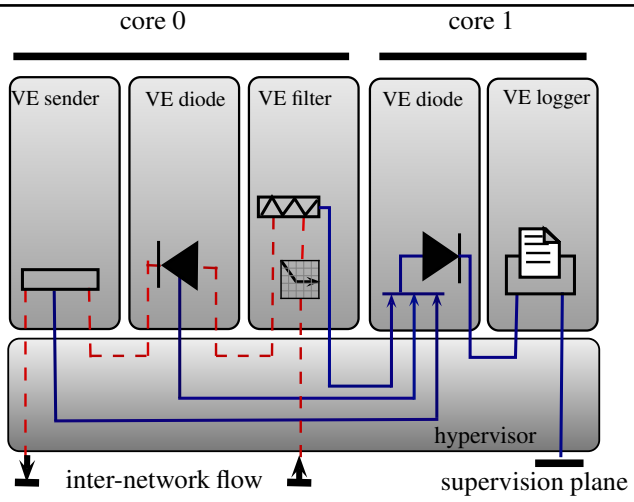

FIGURE III.4.2 – Architecture logicielle type d'une passerelle filtrante

Si la maquette avait été faite sur base d'un ordonnancement hiérarchique, comme par exemple via l'usage d'un micro-noyau de virtualisation type PikeOS II.2.1.5, il aurait alors été nécessaire de déterminer correctement à la fois l'ordonnancement des différents conteneurs et celles des tâches dans ces derniers. Pour répondre à cette problématique, j'ai défini dans la Table III.3.2 cinq équations permettant de déterminer si le système est alors apte à garantir l'ordonnançabilité de l'ensemble des fonctions temps réel, dans le cadre d'un ordonnancement de type EDF sur un partitionnement global de type TDM.

A propos du conteneur de supervision

Comme vu précédemment, le connecteur a souvent une fonction de filtrage, soit pour valider le contenu des flux à transiter, soit pour assurer un transit à sens unique. Dans ces différents cas, la fonction de filtrage doit pouvoir remonter des alertes en cas de non-respect de la politique de filtrage ou en cas d'incapacité à exécuter correctement son traitement (e.g. dans le cas d'un débit trop important). Ces remontées d'alertes doivent être prises en compte et transmises à un équipement dédié à la supervision. C'est ce qui est décrit dans la Figure III.4.2.

A propos du conteneur de filtrage DPI

Il peut être nécessaire dans les systèmes systroniques que les flux transitant d'un domaine à un autre soient filtrés afin d'assurer la confidentialité de certaines données pour ne faire transiter que celles dont le niveau de déclassification est suffisant pour être transmis dans un domaine de moindre niveau. C'est par exemple le cas des données de cartographie ou de positionnement vis à vis d'un système de communication de type coalition.
Afin d'assurer un tel filtrage, une fonction DPI doit être implémentée et prendre en donnée d'entrée une politique de filtrage qui doit être considérée pour valider chaque flux en cours de

transit, afin de valider ou interdire une remontée d'information vers le domaine de plus faible sécurité. Ce conteneur est intégré dans la Figure III.4.2 sous la dénomination *VE filter*. Son traitement consiste en une vérification de la conformité du flux d'entrée envers la politique de sécurité avant validation et réémission. La validation du flux d'entrée sur la donnée et non sur les en-têtes protocolaires de couche quatre du modèle OSI et inférieur. Une fois validée, la donnée est réémise après reforgeage de l'en-tête réseau. La *VE filter* étant considérée de confiance, les en-têtes forgées sont également considérée comme telles. Un tel mécanisme est appellée séparation protocolaire et permet d'éviter tout usage des en-têtes réseau comme IP ou TCP pour faire transiter de l'information.

Les traitements de la VE filter sont décomposés comme suit :

- Un récepteur de paquet, qui reçoit la donnée et valide son contenu auprès d'une politique de sécurité. Le WCET de cette tâche peut être très variant selon la complexité de la politique de sécurité et du contenu du paquet. Il est donc nécessaire de le mesurer soit via une étude statistique sur son coût d'exécution en générant un flux réseau représentatif, permettant d'avoir une valeur approchante mais non garantie, soit en s'appuyant sur une analyse statique du code.
- Un émetteur, récupérant les paquets tagués par le récepteur comme valide et qui les re-expédie vers le compartiment suivant. Le WCET de cette tâche est assez stable, son coût n'étant dépendant que de la taille de trame.

Dans le cas d'un véritable système DPI avec une politique de sécurité complexe, l'implémentation du filtre serait beaucoup plus complexe. Il impliquerait entre autres un mécanisme permettant de concaténer les messages afin de reconstituer l'information de niveau application avant de pouvoir déterminer sa compatibilité avec la politique de sécurité. Il impliquerait de plus un mécanisme de gestion de queues de messages multiples couplées à une politique de qualité de service assurant la priorisation de certains flux par rapport à d'autres. Dans le cadre de la maquette, la politique DPI en elle-même n'est pas l'élément dont je souhaite démontrer la faisabilité. Cette dernière est donc simple et s'appuie sur un protocole TLV qui assure que l'information de niveau applicatif est entièrement présente dans une et une seule trame réseau. Il existe plusieurs type de contenu TLV. Bien que les trames soient parfois de taille identique, le système DPI cherche à étudier l'information qui y est stockée, afin de déterminer une action en conséquence. Le filtre est donc en charge d'assurer que le contenu d'une trame, dans le cadre d'un séquencement de trame valide (e.g. type 1, type 2, type 2, type 5) est bien compatible avec la politique de sécurité.

A propos du conteneur Diode

Il arrive dans certains cas que la communication entre deux conteneurs soit garantie comme étant en sens unique. C'est le cas lorsque l'on change de domaine de sécurité (lors de la remontée d'informations entre un domaine de sécurité et un autre de niveau supérieur) mais aussi lorsque l'on change de domaine de sûreté (afin d'assurer que le domaine de moindre sûreté ne

vient pas impacter le domaine de plus fort niveau de sûreté).
Ainsi, dans les systèmes systroniques, les remontées d'informations coalitions vers le domaine pays doivent dans le même temps assurer que les données pays ne sont pas renvoyées vers le domaine coalition. L'usage d'un élément certifiable en coupure permet d'assurer cette fonction. C'est le but du conteneur Diode.
Le but de ce conteneur est d'assurer un transfert sens unique entre deux domaines. Pour des raisons de sécurité liées aux problématiques de canaux cachés[Lam73], ce conteneur doit également assurer que le comportement du récepteur de domaine de sécurité plus élevé ne soit jamais connu de l'émetteur, dont le domaine de sécurité est plus faible. Ainsi, l'émetteur ne peut retirer aucune information du débit de son émission, ce dernier n'étant par corrélé au comportement du récepteur.
Une telle architecture a pour conséquence que le conteneur diode peut perdre des paquets si le récepteur n'est pas apte à les recevoir. En effet, faute d'en informer l'émetteur, le conteneur diode doit alors détruire les paquets et lever une alerte auprès d'un environnement de supervision de niveau de sécurité apte à traiter cette information. Le conteneur de supervision est visible dans la Figure III.4.2, et a entre autre pour but de récupérer les alertes du conteneur Diode.

A propos du conteneur émetteur périodique

L'émetteur périodique (nommée VE Sender) exécute une tache périodique pour l'émission des paquets en sortie de passerelle. Le but de ce conteneur est d'assurer un débit (en nombre de trames) fixe, avec une période inter-trames également garantie. Ce conteneur permet de répondre à plusieurs besoins :

1. En cas d'émission vers un domaine de sûreté élevé, ce conteneur empêche tout flooding d'interface, en limitant le débit à une valeur maîtrisée. Ce comportement permet à la passerelle d'être compatible de l'Exigence REQ_TEMPS_002.
2. Afin d'éviter tout canal caché basé sur la période inter-trame, cette dernière est remaniée à une valeur fixe. Ainsi, l'utilisation d'une variation de période inter-trame par l'émetteur afin d'envoyer de la donnée en morse est bloquée. En cas de non émission de trame, l'émetteur renvoie alors la dernière trame, jusqu'à arrivée d'une nouvelle trame à émettre. Ce comportement permet d'être compatible avec l'Exigence REQ_SECUR_005.

La période d'émission de l'émetteur périodique définit le débit maximum supporté par la passerelle.

Afin d'assurer un comportement temps réel de la passerelle, cette dernière doit pouvoir assurer un coût de traversée pire cas (WCTT - Worst Case Traversal Time). La tâche d'émission périodique de paquet s'exécute donc selon un schéma périodique strict. En fonction du profil de flux entrant, il est nécessaire de définir un buffer d'entrée permettant de supporter des bursts.

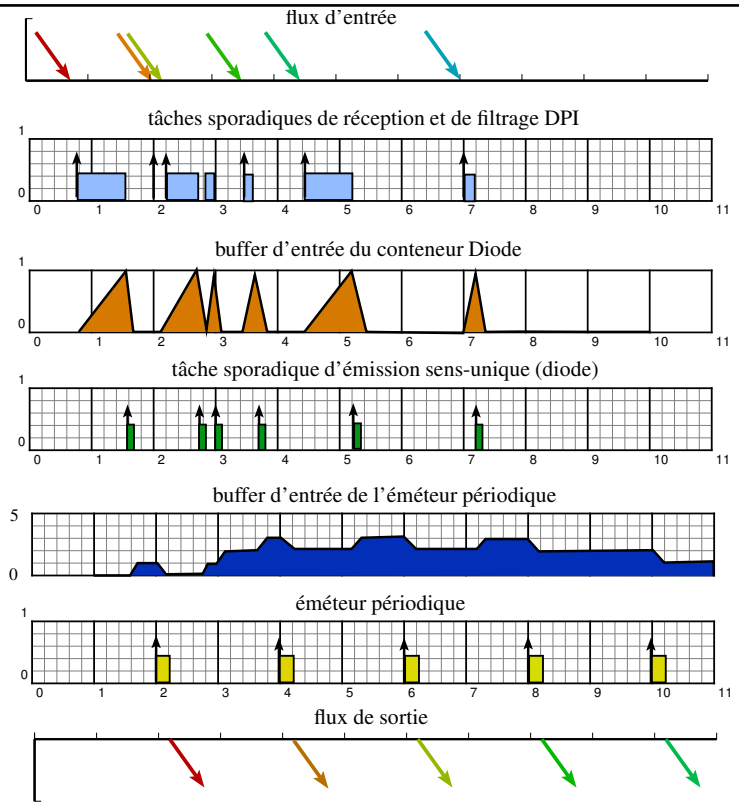

FIGURE III.4.3 – Contrôle du débit à l'émission de la passerelle par l'émetteur périodique

Ce buffer est déterminé en nombre de paquets, afin de garantir une profondeur dépendante du nombre de paquets exclusivement, sans être impacté par la taille de ces derniers. Cela permet ainsi de mieux maîtriser le débit entrant en paquet, sans considérer leur variation de taille. Un telle architecture de buffer d'entrée s'appuie sur le principe de Queuing Ports de l'ARINC 653 comme décrit dans [Ale+07] et [ARI10]
La Figure III.4.3 montre un exemple de mécanisme de forwarding d'une passerelle générique.

Mesure empirique du coût de traversée pire cas par maquettage d'une passerelle

Architecture et configuration

La configuration de la maquette et le flux d'entrée sont décrits dans la Table III.4.1.

Le maquettage initial a été construit sur l'architecture logicielle de conteneurisation LXC (Linux Containers). Cette architecture s'appuie sur un seul noyau et un seul ordonnanceur en charge de gérer les tâches des différents conteneurs. Dans ce cadre, l'ordonnancement n'est pas de type hiérarchique, du fait de l'unicité de l'ordonnanceur. Néanmoins, ce dernier est apte à différentier les différents conteneurs et donc de gérer une politique d'ordonnancement variable

Item	**Description**
Architecture	i686 core 2 duo 2Ghz
NICs	DLINK-RTL8139, VIA VT6105
OS	Debian Squeeze
Noyau	linux 3.2.12-rt24
	pas de support ACPI
	mode FULL_PREEMPT
Type de compartimentation	LXC
Sécurité	protection noyau (mémoire et durcissement compartimentation LXC)
Flux d'entrée	**Description**
type de flux	UDP sur IPv4
type de payload	TLV
	T (*champs Type*) taille de champs de 1 octet
	L (*champs Taille*) taille de champs de 1 octet
nombre de types	5 types
taille de payload	type 0 : 50 octets
	type 1 : 50 octets
	type 2 : 30 octets
	type 3 : 30 octets
	type 4 : 70 octets
contenu de la payload	contenu ASCII

TABLE III.4.1 – Description de la configuration utilisée pour la mesure

par conteneur. Dans le cadre du maquettage, la différenciation est limitée à l'affinité CPU, afin de séparer les tâches de supervision et de traitement sur des coeurs disjoints. Ainsi, les sets de tâches des conteneurs DPI, diode et Émetteur périodique sont tous réunis sur un seul cœur, avec une politique d'ordonnancement de type POSIX SCHED_FIFO.

Afin de valider le comportement temps réel souple de Linux, ce dernier a été validé et utilisé avec le patch Linux-RT. Ce patch permet de fortement réduire l'impact du système Linux sur l'ordonnancement des tâches temps réel. En effet ce dernier permet entre autre de :

- Supporter le mode FULL_PREEMPT, rendant le noyau preemptible (à l'exception de certains éléments comme les gestionnaires d'interruption) ;
- Débrayer les priorités des différents threads kernel, afin qu'ils ne puissent être plus prioritaire que les tâches temps réel ;
- Limiter l'exécution des différentes *softirq* Linux au contexte du thread ksoftirqd, afin qu'elles ne puissent s'exécuter dans un contexte d'appel système.

La capacité du noyau Linux à supporter le temps réel souple a été mesuré dans [al09].

La Figure III.4.4 montre que la tâche de priorité la plus élevée (ordonnancement SCHED_FIFO, priorité 99) s'exécute avec une latence au démarrage d'au maximum 300 microsecondes. Cette mesure a été effectué avec l'outillage de mesure temps réel cycletest util-

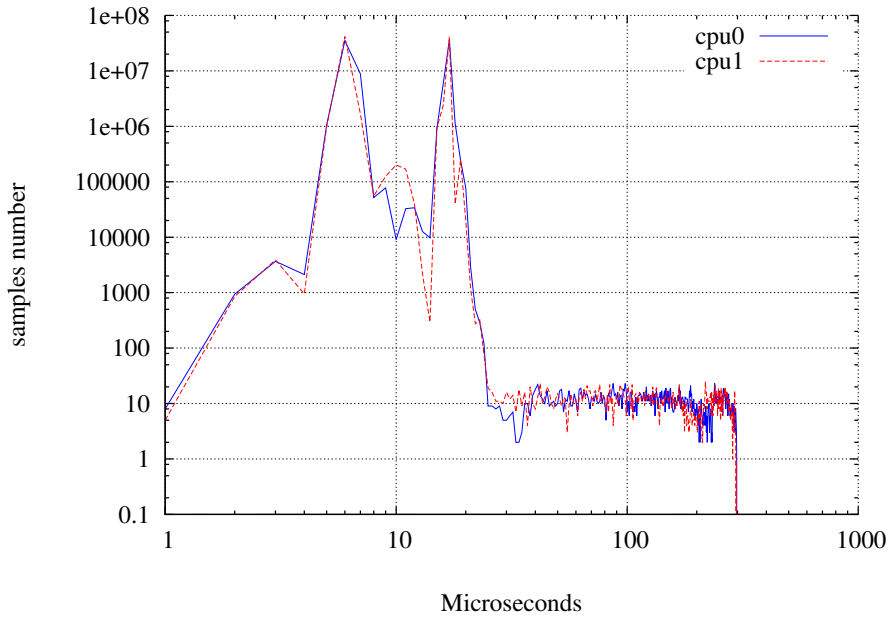

FIGURE III.4.4 – Latence au démarrage de la tâche de priorité la plus élevée

Tâche	Politique d'ordonnancement	profil d'ordonnancement	priorité	WCET (mesuré)	Période
Filter (nommée F)	SCHED_FIFO	sporadic	97	90us	1 ms
Diode (nommée D)	SCHED_FIFO	sporadic	98	60us	1 ms
Sender (nommée S)	SCHED_FIFO	periodic	99 (highest)	80us	10 ms

TABLE III.4.2 – Description des choix d'ordonnancement de tâches pour la maquette

isé dans [lab12], pendant une durée de 5h avec une charge CPU de 40 et un traitement de flux représentatif du besoin, ce qui correspond à un nombre d'échantillons (en terme de jobs) supérieur à 100 millions.
La mesure a été faite pour une tâche s'exécutant dans le conteneur *VE sender*, avec une interconnexion des conteneurs via sockets unix, comme décrit dans la Figure III.4.5.

Dans le cadre de la maquette, le système a été implémenté avec une politique d'ordonnancement de type Rate Monotonic, avec la configuration suivante :

La plus grande priorité est donnée à la tache Sender, puis à la tache Diode et enfin à la tache Filter. Ce choix a été fait afin de réduire la latence de traversée, en priorisant l'émission de données bufferisée par rapport à la réception de nouvelles données.

Filtrage de paquets avancé (DPI)

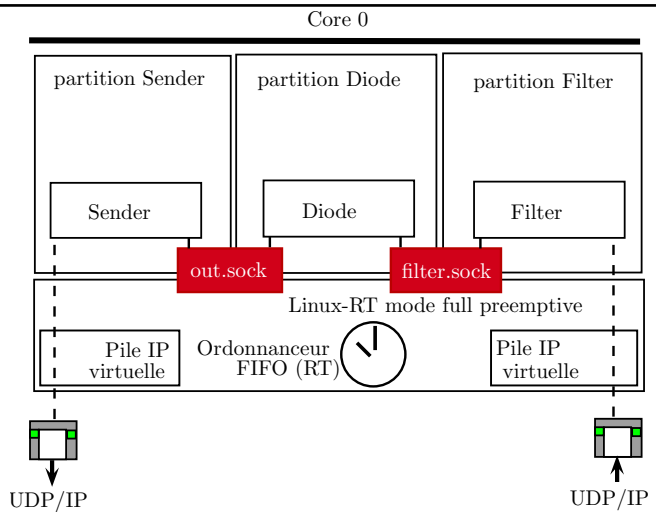

FIGURE III.4.5 – Architecture logicielle générale de la maquette

Afin de déterminer un automate de filtrage de paquets dans la tâche Filter, j'ai écrit un protocole très simple, basé sur une structure TLV (Type/Length/Value). Ce protocole permet 5 types différents, sans récursion. Chaque type possède sa propre taille et son propre contenu. Un émetteur et un récepteur ont été implémentés pour émettre le flux sur du protocole UDP vers la tâche Filter de la passerelle d'une part, et recevoir le flux réencapsulé dans du protocole UDP par la tâche Sender de la passerelle d'autre part. L'automate de la tâche Filter a pour but de valider les différents types, et pour chaque vérifier la taille et la conformité du contenu par rapport à une politique de sécurité chargée en mémoire.
Si le paquet est invalide (non conforme à la politique de sécurité) ce dernier est droppé et un compteur de statistiques est mis à jour. Dans un cas plus complet, ce dernier serait lu par un conteneur de supervision pour remonter une alerte.

Diode logicielle

La Diode logicielle est une tâche sporadique du conteneur *VE Diode*. Cette tâche lit les données transmise par la tâche de Filtrage du conteneur *VE Filter* et les transmet au conteneur *VE Sender*, sans traitement complémentaire. Son coût d'exécution est donc assez stable, seule la taille des paquets reçus et émis étant impactant.

Émetteur périodique

L'émetteur périodique est la tâche du conteneur *VE Sender* en charge d'assurer la maîtrise du flux de sortie, via un comportement prédictible. En effet, cette dernière est périodique et assure l'émission d'un paquet vers l'extérieur à chaque période. Ce dernier peut être un nouveau paquet (si le buffer d'entrée de la tâche a été remplit par la Diode logicielle) ou le même paquet que

précédement (si aucun nouveau paquet n'est arrivé).
L'émetteur périodique ne supportant pas, dans le cadre de ce maquettage, un mode burst, il est le facteur limitant du débit maximum transféré par la passerelle.

Communication entre les conteneurs

Comme vu plus haut, dans le cadre du maquettage la communication entre les conteneur a été faite au travers de sockets unix, via des sockets nommées remappées dans les différents conteneurs. En effet, l'usage d'IPC POSIX standard comme les pipes ou les SHM est rendu impossible par la mise en conteneur. En effet, le but est de se rapprocher d'une architecture logicielle dont les canaux de communications entre conteneurs sont réduits exclusivement à ceux explicitement déclaré, comme je l'ai décrit dans le Chapitre III.2.2.1. De plus, dans le conteneur Diode, la socket unix partagée avec le conteneur Filter est montée dans un système de fichier virtuel en lecture seule, afin d'assurer qu'une donnée ne peut être retransférée vers l'arrière.
L'usage de ces sockets permet de transférer des données sans aucun ajout d'en-têtes protocolaires. Ainsi, les couches 1 à 4 du flux réseau entrant est décapsulée par la tâche de Filtrage DPI, et seule la tâche Émetteur périodique réencapsule les données à l'émission.

Résultats empiriques

Plusieurs mesures ont été effectuée afin de valider l'efficacité de l'architecture logicielle et sa capacité à supporter les exigences temps réel. Afin de démontrer que la passerelle garantie la périodicité du flux émis quel que soit le profil de flux entrant, un flux entrant dont la période inter-trame est aléatoire a été généré afin de valider le comportement de la passerelle. Pendant l'émission de ce flux, le WCET et la période des trois tâches ont été mesurés. Les périodes minimums de la tâche Filter et Diode n'ont pas été définies, ces dernières étant directement dépendantes de la période inter-trame du flux entrant.

La Table III.4.3 décrit le premier jeu de mesures

Un premier jeu de mesure a été fait en utilisant un flux d'entrée généré aléatoireement avec une période inter-trame de 10 millisecondes. Le profil de flux est donné dans la Table III.4.1.
Les Figures III.4.7, III.4.8 et III.4.9 montrent le profil d'exécution des trois tâches de la passerelle dans le cadre du traitement de ce flux. On démontre, au travers de ces mesures, que quel que soit le profil de flux entrant et son impact sur les tâches Filter et Diode, l'Émetteur périodique garantit un flux maîtrisé en sortie. La faible variation de sa période est la conséquence de la variation de la latence d'exécution du noyau Linux, qui a été mesuré dans le cadre de la Figure III.4.4. Ainsi, le flux sortant possède une période inter-trame qui correspond à la période de l'Émetteur périodique.

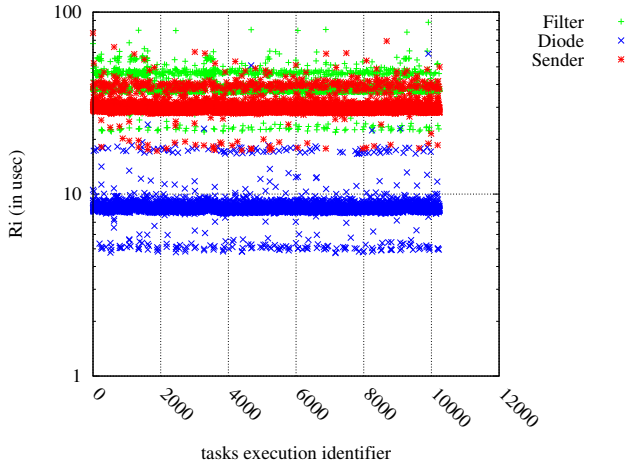

FIGURE III.4.6 – Test 1 : WCET mesurée pour Filtre, Diode et Émetteur périodique

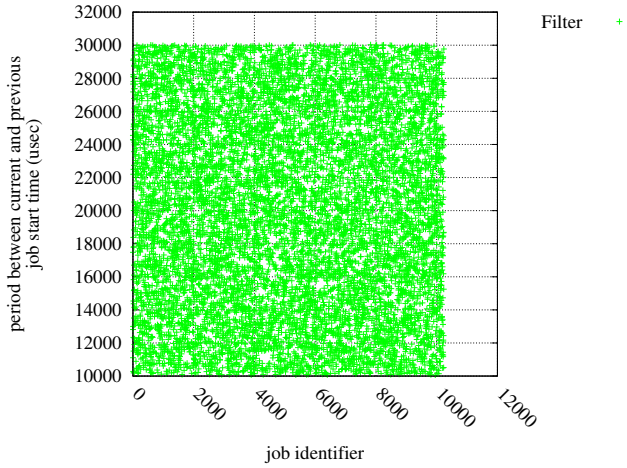

FIGURE III.4.7 – Test 1 : Variation de période pour la tâche Filter

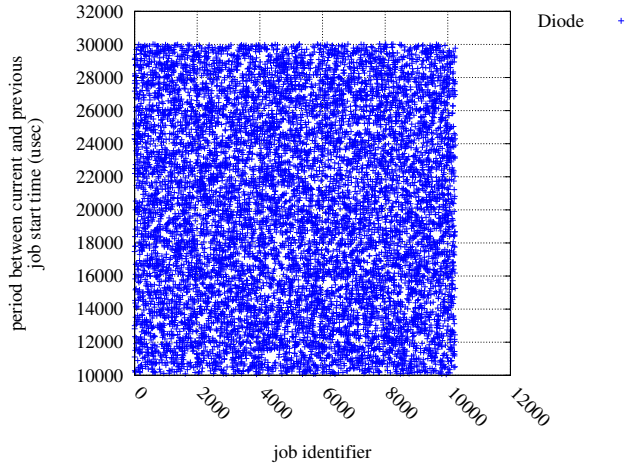

FIGURE III.4.8 – Test 1 : Variation de période pour la tâche Diode

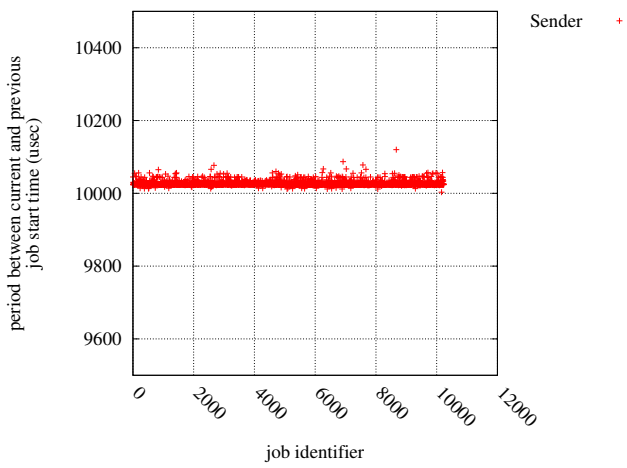

FIGURE III.4.9 – Test 1 : Variation de période pour la tâche Émetteur périodique

Jeu de test 1	
échantillons en entrée	10.000
période minimum du flux d'entrée	10 ms
période maximum du flux d'entrée	30 ms
variation de la période du flux d'entrée	aléatoire
contenu du flux d'entrée	protocole TLV compatible de la politique de sécurité DPI
choix du type TLV du flux d'entrée	aléatoire
débit moyen du flux d'entrée	68.8 Ko/s
nombre de paquets perdu(s)	0

TABLE III.4.3 – Profil du jeu de test 1

Afin de vérifier la stabilité de la passerelle, un second flux a été généré avec une période inter-trame beaucoup plus courte, et supérieur à la capacité d'absorbtion de la passerelle. Ce flux est décrit dans la Table III.4.4.

Jeu de test 2	
échantillon d'entrée	10.000
période minimum du flux d'entrée	1 ms
période maximum du flux d'entrée	1 ms
contenu du flux d'entrée	protocole TLV compatible de la politique de sécurité
choix de la payload du flux d'entrée	aléatoire
débit moyen du flux d'entrée	688 Ko/s
paquets perdu(s)	90%

TABLE III.4.4 – Profil du jeu de test 2

Étant donné que la période minimum des tâches Filter et Diode n'est pas spécifiée, leur période minimum correspond à la plus petite période autorisée par le système d'exploitation, qui dans le cadre de la maquette a été positionnée à 1 milliseconde. Les Figures III.4.10 et III.4.11 le démontre. Ces Figures montrent également des anomalies dans la période du flux d'entrée, accroissant la période de ces deux tâches avec une probabilité moyenne de 0.02% dans le cadre de la mesure. Dans le même temps, l'Émetteur périodique maintient sa période et n'est d'aucune manière impacté par l'ordonnancement des tâches Filter et Diode, comme le montre la Figure III.4.12.

III.4.3 Limitations de la solution technique et compléments

Nous avons vu que le connecteur ne permet pas le transfert d'un grand nombre de paquets par seconde, la limitation venant de la période d'exécution de la tâche émetteur périodique. Il est cependant possible d'accroître fortement le débit supporté via l'envoi de burst par l'émetteur

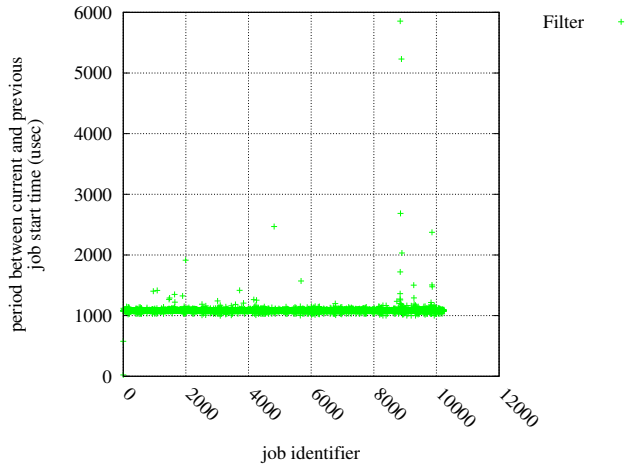

FIGURE III.4.10 – Test 2 : Variation de la période de la tâche Filter

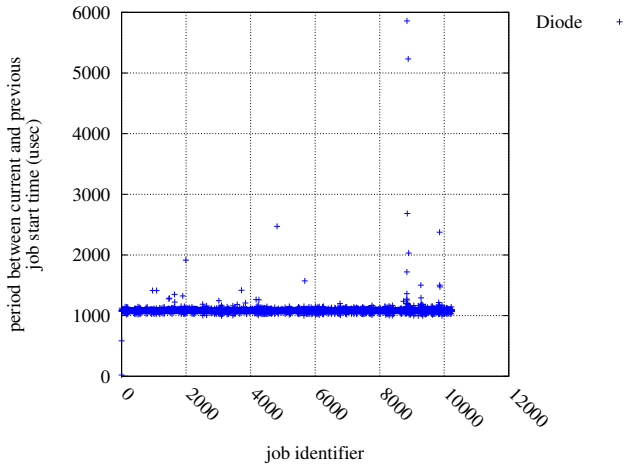

FIGURE III.4.11 – Test 2 : Variation de la période de la tâche Diode

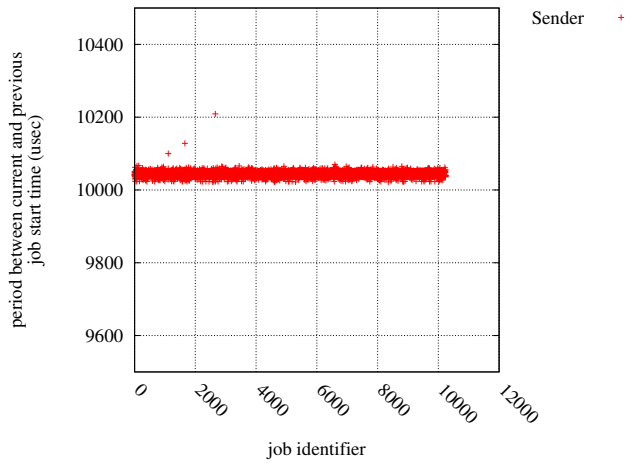

FIGURE III.4.12 – Test 2 : Variation de la période de la tâche Émetteur périodique

périodique. L'usage de bursts permet d'accroître le débit mais implique plusieurs propriétés afin d'assurer malgré tout une maîtrise du profil de flux en sortie du connecteur :

- La taille du burst doit être fixe ;
- Le profil de flux sortant doit conserver un débit maximum déterminé, afin de permettre le calcul de la profondeur des buffers de l'émetteur périodique en conséquence ;
- Les profondeurs de buffers doivent être déterminées en nombre de paquets et non en octets, afin de ne pas être impactés par la variation de la taille des paquets.

Le maquette est basée sur une solution de compartimentation applicative sous Linux. Bien que plusieurs éléments de sécurisation et de temps réel y ait été apportés, cette dernière ne permet ni une certification élevée en sécurité, ni une certification élevée en temps réel. L'usage d'un micro-noyau temps réel est nécessaire. Les différentes fonctions de sécurités étant apportées par les moniteurs de sécurités, hors du champ du micro-noyau. Dans le cadre du transfert de flux à multiple criticité avec de fortes exigences de sécurité, la vérification du flux peut impliquer l'attente de plusieurs paquets successifs afin de reconstruire les informations protocolaires de couche session et supérieures. Ce traitement, devant être fait au niveau du filtre DPI, génère une latence dépendante du nombre de paquets devant être disséqués pour déterminer si le flux peut être autorisé selon la politique de sécurité locale. Ce calcul est néanmoins réalisable pour un protocole de communication donné.

Conclusion générale

Dans le cadre de ma thèse, j'ai proposé une solution afin de résoudre la problématique d'interconnexion des différents réseaux véhiculaires qui ont été déployés dans les systèmes des véhicules militaires et qui commencent à apparaître pour certains d'entre eux dans les véhicules civils. Il s'agit à la fois de réseaux de bord, en général basés sur des bus de type CAN ou équivalents et exigeants de fortes contraintes de sûreté, mais également de réseaux multimédias, dont les niveaux de sécurité varient en fonction des données qu'ils font transiter, et dont la criticité est plus ou moins élevée. Ces réseaux, initialement complètement indépendants, sont aujourd'hui interconnectés afin de permettre le transfert de données vers des interfaces de gestion unifiées locales ou parfois distantes. Néanmoins, les domaines de sécurité et de sûreté associés à ces différents réseaux doivent être respectés :

- en terme de sûreté de fonctionnement, la problématique temps réel est au cœur des contraintes de transmissions de l'information. Les informations doivent être transmises avec une échéance maîtrisée et sans perte dans le cadre d'un traitement nominal. Les domaines de moindre sûreté ne doivent pas impacter le niveau de sûreté des domaines les plus contraints, comme par exemple celui associé au réseau de bord ;
- en terme de sécurité, les données transitant d'un domaine à l'autre sont validées au moment du changement de domaine, et autorisées selon la politique de sécurité associée au changement de domaine.

Le respect des domaines de sécurité et de sûreté implique la présence de passerelles trans-domaines pour assurer une étanchéité suffisante à la fois en terme de sûreté de fonctionnement et en terme de sécurité. Ces passerelles doivent pouvoir garantir un comportement à la fois temps réel et sûr.

Ces passerelles étant déployées dans des véhicules, elles sont également contraintes par des exigences de poids, de consommation, de volume et de certification en terme d'embarcabilité (prise en compte des contraintes de vibrations, de poussière, de température ambiante).

Au regard de la problématique initiale des systèmes Systroniques, le but de ma thèse a été de proposer une architecture logicielle pour une telle passerelle :

- compatible avec les contraintes de sécurité présentes dans les véhicules militaires (confidentialité des données, intégrité, disponibilité),

– compatible avec les contraintes de temps réel et de sûreté de fonctionnement, afin d'agir comme un élément cloisonnant entre un environnement certifié et un environnement non-certifié apte à transmettre de l'information à échéance maîtrisée.

Au delà des contraintes de sécurité et de sûreté de fonctionnement, correspondant à des contraintes métier, j'ai souhaité construire une solution évolutive et paramétrable, afin de pouvoir instancier plusieurs profils de passerelles en m'appuyant sur une modularité intrinsèque à l'architecture logicielle que je propose. Le choix d'architecture a donc été un point crucial dans la recherche de la solution.

Ainsi, au travers de mes travaux, j'ai proposé un framework logiciel générique et évolutif plutôt qu'une solution répondant à un besoin très spécifique. Ce framework a pour but de servir de structure de base pour construire des passerelles d'interconnexion tant pour des besoins sécuritaires que pour des besoins de sûreté de fonctionnement. Les différentes fonctions que je propose doivent être vues comme des modules facultatifs autonomes, à accorder selon le besoin, pour construire une solution complète, et doivent donc être enrichis au fur et à mesure, pour apporter des fonctions pré-construites au framework.
L'usage d'un tel framework permet ainsi de décliner une gamme complète de produits, et donc de répondre à des ensembles de besoins les plus larges possible.

Dans le cadre de ma thèse, j'ai validé la conformité de l'architecture générale en terme de sécurité et de temps réel. Pour cela, je me suis appuyé sur une architecture de type MultiLevel Security, basée sur un micro-noyau considéré de confiance, compatible avec les exigences temps réel au travers d'une certification DO-178B DALB, ainsi que sur une architecture logicielle basée sur Linux dans le cadre des maquettages.

Les contraintes de sécurité que j'ai prises en compte dans le cadre de ma thèse ciblent la confidentialité et l'intégrité des données. La disponibilité n'est pas considérée et nécessiterait quant à elle une étude complémentaire afin de compléter la solution pour intégrer le support des exigences associées.

Aux différentes contraintes de sécurité sont associés des modules logiciels certifiables, visant chacun à implémenter une fonction de sécurité donnée. Ces modules sont appelés *moniteurs de sécurité*. Ces moniteurs sont autonomes et compatibles avec l'ensemble des contraintes de temps réel. La sécurité de l'ensemble de la solution se construit au travers du chaînage de ces différents modules, en fonction des exigences de sécurité requises. Je les associe, selon le besoin, à des machines virtuelles hébergeant un système d'exploitation riche, afin de pouvoir profiter de l'implémentation de piles protocolaires préexistantes. Les machines virtuelles, à l'inverse des moniteurs de sécurité, ne sont pas certifiables du fait de leur volumétrie, et nécessitent donc l'intégration en entrée et en sortie de moniteurs de sécurité en charge de valider les traitements qu'ils effectuent.

La validation des contraintes de temps réel se fait alors au travers de la validation de l'ordonnançabilité de la solution complète, et doit donc être validée pour chaque construction de solution. Cependant, afin de formaliser une telle validation, j'ai défini dans un premier temps

un ensemble d'équations permettant de valider l'ordonnançabilité d'une solution générique sur une architecture MLS, telle qu'utilisée dans ma thèse.

La cible industrielle de ma thèse est un produit embarqué véhiculaire à fortes contraintes de taille, de poids et de consommation énergétique. Cela implique qu'il n'est pas possible de dimensionner l'architecture matérielle pour assurer l'ordonnançabilité de l'ensemble des traitements avec un calcul du WCET très pessimiste. Afin d'assurer malgré tout une garantie de traitement correct, j'ai proposé une solution basée sur une approche à criticité mixte, déterminant hors-ligne quels traitements doivent être assurés et quels traitements peuvent être arrêtés en cas de dépassement de charge impactant l'ordonnançabilité de l'ensemble des tâches. Le choix des traitements n'étant pas soumis au choix d'un flux, l'usage de la criticité mixte peut être un bon complément à l'usage de la Qualité de Service au sens Télécom du terme. Ainsi, plutôt que de prioriser des flux réseau, il est possible de réduire les travaux successifs effectués sur ces flux en fonction du niveau de criticité de chaque travail. Dans le cadre de ma thèse, la solution cible le transfert de divers flux utilisant divers éléments d'entrée/sortie (bus CAN, Ethernet, etc). Elle ne s'appuie donc pas sur le principe de Qualité de Service Réseau, cette dernière étant spécifique au type de média et de protocole utilisé pour faire transiter les données. Néanmoins, je prend comme hypothèse que le réseau est apte à assurer des contraintes de latence sur les flux, et donc supporte de tels principes.

Limites de la solution et perspectives

La solution que je propose permet de répondre à diverses exigences de sécurité et de sûreté de fonctionnement. Pour cela, je propose l'intégration de plusieurs moniteurs de sécurité, comme la diode logicielle ou encore l'émetteur périodique. Néanmoins, seule la diode logicielle a été validée en terme de sécurité, au travers d'une Cible de Sécurité de Premier Niveau (CSPN). Néanmoins, le moniteur diode, tout comme les autres moniteurs, n'ont pas été validés en terme de fiabilité de fonctionnement, au travers d'une validation SIL. Cette validation est nécessaire pour assurer la conformité de l'implémentation du moniteur avec les contraintes de temps réel de la passerelle dans un cadre véhiculaire.

De plus, la présence de compartiments non-certifiable dans le chemin de données, pour le traitement des flux télécom de type IP, ne permet pas de garantir de manière forte une latence de traversée maximum de la passerelle. Pour cela, il faut que l'ensemble des éléments du chemin de données soient temps réel. A défaut de pouvoir utiliser dans ma thèse une pile réseau certifiée ou certifiable, je me suis appuyé sur un noyau Linux utilisant le patch LinuxRT, limitant ainsi le comportement non temps réel du noyau Linux. Dans le cadre d'une véritable passerelle, et en fonction du niveau de certification demandé, il peut être nécessaire de remplacer Linux par un environnement certifiable sachant répondre au même besoin (comme par e.g. la personnalité POSIX du micro-noyau PikeOS, ou au travers de piles réseaux propriétaires).
La présence d'un système d'exploitation GNU/Linux impacte également le niveau de sécurité

du système. Dans le cadre de ma thèse, je me suis prémunis d'une potentielle corruption de ce dernier en m'appuyant sur des moniteurs de sécurité en amont et en aval, afin de limiter fortement l'impact d'une telle corruption. Au final, seule la disponibilité des flux est impactée, si la corruption de l'environnement GNU/Linux se traduit par un blocage des flux. Pour cette raison, je ne répond pas à l'exigence de disponibilité (au sens de la sécurité). Cependant, la corruption de l'environnement GNU/Linux de la passerelle reste complexe pour un attaquant, si on ajoute des protections physiques aux différentes protections logiques apportées par le Separation Kernel et les moniteurs de sécurités. On peut pour cela s'appuyer sur les mécanismes classique de durcissement des équipements militaires, en fonction de leur impact en terme de poids et de volume.

Ainsi, le framework que j'ai proposé permet de répondre à un certain nombre de contraintes formalisées dans le Chapitre III.1, qu'elles soient liées à la sécurité, la sûreté de fonctionnement, ou l'embarquabilité de la solution. En enrichissant les différents modules que j'ai proposé et en implémentant les fonctions décrites comme le support de la criticité mixte, elle permettrait potentiellement de construire des solutions modulaires et certifiables pour répondre aux nouveaux besoins naissants de véhicules connectés et autonomes, tant dans le cadre civil que dans le cadre militaire.

Les différents travaux que nous avons réalisés ont également permis, dans le cadre d'un ordonnancement à criticité mixte, d'intégrer la possibilité de réinitialiser le niveau de criticité d'un jeu de tâches. Néanmoins, ces travaux ont été menés dans un cadre mono-cœur. La problématique de retardement de l'accroissement de criticté, tout comme la capacité à réinitialiser la criticité d'un jeu de tâche exécutée dans un contexte multi-cœurs reste donc entière.

Dans le même temps, l'usage d'ordonnancement hiérarchique, permettant de décorréler des familles de traitements (par exemple pour des raisons de sécurité ou encore de sûreté de fonctionnement), reste aujourd'hui complexe dans le cadre des architectures multi-cœurs. Les interactions et les problématiques de communications entre les différentes fonctions exécutées sur différents cœurs sont également potentiellement impactantes en terme de sécurité (via l'usage potentiel de canaux cachés en mesurant les temps d'accès au bus ou via l'usage des contrôleurs de cache partagés)

Il reste ainsi aujourd'hui un grand nombre de sujets complexes à étudier afin de pouvoir intégrer dans des produits industriels les derniers travaux tant en terme de temps réel que de sécurité logicielle, comme par exemple :

- la capacité à intégrer un comportement temps réel basé sur la criticité mixte sur une architecture de type Multi-Processor System on Chip (MPSoC),
- la capacité de passage à l'échelle (accroissement du débit et du nombre d'interfaces réseau) d'une telle architecture, sur les processeurs spécialisés dans le réseau, comme les processeurs Cavium Octeon ou Freescale CorIQ,
- la possibilité d'intégrer des traitements associés à des domaines de sécurité disjoints sur un même SoC tout en assurant un cloisonnement à la fois logiciel et matériel des chemins

de données (impact de l'usage des contrôleurs de caches locaux et/ou partagés et de l'interconnect),
- l'impact et la compatibilité des travaux sur les contrôleurs de caches tant sur les propriétés temps réel que les propriétés sécuritaires d'un système multi-niveaux de sécurité (MLS),
- l'impact de la virtualisation matérielle des processeurs récents (Freescale QorIQ ou ARM Cortex A15, A53 ou A57), tant en terme de sécurité qu'en terme de latence d'ordonnancement des machines virtuelles,
- la mesure de l'impact de moniteurs de sécurité multiples et complexes dans le cadre de passerelles garantissant un temps de traversée borné, comme par exemple des moniteurs DPI complexes, en charge de détecter des motifs et comportements dynamiques sur des profils de flux divers.

Glossaire des termes techniques

B | C | D | E | F | G | I | M | N | P | S | T

B

Besoin d'en connaître

On appelle droit d'en connaître l'ensemble de connaissances minimum suffisant pour un sujet (une personne ou encore un processus logiciel) pour mener à bien une tâche.

C

Canal Auxiliaire

On appelle canal auxiliaire un élément comportemental lié au fonctionnement de la solution permettant à un attaquant, de manière non-invasive (e.g. via une mesure de temps), de déterminer des informations sur un élément auquel il n'a pas les droits d'accès.

Canal caché

On appelle canal caché tout élément logiciel ou matériel dont il est possible de détourner l'usage afin de pouvoir faire transiter volontairement de l'information vers un sujet n'ayant pas les droits d'accès à celle-ci.

Certification par composition

On appelle certification par composition un principe permettant, lorsque la TCB est non modifiée, de ne certifier que les nouveaux composants venant s'appuyer sur cette dernière. Il n'est alors pas nécessaire de certifier à nouveau l'ensemble de l'architecture logicielle, TCB comprise.

Confidentialité

La confidentialité est la propriété d'un objet à n'être accessible qu'à des sujets ayant les droits suffisants pour accéder à l'information qu'il porte.

D

Deep Packet Inspection

On appelle DPI l'ensemble des fonctions en charge de valider un ou plusieurs flux réseau afin de garantir que son contenu, mais également son comportement (profil, débit, émetteur, destination, etc.) sont conformes d'une politique de sécurité.

Disponibilité

La disponibilité (au sens sécuritaire) est la propriété d'un objet à être accessible et utilisable sur demande d'une entité autorisée.

Domaine de sécurité

On appelle domaine de sécurité un système dans lequel l'ensemble des éléments doivent respecter les mêmes contraintes d'intégrité, de confidentialité et de disponibilité.

Domaine noir

On appelle domaine noir un domaine de sécurité dans lequel les données transitant sont modifiées pour en assurer la confidentialité, l'intégrité et la disponibilité. On dit alors que les données sont noircies.

Domaine rouge

On appelle domaine rouge un domaine de sécurité dans lequel les données transitant ne sont pas protégées. On parle alors de données rouges. Le réseau faisant transiter ces données dans un domaine rouge est alors appelé réseau rouge.

Droit d'en connaître

On appelle droit d'en connaître l'ensemble de connaissances formalisé dans une politique de gestion des droits afin de répondre au(x) besoin(s) d'en connaître d'un sujet.

E

Environnement Virtuel

On appelle Environnement Virtuel (ou VE, Virtual Environment) un compartiment logiciel en charge d'exécuter un environnement logiciel complet, à l'exception du noyau. C'est le cas par exemple des technologies OpenVZ ou LXC [Bha+11]. Une telle architecture logicielle ne nécessite pas de moniteur de machine virtuelle (ou VMM).

Extended Secure Separation Kernel

J'appelle Extended Secure Separation Kernel une architecture logicielle associant un socle logiciel assurant une compartimentation forte et des compartiments certifiables répondant à des exigences de sécurité. L'ensemble est intégrable comme une TCB (Trusted Computing Base) et ses principes de modularité lui permet de répondre à diverses exigences de sécurité.

F

Facteur de forme

On appelle facteur de forme les dimensions, le type d'alimentation, le nombre de ports ou encore le positionnement des connecteurs. Il existe un grand nombre de facteurs de formes industriels, comme le microTCA [Wal08].

Fonction de sécurité

On appelle fonction de sécurité un traitement logiciel spécialisé dans la réponse à un besoin sécuritaire donné. Une fonction de sécurité peut être non-certifiable (par exemple en tant que processus logiciel d'un système d'exploitation riche) ou certifiable (sous forme d'un élément logiciel de petite taille et autonome ou s'appuyant sur un socle lui-même certifiable).

G

Guichet

On appelle guichet un équipement logiciel en charge de transcoder une information d'un réseau à un autre. Il peut par exemple décapsuler une données transmise sur un réseau IP vers un lien point à point. Un guichet peut posséder des propriétés complémentaires, comme la validation de la donnée à transmettre, ou encore le chiffrement de cette dernière.

I

Imputabilité

On appelle imputabilité la garantie de traçabilité entre une action effectuée sur le système et l'identité de l'utilisateur responsable de cette action.

Intégrité

L'intégrité (au sens sécuritaire) est une propriété d'un objet à ne pas avoir été modifié par un quelconque sujet non autorisé.

M

Machine virtuelle

On appelle machine virtuelle (ou VM, Virtual Machine) une implémentation logicielle d'une machine (i.e. d'un ordinateur) qui exécute des programmes comme une machine physique. Il existe deux familles de machines virtuelles :

- une machine virtuelle système, représentant une plateforme matérielle complète et qui supporte l'exécution d'un système d'exploitation ;
- une machine virtuelle de processus, dont le but est d'exécuter un seul programme et qui permet d'assurer la portabilité de ce dernier. C'est par exemple le cas de la machine virtuelle Java.

Moniteur de Machine Virtuelle

On appelle Moniteur de Machine Virtuelle [PG74] (ou VMM, Virtual Machine Monitor) le composant logiciel en charge de permettre l'exécution d'une ou plusieurs machines virtuelles.

Moniteur de sécurité

J'appelle moniteur de sécurité un élément logiciel autonome et certifiable portant une fonction de sécurité. Il a pour charge de répondre à un besoin sécuritaire particulier.

N

Niveau d'assurance

On appelle niveau d'assurance le niveau de certification exigé sur les différents composants logiciels et matériels en terme de fiabilité ou de temps réel. Dans le cadre des systèmes véhiculaires, on s'appuie sur la norme SIL (Safety Integrity Level) [SS10], graduée de 1 (niveau le plus faible) à 4 (niveau le plus fort). Cette dernière s'instancie, dans le cadre du logiciel embarqué véhiculaire, au travers des règles édictées par l'association MISRA (Motor Industry Software Reliability Association) [Ass13].

P

Politique de sécurité

Pour garantir la sécurité d'un domaine de sécurité donné, on définit une politique de sécurité, en charge de préciser l'ensemble des fonctions de sécurité nécessaire au respect des contraintes de sécurité du domaine.

S

Security Target

On appelle Security Target (ST, CdS en français) le document qui définit de manière complète et rigoureuse le problème sécuritaire associant un produit.

Surface d'attaque

On appelle surface d'attaque l'ensemble des portes d'entrées permettant d'impacter la sécurité du système. Il peut s'agir d'une interface réseau non protégée ou encore de processus logiciels possédant des failles exploitables. La mise en place de fonctions de sécurité (comme par exemple un pare-feu) permet de réduire la surface d'attaque.

Système

On appelle système un ensemble d'éléments matériels et logiciels interagissant ensemble dans le but d'effectuer une fonction ou un ensemble de fonctions connexes donné(e).

Système critique

On appelle système critique un système pouvant mettre en danger des vies humaines ou du matériel en cas de défaillance. C'est le cas par exemple des systèmes avioniques ou véhiculaires.

Système de systèmes

On appelle système de systèmes un ensemble complexe faisant s'interconnecter plusieurs systèmes entre eux. Une telle interconnexion peut être par exemple la conséquence d'un besoin de supervision centralisé.

Système rouge-noir

Un système est dit rouge-noir lorsqu'il interconnecte un domaine rouge à un domaine noir. Une telle interconnexion implique un élément de confiance en bordure des domaines,

en charge d'intégrer un mécanisme de protection des données du domaine rouge avant de les transmettre au domaine noir. Cet élément de confiance est également en charge de valider toute transmission venant du domaine noir vers le domaine rouge.

Système sûr

Un système est dit sûr si son niveau d'assurance est compatible avec son niveau de criticité. Il possède donc des propriétés de fiabilité et des propriétés de temps réel adéquates pour assurer la protection des vies humaines ou du matériel correspondant à son niveau de criticité. Les problématiques de sécurité telles que définies dans la Section I.1.2 ne font pas partie des propriétés d'un système sûr.

Système systronique

Lorsqu'un système de systèmes est construit pour être intégré à un véhicule, et est en charge d'associer des systèmes véhiculaires (réseau de bord) à des systèmes de communications complexes, on parle alors de système systronique.

T

Target Of Evaluation

On appelle Target Of Evaluation (TOE) l'ensemble des éléments logiciels et matériels qui seront évalués dans le cadre d'une certification, ainsi que le niveau d'évaluation attendu, au sens des Critères Communs.

Trusted Computing Base

On appelle Trusted Computing Base (TCB) l'ensemble des éléments logiciels et matériels de la solution cible qui sont considérés comme de confiance et dont la corruption génère une faille dans les propriétés de sécurité de la solution.

Bibliographie

[AB98] L. ABENI et G. BUTTAZZO. “Integrating multimedia applications in hard real-time systems”. Dans : *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*. IEEE. 1998, pages 4–13 (cf. page 64).

[Abe+02] Luca ABENI, Ashvin GOEL, Charles KRASIC, Jim SNOW et Jonathan WALPOLE. “A measurement-based analysis of the real-time performance of Linux”. Dans : *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*. IEEE. 2002, pages 133–142 (cf. page 108).

[al09] Betz W. et AL. “Experimental evaluation of the Linux RT Patch for real-time applications”. Dans : *Emerging Technologies & Factory Automation, 2009*. 2009 (cf. page 142).

[Ale+07] R. L. ALENA, J. P. OSSENFORT, K. I. LAWS, A. GOFORTH et F. FIGUEROA. “Communications for integrated modular avionics”. Dans : *Aerospace Conference, 2007 IEEE*. 2007, pages 1–18 (cf. page 141).

[AF+06] Jim ALVES-FOSS, Paul W OMAN, Carol TAYLOR et W Scott HARRISON. “The MILS architecture for high-assurance embedded systems”. Dans : *International journal of embedded systems* 2.3 (2006), pages 239–247 (cf. pages 48, 78).

[ABK98] R. ANDERSON, E. BIHAM et L. KNUDSEN. “Serpent : A proposal for the advanced encryption standard”. Dans : *NIST AES Proposal* (1998) (cf. page 20).

[AMN06] Padma APPARAO, Srihari MAKINENI et Don NEWELL. “Characterization of network processing overheads in Xen”. Dans : *Virtualization Technology in Distributed Computing, 2006. VTDC 2006. First International Workshop on*. IEEE. 2006, pages 2–2 (cf. page 53).

[ARI10] ARINC. *ARINC Report 653 P1-3 : Avionics Application Software Interface, Part 1, Required Services*. 2010 (cf. page 141).

[Ass13] Motor Industry Software Reliability ASSOCIATION. *Motor Industry Software Reliability Association web site*. 2013. URL : `http://www.misra.org.uk/` (cf. pages 7, 162).

[Aud91] N C AUDSLEY. *Optimal Priority Assignment and Feasibility of Static Priority Tasks With Arbitrary Start Times*. Rapport technique YCS-164. 1991 (cf. pages 65, 114).

[Ban11] S. BANESCU. "Cache Timing Attacks". Dans : (2011) (cf. page 38).

[Bar+03] P. BARHAM et al. "Xen and the art of virtualization". Dans : *ACM SIGOPS Operating Systems Review*. Tome 37. 5. ACM. 2003, pages 164–177 (cf. pages 52, 63).

[BLS10a] Sanjoy BARUAH, Haohan LI et Leen STOUGIE. "Mixed-criticality scheduling : improved resource-augmentation results". Dans : *Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2010, pages 13–22 (cf. page 115).

[BLS10b] Sanjoy BARUAH, Haohan LI et Leen STOUGIE. "Towards the Design of Certifiable Mixed-criticality Systems". Dans : *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2010, pages 13–22 (cf. page 66).

[Bar+10] Sanjoy K BARUAH et al. "Scheduling real-time mixed-criticality jobs". Dans : *Mathematical Foundations of Computer Science 2010*. Springer, 2010, pages 90–101 (cf. pages 66, 110, 114).

[BBD11] S.K. BARUAH, A. BURNS et R. I. DAVIS. "Response-Time Analysis for Mixed Criticality Systems". Dans : *32nd IEEE Real-Time Systems Symposium*. IEEE Computer Society, 2011, pages 34–43 (cf. page 66).

[BEB02] Robert T BAUM, Edward M EGGERL et William R BURTON. *Protocol separation in packet communication*. US Patent 6,456,632. 2002 (cf. page 84).

[BB09] Christoph BAUMANN et Thorsten BORMER. "Verifying the PikeOS Microkernel : First Results in the Verisoft XT Avionics Project". Dans : *Doctoral Symposium on Systems Software Verification (DS SSV'09) Real Software, Real Problems, Real Solutions*. 2009, page 20 (cf. page 58).

[Bel05] D. E. BELL. "Looking back at the bell-la padula model". Dans : *Computer Security Applications Conference, 21st Annual*. 2005, pages 337–351. URL : `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1565261` (cf. page 46).

[BL73] D. E. BELL et L. J. LAPADULA. *Secure computer systems : Mathematical foundations*. Rapport technique. DTIC Document, 1973 (cf. page 46).

[Ber+05] Vincent BERK, Annarita GIANI, George CYBENKO et NH HANOVER. “Detection of covert channel encoding in network packet delays”. Dans : *Rapport technique TR536, de lUniversité de Dartmouth. Novembre* (2005) (cf. pages 19, 20).

[Ber05] D.J. BERNSTEIN. *Cache-timing attacks on AES*. 2005 (cf. pages 38, 94).

[Bet+09] D.D. BETTGER et al. *Secure high performance multi-level security database systems and methods*. US Patent App. 12/497,408. 2009 (cf. pages 48, 49).

[Bha+11] Gautam BHANAGE, Ivan SESKAR, Yanyong ZHANG, Dipankar RAYCHAUDHURI et Shweta JAIN. “Experimental evaluation of openvz from a testbed deployment perspective”. Dans : *Testbeds and Research Infrastructures. Development of Networks and Communities*. Springer, 2011, pages 103–112 (cf. pages 24, 160).

[BB04] E. BINI et G. C. BUTTAZZO. “Schedulability Analysis of Periodic Fixed Priority Systems”. Dans : *IEEE Transaction on Computers* (2004), pages 1462–1473 (cf. pages 117, 121).

[BDNB06] Enrico BINI, Marco DI NATALE et Giorgio BUTTAZZO. “Sensitivity Analysis for Fixed-Priority Real-Time Systems”. Dans : *18th Euromicro Conference on Real-Time Systems*. IEEE Computer Society, 2006, pages 13–22 (cf. page 117).

[Blu+11] N. BLUM, J. YAMADA, A. FUKAYAMA, T. MAGEDANZ et N. UCHIDA. “A Smart Information Sharing Architecture in a Multi-Access Network, Multi-Service Environment”. Dans : *Communications (ICC), 2011 IEEE International Conference on*. IEEE. 2011, pages 1–6 (cf. page 48).

[Boe+08] Carolyn BOETTCHER, Rance DELONG, John RUSHBY et Wilmar SIFRE. “The MILS component integration approach to secure information sharing”. Dans : *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*. IEEE. 2008, pages 1–C (cf. page 48).

[BM06] J. BONNEAU et I. MIRONOV. “Cache-collision timing attacks against AES”. Dans : *Cryptographic Hardware and Embedded Systems-CHES 2006* (2006), pages 201–215 (cf. page 38).

[BGM07] Lamine BOUGUEROUA, Laurent GEORGE et Serge MIDONNET. “Dealing with execution-overruns to improve the temporal robustness of real-time systems scheduled FP and EDF”. Dans : *Second International Conference on Systems ICONS’07*. IEEE. 2007, page 52 (cf. pages 15, 110, 117, 119).

[CIBM01] Marti CAMPOY, A Perles IVARS et JV BUSQUETS-MATAIX. “Static use of locking caches in multitask preemptive real-time systems”. Dans : *Proceedings of IEEE/IEE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium)*. Citeseer. 2001 (cf. page 39).

[CS07] Jichuan CHANG et Gurindar S. SOLHI. “Cooperative Cache Partitioning for Chip Multiprocessors”. Dans : (2007) (cf. page 95).

[CDV07] Lucy CHERKASOVA, Gupta DIWAKER et Amin VAHDAT. *Comparison of the three CPU schedulers in Xen*. Rapport technique. 2007 (cf. page 52).

[CGV07] Ludmila CHERKASOVA, Diwaker GUPTA et Amin VAHDAT. “Comparison of the three CPU schedulers in Xen”. Dans : *Performance Evaluation Review* 35.2 (2007), page 42 (cf. page 53).

[Com95] TIS COMMUNITY. *Executable and Linking Format (ELF) Specification*. Rapport technique 1.2. Tool Interface Standard, 1995 (cf. page 28).

[CAA08] T. CUCINOTTA, G. ANASTASI et L. ABENI. “Real-time virtual machines”. Dans : *Proceedings of the 29th IEEE Real-Time System Symposium (RTSS 2008)–Work in Progress Session, Barcelona*. Citeseer. 2008 (cf. pages 63, 64).

[CAA09] T. CUCINOTTA, G. ANASTASI et L. ABENI. “Respecting temporal constraints in virtualised services”. Dans : *Computer Software and Applications Conference, 2009. COMPSAC’09. 33rd Annual IEEE International*. Tome 2. 2009, pages 73–78. URL : `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5254145` (cf. pages 53, 63).

[DR98] J. DAEMEN et V. RIJMEN. “AES proposal : Rijndael”. Dans : (1998) (cf. page 20).

[DS09] I. DINUR et A. SHAMIR. “Side Channel Cube Attacks on Block Ciphers”. Dans : *IACR ePrint Archive, ePrint* 127 (2009) (cf. page 20).

[DWSHAF05] Dr. Paul W. Oman DR. W. SCOTT HARRISON Dr. Nadine Hanebutte et Dr. Jim ALVES-FOSS. *CrossTalk : The Journal of Defense Software Engineering. Volume 18, Number 10*. Rapport technique. DTIC Document, 2005 (cf. page 47).

[FCK95] David FERRAIOLO, Janet CUGINI et D Richard KUHN. “Role-based access control (RBAC) : Features and motivations”. Dans : *Proceedings of 11th Annual Computer Security Application Conference*. sn. 1995, pages 241–48 (cf. page 46).

[Fra+04] Keir FRASER et al. “Safe hardware access with the Xen virtual machine monitor”. Dans : *1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS)*. 2004 (cf. page 52).

[Fre08] FREESCALE. *QorIQ P4080 Communications Processor Product Brief*. Rapport technique 1.09. Freescale Semiconductor, Inc, 2008 (cf. page 40).

[Gad+12] Francesco GADALETA, Raoul STRACKX, Nick NIKIFORAKIS, Frank PIESSENS et Wouter JOOSEN. "On the effectiveness of virtualization-based security". Dans : *status : accepted* (2012) (cf. page 54).

[Ger06] Fabien GERMAIN. "Sécurité cryptographique par la conception spécifique de circuits intégrés". Thèse de doctorat. 2006 (cf. page 20).

[Gov+07] Sriram GOVINDAN, Arjun R NATH, Amitayu DAS, Bhuvan URGAONKAR et Anand SIVASUBRAMANIAM. "Xen and co. : communication-aware cpu scheduling for consolidated xen-based hosting platforms". Dans : *Proceedings of the 3rd international conference on Virtual execution environments*. ACM. 2007, pages 126–136 (cf. page 53).

[GWV03] David GREVE, Matthew WILDING et W Mark VANFLEET. "A separation kernel formal security policy". Dans : *Fourth International Workshop on the ACL2 Prover and Its Applications (ACL2-2003), Boulder, CO*. Citeseer. 2003 (cf. page 48).

[GK08] Stefan GROESBRINK et Timo KERSTAN. "Modular paging with dynamic TLB partitioning for embedded real-time systems". Dans : *Industrial Embedded Systems, 2008. SIES 2008. International Symposium on*. IEEE. 2008, pages 261–264 (cf. page 39).

[Gua+11] Nan GUAN, Pontus EKBERG, Martin STIGGE et Wang YI. "Effective and Efficient Scheduling of Certifiable Mixed-Criticality Sporadic Task Systems". Dans : *32nd IEEE Real-Time Systems Symposium*. IEEE Computer Society, 2011, pages 13–23 (cf. page 66).

[Gua+12] Nan GUAN, Mingsong LV, Wang YI et Ge YU. "WCET Analysis with MRU Caches : Challenging LRU for Predictability". Dans : *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*. IEEE. 2012, pages 55–64 (cf. pages 33, 39).

[HHD11] Wei HAN, Yeping HE et Liping DING. "Verifying the Safety of Xen Security Modules". Dans : *Secure Software Integration & Reliability Improvement Companion (SSIRI-C), 2011 5th International Conference on*. IEEE. 2011, pages 30–34 (cf. page 54).

[Han78] Wilfred J HANSEN. "Measurement of program complexity by the pair :(Cyclomatic Number, Operator Count)". Dans : *ACM SIGPLan Notices* 13.3 (1978), pages 29–33 (cf. page 79).

[Hei+06] Constance L HEITMEYER, Myla ARCHER, Elizabeth I LEONARD et John MCLEAN. “Formal specification and verification of data separation in a separation kernel for an embedded system”. Dans : *Proceedings of the 13th ACM conference on Computer and communications security*. ACM. 2006, pages 346–355 (cf. page 48).

[HAT07] Henri HEMERY, Walter AKMOUCHE et Frédéric TATOUT. “Utilisation de la methodologie EBIOS en securite globale”. Dans : *REE. Revue de l’électricité et de l’électronique* 10 (2007), pages 29–125 (cf. page 81).

[KW07] Robert KAISER et Stephan WAGNER. “Evolution of the PikeOS microkernel”. Dans : *First International Workshop on Microkernels for Embedded Systems*. 2007, page 50 (cf. pages 56, 107).

[Kes09] U. KESKIN. “In-vehicle networks : a litterature survey”. Dans : (2009) (cf. page 132).

[Kiv+07] A. KIVITY, Y. KAMAY, D. LAOR, U. LUBLIN et A. LIGUORI. “kvm : the Linux virtual machine monitor”. Dans : *Proceedings of the Linux Symposium*. Tome 1. 2007, pages 225–230 (cf. page 63).

[Kle+09] Gerwin KLEIN et al. “seL4 : Formal verification of an OS kernel”. Dans : *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM. 2009, pages 207–220 (cf. pages 53, 56).

[KE09] Kushal KOOLWAL et RDOS ENGINEER. “Myths and realities of real-time linux software systems”. Dans : *Proc. Real-Time Linux Workshop (RTLWS 2009)*. 2009 (cf. page 108).

[Kum+06] Sailesh KUMAR, Sarang DHARMAPURIKAR, Fang YU, Patrick CROWLEY et Jonathan TURNER. “Algorithms to accelerate multiple regular expressions matching for deep packet inspection”. Dans : *ACM SIGCOMM Computer Communication Review* 36.4 (2006), pages 339–350 (cf. page 93).

[lab12] open source automation development LAB. *latency plots for real-time tasks using cycletest*. 2012. URL : `https://www.osadl.org` (cf. page 143).

[Lag06] P. LAGADEC. “Diode réseau et ExeFilter : 2 projets pour des interconnexions sécurisées”. Dans : *Proc. SSTIC* (2006) (cf. pages 45, 49, 90).

[LNR11] Karthik LAKSHMANAN, Dionisio de NIZ et Ragunathan RAJKUMAR. “Mixed-Criticality Task Synchronization in Zero-Slack Scheduling”. Dans : *17th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2011, pages 47–56 (cf. page 66).

[Lam73] Butler W LAMPSON. “A note on the confinement problem”. Dans : *Communications of the ACM* 16.10 (1973), pages 613–615 (cf. page 140).

[Lew+07] Mark LEWANDOWSKI, Mark J STANOVICH, Theodore P BAKER, Kartik GOPALAN et A-IA WANG. "Modeling device driver effects in real-time schedulability analysis : Study of a network driver". Dans : *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS'07. 13th IEEE*. IEEE. 2007, pages 57–68 (cf. page 108).

[LB10] Haohan LI et Sanjoy BARUAH. "An Algorithm for Scheduling Certifiable Mixed-Criticality Sporadic Task Systems". Dans : *31st IEEE Real-Time Systems Symposium*. Washington, DC, USA : IEEE Computer Society, 2010, pages 183–192 (cf. page 66).

[Lie95] Jochen LIEDTKE. "On micro-kernel construction". Dans : *ACM SIGOPS Operating Systems Review* 29.5 (1995), pages 237–250 (cf. page 56).

[Lie96] Jochen LIEDTKE. "Toward real microkernels". Dans : *Communications of the ACM* 39.9 (1996), pages 70–77 (cf. page 56).

[LB05] G. LIPARI et E. BINI. "A methodology for designing hierarchical scheduling systems". Dans : *Journal of Embedded Computing* 1.2 (2005), pages 257–269 (cf. page 62).

[Liu00] Jane W. S. LIU. *Real-Time Systems*. 1st. Upper Saddle River, NJ, USA : Prentice Hall, 2000. ISBN : 0130996513 (cf. pages 113, 114).

[MS+10] M. MAROUF, Y. SOREL et al. "Schedulability conditions for non-preemptive hard real-time tasks with strict period". Dans : *Proceedings of the 18th International Conference on Real-Time and Network Systems*. 2010, pages 50–58 (cf. page 106).

[MM06] D. MUROTAKE et A. MARTIN. "A high assurance wireless computing system (HAWCS) for software defined radio". Dans : *Proceeding of the SDR 06 Technical Conference and Product Expositi*. 2006 (cf. page 46).

[NLR09] Dionisio de NIZ, Karthik LAKSHMANAN et Ragunathan RAJKUMAR. "On the Scheduling of Mixed-Criticality Real-Time Task Sets". Dans : *30th IEEE Real-Time Systems Symposium*. Washington, DC, USA : IEEE Computer Society, 2009, pages 291–300 (cf. page 66).

[OST06] D. OSVIK, A. SHAMIR et E. TROMER. "Cache attacks and countermeasures : the case of AES". Dans : *Topics in Cryptology–CT-RSA 2006* (2006), pages 1–20 (cf. page 20).

[Per05] C. PERCIVAL. "Cache missing for fun and profit". Dans : *BSDCan 2005* (2005) (cf. pages 37, 94).

[PG74] G. J. POPEK et R. P. GOLDBERG. “Formal Requirements for Virtualizable Third Generation Architectures”. Dans : *Communications of the ACM 17, no. 7* (1974), pages 412–421 (cf. pages 23, 24, 25, 161).

[Pro14] The Common Criteria PROJECT. *Common Criteria Evaluation & Validation Scheme*. 2014. URL : http://www.niap-ccevs.org/Documents_and_Guidance/cc_docs.cfm (cf. pages 6, 46).

[Pua06] Isabelle PUAUT. “WCET-centric software-controlled instruction caches for hard real-time systems”. Dans : *Real-Time Systems, 2006. 18th Euromicro Conference on*. IEEE. 2006, 10–pp (cf. page 39).

[Reg+03] J. REGEHR, A. REID, K. WEBB, M. PARKER et J. LEPREAU. “Evolving real-time systems using hierarchical scheduling and concurrency analysis”. Dans : *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*. 2003, pages 25–36. URL : http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1253251 (cf. page 104).

[Iso] *Road vehicles – Controller area network (CAN)*. Norm. 2003 (cf. page 8).

[RO08] Bernhard Frömel et al. ROMAN OBERMAISSER. “Integrating Safety and Multimedia Subsystems on a Time-Triggered System-on-a-Chip”. Dans : *International Symposium on Industrial Electronics, ISIE 2008*. 2008 (cf. page 132).

[Rus81] John RUSHBY. “Design and Verication of Secure Systems”. Dans : *ACM Operating System Review vol. 15* (1981), pages 12–21 (cf. pages 46, 48).

[RW10] Joanna RUTKOWSKA et Rafal WOJTCZUK. “Qubes os architecture”. Dans : *Invisible Things Lab Tech Rep* (2010) (cf. page 54).

[SS94] Ravi S SANDHU et Pierangela SAMARATI. “Access control : principle and practice”. Dans : *Communications Magazine, IEEE* 32.9 (1994), pages 40–48 (cf. page 21).

[Sch10] R. SCHAEFFER. *National Information Assurance (IA) Glossary*. 2010 (cf. pages 5, 47).

[Sec12] Thales SECURITY. *Trusted Security Filter (TSF 101)*. 2012. URL : http://jp.thales-esecurity.com/Products/Security\%20Filters/TSF101.aspx (cf. pages 9, 45, 49, 90).

[Sem05] Freescale SEMICONDUCTOR. *PowerPC e500 Core Family Reference Manual*. Rapport technique. 2005 (cf. page 101).

[SS10] David J SMITH et Kenneth GL SIMPSON. *Safety Critical Systems Handbook : A STRAIGHTFOWARD GUIDE TO FUNCTIONAL SAFETY, IEC 61508 (2010 EDITION) AND RELATED STANDARDS, INCLUDING PROCESS IEC 61511 AND MACHINERY IEC 62061 AND ISO 13849*. Access Online via Elsevier, 2010 (cf. pages 7, 162).

[Smi+08] Randy SMITH, Cristian ESTAN, Somesh JHA et Shijin KONG. "Deflating the big bang : fast and scalable deep packet inspection with extended finite automata". Dans : *ACM SIGCOMM Computer Communication Review* 38.4 (2008), pages 207–218 (cf. page 93).

[Sta12] International Organization for STANDARDIZATION. *Road vehicles – Functional safety – Part 10 : Guideline on ISO 26262*. 2012. URL : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54591 (cf. page 94).

[Ste+09] Wilfried STEINER, Günther BAUER, Brendan HALL, Michael PAULITSCH et Srivatsan VARADARAJAN. "Ttethernet dataflow concept". Dans : *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*. IEEE. 2009, pages 319–322 (cf. page 135).

[Sys13] SYSGO. *PikeOS SIL 4 certification on multi-core plateform*. 2013. URL : http://www.sysgo.com/news-events/press/press/details/article/pikeosTM-sil-4-certification-on-multi-core-platform/ (cf. page 94).

[Lxc] *The Linux Containers*. 2013. URL : http://lxc.sourceforge.net (cf. page 55).

[TRW09] Yuan TIAN, Guoqiang REN et Qinzhang WU. "Implementation of Real-time Network Extension on Embedded Linux". Dans : *Communication Software and Networks, 2009. ICCSN'09. International Conference on*. IEEE. 2009, pages 163–167 (cf. page 108).

[Ose] *Time-Triggered Operating System*. OSEK. 2001 (cf. page 112).

[Ves07] Steve VESTAL. "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance". Dans : *Proceedings of the 28th IEEE International Real-Time Systems Symposium*. IEEE Computer Society, 2007, pages 239–243 (cf. pages 65, 114).

[Wal08] John B WALROD. "Open-standard ATCA and microTCA platforms for ocean observatories". Dans : *OCEANS 2008-MTS/IEEE Kobe Techno-Ocean*. IEEE. 2008, pages 1–7 (cf. pages 5, 160).

[Whi+04] Peter Duncan WHITE et al. *Separation kernel with memory allocation, remote procedure call and exception handling mechanisms*. US Patent 6,772,416. 2004 (cf. page 48).

[Wik10] WIKIPEDIA. "Cache algorithms". Dans : (2010) (cf. page 31).

[Wik09] WIKIPEDIA. "Pseudo Least Recently Usedalgorithm". Dans : (2009) (cf. page 35).

[Wil03] Matthew WILCOX. "I'll do it later : Softirqs, tasklets, bottom halves, task queues, work queues and timers". Dans : *Linux. conf. au*. 2003 (cf. page 62).

[Xu+11] Yunjing XU et al. "An exploration of L2 cache covert channels in virtualized environments". Dans : *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM. 2011, pages 29–40 (cf. page 21).

[Zha+12] Yinqian ZHANG, Ari JUELS, Michael K REITER et Thomas RISTENPART. "Cross-VM side channels and their use to extract private keys". Dans : *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM. 2012, pages 305–316 (cf. page 21).

VEHICULAR SYSTEMS WITH MULTIPLE SECURITY AND CRITICALITY DOMAINS : A REAL-TIME SYTRONIC GATEWAY

Abstract

Nowadays, vehicular systems are composed of more and more interconnected systems. Those systems manage a lot of complex functions and must comply with various safety-critical requirements (such as real-time) but also more and more with security requirements. With the new connected vehicles, it is necessary to make these various systems communicate, in order to manage locally or remotely the overall vetronic system.
Make these systems communicate, moreover in military vehicles, implies to support various constraints. Theses constraints need to be supported by specific elements, used as gateways between each vehicle system needing external communication. This gateway has to protect each system in term of safety and security, but also has to guarantee an efficient upper-bounded transfer between them.

In this thesis, we have proposed a software architecture for these gateways, compliant with the various vehicular security and safety requirements. The solution is proposed as a framework, supporting a modular configuration and able to aggregate various modules on a partitioned software architecture. Such an aggregation is then able to respond to the various vehicular specific needs such as security and real-time.

Keywords : Real-Time, Security, Virtualization, Vetronic, Hypervisor, MLS

SYSTÈMES VÉHICULAIRES À DOMAINES DE SÉCURITÉ ET DE CRITICITÉ MULTIPLES : UNE PASSERELLE SYSTRONIQUE TEMPS RÉEL

Résumé

De nos jours, les véhicules intègrent de plus en plus de systèmes interconnectés. Ces systèmes ont des fonctions aussi nombreuses que complexes et sont soumis à des contraintes de sureté de fonctionnement (dont le temps réel) mais également de plus en plus de sécurité. Avec l'apparition des véhicules connectés, il devient nécessaire de faire communiquer ces différents systèmes, tant pour les gérer au niveau véhiculaire que potentiellement à distance.
Faire communiquer ces différents réseaux, a fortiori dans les véhicules militaires, implique la prise en compte de diverses contraintes. Ces contraintes nécessitent d'être traitées par des éléments en coupure entre les différents systèmes. Un tel élément est alors en charge de protéger ces derniers en terme de sûreté de fonctionnement et de sécurité mais doit également assurer un transfert efficace et borné de l'information.

Dans cette thèse, nous avons proposé une architecture logicielle de passerelle permettant de répondre à ces différentes contraintes et d'assurer ainsi l'interconnexion de tous ces systèmes. La solution se présente comme un framework permettant d'intégrer divers modules sur une architecture partitionnée et sûre, afin de pouvoir répondre à divers besoins spécifiques au systèmes véhiculaires.

Mots-clefs : Temps Réel, Sécurité, Virtualisation, Vétronique, Hyperviseur, MLS