Garantir les temps de réponse des réseaux embarqués à l'aide du calcul réseau

Marc Boyer

En vue de l'obtention de l'Habilitation à Diriger des Recherches

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *18 mars 2021* **par :**

Marc Boyer

Garantir les temps de réponse des réseaux embarqués à l'aide du calcul réseau

---

JURY

Jean-Yves Le Boudec	Professeur des universités	examinateur
Anne Bouillard	Chercheuse	examinatrice
Emmanuel Chaput	Professeur des universités	examinateur
Emmanuel Grolleau	Professeur des universités	rapporteur
Isabelle Puaut	Professeure des universités	rapportrice
Jean-Luc Scharbarg	Professeur des universités	correspondant
Ye-Qiong Song	Professeur des universités	rapporteur

Sommaire

Remerciements

J'ai eu la chance de croiser des dizaines (centaines peut-être même, avec les années) de gens formidables dans mon travail de recherche, et les citer tous nécessiterait un nombre de pages déraisonnable.

Je voudrais d'abord remercier Philippe Owezarsky, qui finissait sa thèse lorsque je la débutais, dont j'ai partagé le bureau pendant 4 ans, et qui m'a servi de conseiller, peut-être même de modèle, dans mon début de carrière. Pour l'anecdote, alors que nous parcourions le programme d'une journée de séminaire, il m'a pointé un exposé sur la théorie du calcul réseau, en me disant "va écouter ça, ça te plaira". À l'époque, j'étais plongé dans "mes" réseaux de Petri et je n'imaginais pas que ce calcul réseau serait mon sujet de recherche principal pendant plus de vingt ans.

Je dois ensuite remercier Christian Fraboul et Jean-Luc Sharbarg qui ont su m'introduire au monde des réseaux embarqués temps-réels, de leurs enjeux, et aux analyses formelles, dont le calcul réseau. Il me faut aussi remercier le reste de l'équipe d'enseignants en réseau, et particulièrement André-Luc Beylot et Emmanuel Chaput qui ont pris le temps de me présenter les détails des architectures réseaux, PDU, trames, paquet, SAP, routage, commutation, etc.

Je voudrais non seulement remercier mais saluer Olivier H. Roux, qui m'a beaucoup appris sur la rédaction de preuves dans les systèmes formels, mais aussi par son attitude sociale dans le monde de la recherche, et sa bonne humeur.

J'ai la chance d'avoir à l'ONERA des collègues de travail formidables. Les premiers sont bien sûr Claire Pagetti et Frédéric Boniol qui m'ont accueilli à l'ONERA, guidé, accompagné et parfois soutenu. Même si nous n'avons que peu de publications communes, nos travaux étant complémentaires, nous avons partagé des projets de recherches, des discussions, scientifiques et personnelles, porté une réflexion sur les stratégies de recherche, etc.

De l'ONERA, je dois aussi remercier David Doose, Guillaume Duffour et Pierre Roux qui m'ont aidé et accompagné en partageant leurs expertises res-

pectives en temps réel et en mathématiques, le tout accompagné de discussions agréables de culture générale scientifique.

Toujours à l'ONERA, je dois remercier Thomas Polacsek qui, en plus d'être devenu un ami (ce qui n'a rien de scientifique), porte une réflexion sur la recherche en général et l'informatique embarqué en particulier dont je me suis beaucoup nourris.

Et pour finir avec l'ONERA, je voudrais remercier Virginie Wiels pour ses valeureux conseils et son soutien, toujours par petites touches subtiles, comme collègues pendant quelques années, puis comme "responsable hiérarchique" par la suite, conseils et partages sur la recherche, le travail, et les enfants parfois aussi.

Le monde de la recherche est un monde particulier où le travail effectif se fait parfois plus avec des personnes à 5km comme à 500km, qu'avec des collègues de couloir, par la magie [1] des systèmes de travail à distance.

Je voudrais donc remercier Anne Bouillard, Eric Thierry et Euriell Le Corronc qui m'ont accompagné dans ma montée en compétences dans le domaine du calcul réseau, dès le projet PEGASE (pour Anne et Thierry) jusqu'à la rédaction de notre livre, avec Anne et Euriell [20].

Les doctorants que j'ai encadré ou encadre encore, William Mangoua Sofack Lucien Rakotomalala et Pierre-Julien Chaine m'ont tous beaucoup apporté, et je les en remercie, mais je me dois de remercier plus particulièrement Hugo Daigmorte qui a fait non seulement sa thèse mais deux stages avec moi, et avec lequel j'ai partagé une proximité scientifique rare.

Il me faut aussi remercier Nicolas Navet, Jörn Migge, Lionel Havet et Jérome Michel de la société RealTime-at-Work. Nous nous sommes rencontrés dans le projet PEGASE, financé par l'ANR. Ils étaient experts du bus temps réel CAN (entre autre) et se proposaient dans le projet d'implanter les algorithmes développés dans le projet. Ils ont pu ensuite transformer ce prototype en outil d'analyse de réseau, reprenant le nom du projet "PEGASE", offrant des outils de conception, simulation et analyse à base de calcul réseau. J'ai avec cette société une relation symbiotique, assez rare et souvent difficile à faire comprendre dans un environnement qui ne pense les relations entre institution qu'entre concurrence, sous-traitance, droits de licence ou complémentarité orthogonale. Cette relation symbiotique fait que nous sommes très souvent partenaires dans des projets de recherche : le donneur d'ordre (agence gouvernementale, client industriel) sait qu'il pourra au sein du même projet bénéficier d'une expertise théorique académique et d'un prototypage de qualité industrielle. J'ajoute qu'ils ont un relationnel extrêmement agréable, courtois, et une forte expertise technique qui fait qu'ils m'ont déjà corrigé des erreurs (par pudeur, je ne les listerais pas ici), suggéré des améliorations. Nous avons co-signé plus d'une dizaine d'article de recherche, et je n'ai pas signalé dans ce manuscrit leur implication à chaque fois

1. Un système informatique acquiert à mon sens le niveau "magique" quand il fonctionne correctement de façon transparente. La réussite d'une technologie, c'est de devenir invisible.

de peur d'être très répétitif. C'est aussi eux qui m'ont conseillé de regarder AVB, puis son évolution TSN, bien avant que ce soient des sujets de premier plan dans le domaine des réseaux.

Je voudrais aussi remercier tous les chercheurs qui ont porté cette théorie du calcul réseau sur laquelle je prends tant de plaisir à travailler, et en premier Jean-Yves Le Boudec que j'ai d'abord admiré pour la qualité de ses contributions scientifique et dont je peux depuis peu apprécier toutes les qualités humaines. Et il y a toute la communauté issue du groupe de recherche DISCO, dirigé par Jens B. Schmitt : Steffen Bondorf, Markus Fidler, Amr Rizk et Florin Ciucu, entre autres. Et ce point singulier de l'autre côté de l'océan, Jorg Liebeherr, que j'aimerais rencontrer plus souvent.

Pour finir, je voudrai remercier Isabelle Puaut, Ye-Qiong Song et Emmanuel Grolleau d'avoir accepté d'examiner mon manuscrit et d'en faire le rapport, ainsi qu'Anne Bouillard, Jean-Yves Le Boudec, Jean-Luc Scharbarg et Emmanuel Chaput, déjà cités précédemment, pour avoir accepter de faire partie de mon jury d'habilitation à diriger les recherches (HDR). Et merci d'avoir malmené ma modestie.

On voit donc qu'il ne faut pas trop tarder à passer son HDR dans sa carrière : ayant côtoyé tellement de gens, cela donne trois pages de remerciements, où chacun n'a pas eu assez pour lui rendre hommage et où d'autre sont absents, par manque de place, pas par manque de reconnaissance.

Introduction

Dans les grandes découvertes, il y a le temps des explorateurs, et le temps des ingénieurs. Dans le far-west, le temps des premiers pionniers, puis le temps du chemin de fer, avant l'arrivée du grand nombre.

Je m'imagine bien comme un de ces premiers ingénieurs du chemin de fer : je ne suis pas un des glorieux découvreurs d'idée novatrice, je suis de ceux qui cartographient, qui consolident, qui essayent de transformer les précieuses idées initiales en une théorie utilisable par l'industrie, avant une mise oeuvre dans des systèmes à disposition du grand public.

Une habilitation à diriger les recherches (HDR) L'arrêté du 23/11/1988 "relatif à l'habilitation à diriger les recherches" définit cette habilitation comme un diplôme qui "sanctionne la reconnaissance

— du haut niveau scientifique du candidat,

— du caractère original de sa démarche dans un domaine de la science,

— de son aptitude à maîtriser une stratégie de recherche dans un domaine scientifique ou technologique suffisamment large,

— et de sa capacité à encadrer de jeunes chercheurs."

La commission recherche de l'Institut National Polytechnique de Toulouse (INPT) précise ses attentes : "Pour présenter ses travaux et son projet, le candidat rédige un mémoire de synthèse qui comprend :

— soit un ou plusieurs ouvrages publiés,

— soit un dossier de travaux, accompagné d'une synthèse de l'activité scientifique permettant de faire apparaître son expérience dans l'animation d'une recherche,

— son projet scientifique.

Le mémoire est une synthèse et une mise en cohérence de ses travaux passés et un développement argumenté de ses perspectives scientifiques et de ses projets de recherche. Il n'a pas vocation à s'apparenter à un manuscrit de thèse et doit s'appuyer sur les publications les plus significatives qui pourront même être intégrées dans le corps du manuscrit.

Il insistera particulièrement sur la partie présentant ses perspectives scientifiques qui doit constituer le cœur de votre demande de HDR."

Je voulais rappeler ces définitions tant "diriger" est de nos jours associé à "directeur", "dirigeant" lui même associé à une relation de pouvoir entre un chef et ses subordonnés. Mais l'étymologie latine, *director*, signifie "le guide". Le directeur, c'est avant tout celui qui a une "direction" au sens d' "objectif", c'est à dire savoir où aller, et par ou passer.

Dans un contexte français, où les titulaires d'HDR n'encadrent pas forcément d'équipe, il faut entendre le terme dans son sens "géographique" (j'aurais dit topologique pour faire matheux) : savoir tout d'abord se situer et se diriger dans un contexte scientifique, puis éventuellement guider de jeunes chercheurs.

Je vais essayer de vous montrer dans ce manuscrit pourquoi je pense avoir, avec le temps, acquis cette capacité à "diriger des recherches".

Ce que ce document n'est pas Pour certains chercheurs, le mémoire d'HDR est l'occasion de mettre dans un même document, dans un même formalisme, un ensemble de travaux réalisés et publiés dans des articles disséminés, et d'en discuter des points techniques que le format des publications en congrès ou journaux ne favorise pas. Pour ma part, j'ai eu l'occasion de faire ce travail dans l'ouvrage co-écrit avec Anne Bouillard et Euriell Le Corronc, [20]. Il n'y aura donc que très peu de détails techniques dans ce manuscrit, qui se concentrera sur la présentation d'une démarche et d'orientations.

Il est aussi courant dans un mémoire d'HDR, de mettre de nombreux indicateurs quantitatifs : nombre de publications de tel type, nombre d'encadrements, nombre de participations à des comités de programme, budget des projets de recherche, etc. Ces indicateurs sont bien sûr nécessaires, et je les ai fournis à la commission recherche de l'INPT, qui les a jugés suffisants pour autoriser mon inscription. Mais je ne les ai pas reportés dans ce manuscrit, car je me méfie de la focalisation sur des indicateurs, qui deviennent peu à peu des objectifs, et cessent d'être de bon indicateurs (ce qui a été popularisé par Charles Goodhart en 1975 dans le domaine économique, "*When a measure becomes a target, it ceases to be a good measure*" [78], puis par Donald T. Campbell dans le domaine des politiques publiques "*The more any quantitative social indicator is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor*" [56]) [2].

2. On pourra m'opposer que si j'avais de brillants indicateurs j'aurais moins de pudeur à ce sujet. Je n'ai aucun moyen de désamorcer cet argument.

Organisation du document Dans un premier temps, je vais présenter les travaux réalisés pendant ma thèse et quelques années qui suivirent, sur la modélisation du temps dans les réseaux de Petri (Chapitre 1).

Le cœur de ce manuscrit sera constitué des chapitres 4 à 7 qui présenteront le cœur de ma contribution : l'analyse de performances garanties dans les réseaux temps-réel. Le chapitre 2 présentera ce que sont ces réseaux temps réel, et me permettra de définir plus précisément la nature de mon travail de recherche. Dans le chapitre 3, je ferai quelques rappels de calcul réseau, une théorie que j'utilise pour étudier certaines propriétés de ces réseaux. Ceci fait, je pourrai présenter mes travaux, en suivant trois axes. Dans le chapitre 4, "Analyser mieux", je présenterai mes premiers travaux dans le domaine, qui consistaient à améliorer la qualité des résultats issus de la théorie du calcul réseau. Dans le chapitre 5, "Analyser plus", je présenterai des extensions faites à la théorie du calcul réseau pour analyser des technologies qui n'étaient pas encore prises en compte. Dans le chapitre 6, "Augmenter la confiance", ce sont mes travaux menés autour de la question de la confiance que l'on peut avoir dans les résultats d'analyses que je présenterai (après avoir présenté en quoi cela répondait un besoin). La dernière partie de mes travaux de recherche, comment en complément du développement de méthodes d'analyse, j'ai aussi évalué des mécanismes réseau eux-mêmes (chapitre 7).

Le chapitre 9, plus court, présentera mes activités de transfert.

Le dernier chapitre présentera les perspectives que je vois à mes travaux, la direction que j'aimerais prendre par la suite.

Chapitre 1

Réseaux de Petri et temps

J'ai travaillé, pendant ma thèse et mes premières années de maître de conférence, sur les réseaux de Petri [69], et plus particulièrement leur extension "temporelle" [17] dans le contexte des systèmes multimédia.

Un réseau de Petri est un modèle de comportement adapté à la modélisation de systèmes parallèles et à gestion de ressources. Le modèle a été étendu afin de prendre en compte l'écoulement du temps dans les systèmes modélisés.

Un réseau de Petri possède :

— une structure, un ensemble de places (qui permettent de représenter des états ou des ressources), un ensemble de transitions (qui servent à représenter des actions), et des arcs orientés qui les relient,

— un état, représenté par la présence de jetons (*token*) dans les places.

Une transition est *tirable* (modélisant la réalisation d'une action) s'il existe au moins un jeton dans chaque place en amont de la transition. Le tir de la transition consomme un jeton dans chaque place en amont et en ajoute un dans chaque place en aval.

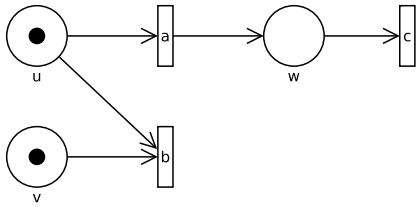


Figure 1 – *Exemple de réseau de* Petri

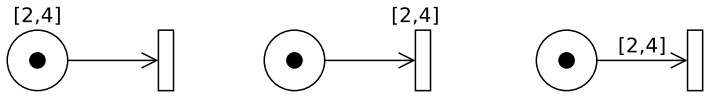


Figure 2 – *Exemple de réseau de* Petri *augmenté d'information temporelle.*

Le réseau de Petri de la Figure 1 a 3 places $\{u, v, w\}$ et trois transitions $\{a, b, c\}$. Dans l'état représenté sur la figures, où il y a un jeton dans u et un second dans v (état noté $\{u, v\}$), on peut tirer soit la transition a (ce qui mène à l'état $\{w, v\}$), puis alors la transition c (ce qui mène à l'état $\{v\}$), soit tirer la transition b (ce qui mène à l'état vide, sans jeton).

Cette sémantique crée un ordre entre les actions, mais ne permet pas de définir de notion de durée. On parle de temps *logique* (a a eut lieu avant b), par opposition à un temps *quantitatif* (une action a lieu 5mn avant une autre).

Pour modéliser le passage du temps, 3 modèles relativement proches avaient été définis : l'un ajoutait un intervalle de temps sur les places, l'autre sur les transitions, et un troisième sur les arcs amont des transitions, comme illustré sur la Figure 2. Le premier exprimait la contrainte suivant laquelle un jeton devait passer au moins 2 unités de temps dans la place, et pas plus de 4 unités de temps. Le second exprimait qu'une transition devait avoir assez de jetons pendant au moins 2 unités de temps avant de pouvoir être tirée, et pas plus de 4 unités de temps. La troisième signifiait que le jeton devait être âgé d'au moins deux unités de temps (mais pas plus de 4 unités de temps) avant de pouvoir être utilisé pour tirer la transition.

Plusieurs questions m'ont alors interpellé : ces modèles sont-ils équivalents ? Certains sont-ils plus expressifs que d'autres ? Quelles sont les généralisations raisonnables de cette sémantique, si on met plusieurs jetons dans une place ?

Il y eut deux phases dans ces travaux : dans un premier temps, j'ai pu par moi même développer et publier quelques résultats [35, 36, 31]. J'ai ensuite alors eu la chance de pouvoir travailler avec Olivier H. Roux, et avec lui généraliser ces travaux en une hiérarchie complète entre modèles [48, 49].

J'ai aussi à cette époque travaillé pour la première fois avec un doctorant, Tarek Sadani, dont je n'étais pas l'encadrant, mais avec lequel j'ai collaboré plusieurs mois sur un point de sa thèse relatif aux réseaux de Petri temporels [108, 109].

J'ai appris plusieurs choses dans cette thèse, "formation par la recherche et pour la recherche". J'ai appris que mon goût "scolaire" pour les travaux théoriques pouvait s'épanouir dans le cadre des méthodes formelles en informatique. J'y ai aussi montré mon appétence à discuter la sémantique des modèles : indépendemment des propriétés qui découlent de leur définition, des modèles peuvent-ils représenter, capturer, de manière relativement naturelle et fidèle, des propriétés des systèmes réels que l'on cherche à étudier à travers eux. J'ai aussi appris, un

peu à mes dépends [1], qu'il ne suffit pas d'avoir produit une recherche pour la voir publiée, encore faut-il le faire de manière lisible. Olivier H. Roux m'a aussi appris une règle importante : dans une preuve un peu complexe, il existe souvent un argument clé. Le mettre en exergue dans un lemme ou une remarque en dehors de la preuve rend la preuve plus lisible. J'essaye autant que possible de le faire dans mes travaux.

Entre ma thèse et mon poste de maître de conférences, j'ai passé une année de PostDoc au LIAFA, sous la direction d'Ahmed Bouajjani sur la vérification de protocole réseau à l'aide du *model-checking*. L'honnêteté me pousse à dire qu'il y a eu plus d'opportunisme dans ce poste que de construction d'une cohérence de carrière, même si je n'ai pas à rougir du travail effectué [53]. Et passer une année dans un prestigieux laboratoire parisien m'a aussi beaucoup appris sur l'organisation de la recherche en France.

1. J'avais repéré que mon résultat publié dans [35] contredisait un des résultats d'un article publié dans un prestigieuse revue [58], et je pensais que cela suffirait à m'en ouvrir les pages.

Chapitre 2

Réseaux temps réels : contexte et problématique

Je vais dans cette partie présenter les objets du monde réel qui constituent mon domaine d'étude, les réseaux temps réel, les propriétés qu'il importe d'étudier, et définir ainsi, par raffinements successifs, ma problématique de recherche.

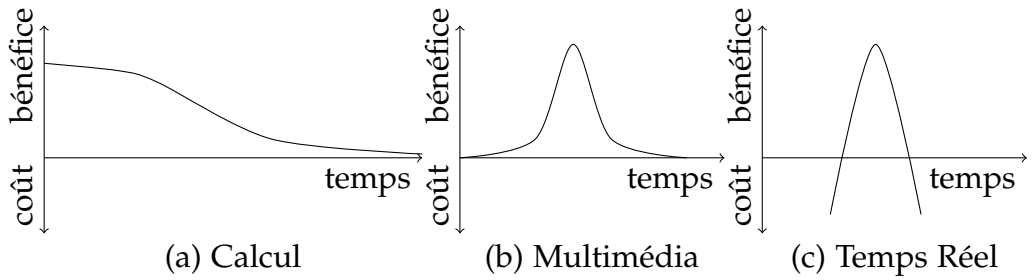
De nombreux systèmes physiques sont de nos jours pilotés, partiellement ou totalement, par des ordinateurs. Avion, automobile, satellite, machine-outil, drone, scanner médical, comme machine à laver, des éléments physiques sont contrôlés par un système incluant un calculateur et du logiciel. On qualifie ces systèmes de "cyber-physique".

2.1. L'informatique et le temps réel

Du point de vue informatique (logiciel et matériel), ces systèmes ont la spécificité d'être "temps réel", c'est-à-dire qu'une commande correcte mais appliquée au mauvais moment, trop tard comme trop tôt, aura un coût, aussi grave qu'une mauvaise commande.

Il y a là un paradigme très différent de celui de l'informatique traditionnelle, dite informatique de "calcul" où le plus tôt est toujours le mieux, et le fait de devoir attendre un résultat (l'ouverture d'un document, la réponse à une recherche Internet...) diminue l'intérêt pour le résultat, mais n'entraîne pas de coût, seulement un agacement.

Pour simplifier, sur le cas d'un véhicule, changer de direction trop tôt ou trop tard est aussi dangereux que ne pas changer de direction, et plus dangereux que



(a) Calcul (b) Multimédia (c) Temps Réel

Figure 3 – *Informatique traditionnelle, multimédia et temps réel*

de ne pas avoir démarré le véhicule.

Le terme "temps réel" est aussi utilisé dans le domaine du multimédia. Dans une bande son, une vidéo, une accélération comme un ralentissement sont une dégradation. Il existe une date nominale associée à chaque élément sonore, à chaque image, mais un décalage temporel peut rendre le bénéfice nul (comme si la vidéo ne s'était pas lancée) mais n'entraîne pas de coût.

Cette distinction entre ces trois types de temps-réel peut s'illustrer avec une fonction qui trace le bénéfice associé à la production d'un résultat en fonction de la date de sa production, comme illustré sur la figure 3. Cette modélisation entre terme de bénéfice/coût vs. temps lié à la mise à disposition d'un résultat permet aussi d'illustrer à quel point les distinctions ne peuvent être absolues : puisqu'il qu'il existe une infinité de fonctions de ce type, on pourrait définir une infinité de types de temps réel.

Notons qu'on peut souvent compenser le fait qu'un résultat soit prêt avant sa date d'utilisation en le gardant en mémoire (sous réserve d'avoir assez de mémoire), et en ne l'utilisant qu'à la date idoine.

Ma recherche se situe dans le cadre de ces systèmes temps-réels, critiques, où le fait de recevoir une donnée trop tard aurait un coût inacceptable.

Mon travail de recherche consiste à garantir qu'un résultat sera fourni avant une date limite, et qu'il y aura suffisamment de mémoire pour le stocker, particulièrement quand ce résultat doit traverser un réseau de communication.

2.2. Architecture de communication

Un réseau de communication informatique est un dispositif physique et logiciel qui permet de transporter des données entre deux entités physiquement distinctes.

Il existe de nombreuses technologies de communication : conçus pour des distances variée, les longues communications (allant du kilomètre à la traversée d'un océan), des communications courtes (à l'intérieur d'un même bâtiment, ou d'un même véhicule entre quelques décimètres à quelques centaines de mètres), voire très courtes (à l'intérieur d'une même carte informatique, ou d'une même puce) ;

utilisant des supports filaires ou sans fils ; utilisant des ondes électromagnétiques ou optiques ; offrant des débits variant du Kb/s au Tb/s ; le tout à des prix allant du centime à la dizaine de milliers d'euros par composant...

Un réseau est généralement un assemblage de composants de différentes technologies. Dans l'informatique grand public, c'est le protocole IP qui sert de "glue" entre ces technologies distinctes, protocole qui n'a pas le temps-réel dans ses principes fondateurs.

Salarié de l'ONERA, ma recherche se porte plus spécialement sur les réseaux du domaine aérospatial.

Dans un véhicule aérospatial, on va trouver de quelques dizaines à quelques milliers de composants reliés par un réseau, utilisant différentes technologies.

Dans un réseau d'avionique civile moderne, on trouvera typiquement un cœur de réseau haut-débit, avec une technologie de type AFDX [11], inter-connectant des calculateurs de puissance moyenne. Mais certains autres équipements peuvent être trop petits ou trop anciens pour une connectique AFDX et utiliser des bus de type CAN [83, 12] ou 1553 [102] par exemple. Pour des questions de fiabilité, on pourra aussi trouver des connections redondantes avec le réseau AFDX, mais utilisant une technologie différente. Cela peut donner une architecture réseau ressemblant à la figure 4.

Dans une automobile, un sous-marin, un lanceur de type Ariane, on va retrouver le même type d'architecture, avec des technologies éventuellement différentes et un nombre de composants différent.

Notons qu'il n'existe pas dans le monde des systèmes critiques de couche d'interconnexion standard entre les différentes technologies, à l'image d'IP dans l'informatique classique.

Par essence, un réseau est une ressource partagée : sa puissance de transfert sera partagée entre les différents flux de données qui y transitent. D'un point de vue temps-réel, cela signifie que le temps de traversée d'un flux, que l'on appelle *latence*, dépend des caractéristiques des autres flux de données tout autant que de ses propres caractéristiques, et surtout, de la manière dont le réseau privilégie ou non chaque flux de données.

Je peux donc préciser mon travail de recherche : il consiste à garantir que le temps de traversée du réseau par une donnée sera toujours plus petit qu'une exigence fixée, et que les éléments internes au réseau auront toujours assez de mémoire pour la stocker.

2.3. L'intégration : un processus industriel

Les deux paragraphes précédents ne parlent que de technologie, et sont une description du réseau embarqué dans un système cyber-physique. Mais un tel

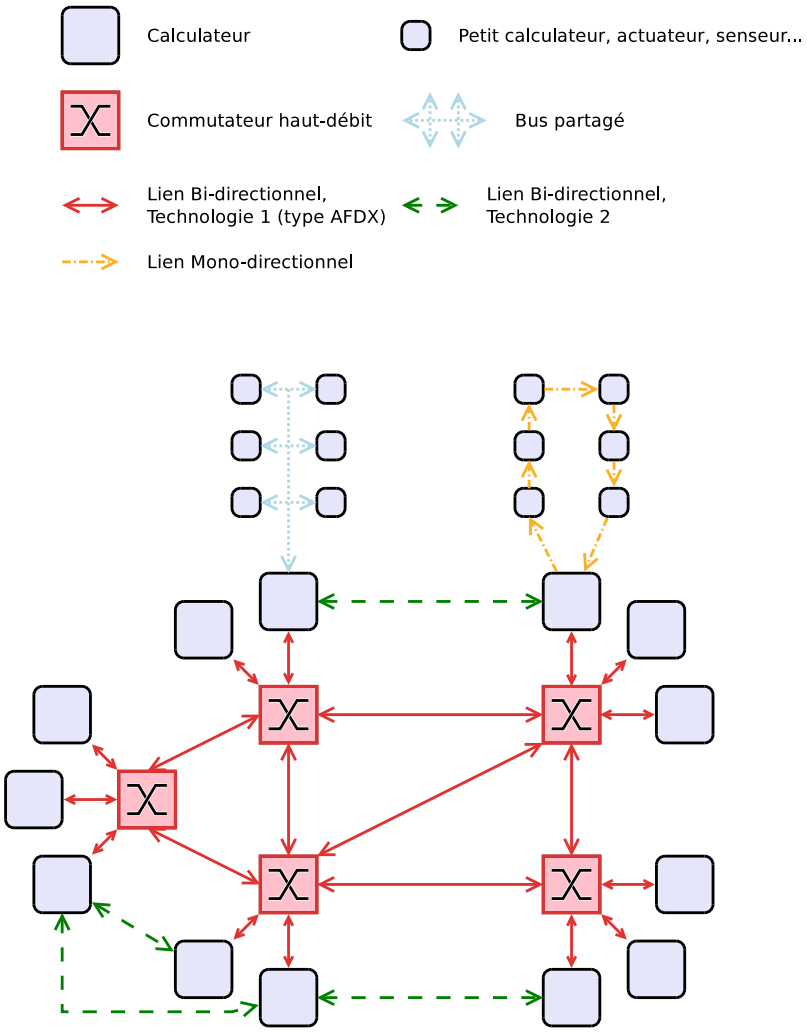


Figure 4 – *Réseau avionique comme assemblage de technologies distinctes*

système est conçu par un industriel, et l'association de sa complexité et de son exigence de fiabilité obligent à définir et maîtriser son processus de conception.

Pour donner une idée de cette complexité, on peut déjà se restreindre au cœur du réseau : sur un avion civil moderne, il va faire transiter plusieurs milliers de flots de données. Ces flots sont issus de centaines de calculateurs, développés pour partie par l'avionneur, mais surtout par des sous-traitants, voire développés en commun. Ce sont des dizaines d'équipes dont le travail influe sur les performances du réseau.

Dans le cadre d'un avion civil qui doit passer un processus de certification avant d'obtenir un agrément d'utilisation dans l'espace aérien, cela impose de plus d'être capable, lors du processus de certification, de convaincre une autorité du fonctionnement correct du système.

Pour gérer cette complexité, l'intégrateur utilise la notion d'interface : il ne spécifie pas *exactement* le comportement de chaque composant, mais un contrat d'utilisation.

Cette notion de contrat simplifie le travail :

1. seul le contrat de chaque composant est pris en compte pour vérifier la correction globale du système, et le travail de conception global peut donc se faire avant d'avoir réalisé chaque composant,
2. localement, la vérification se limite à vérifier qu'un composant respecte son contrat,
3. toute modification locale qui ne modifie pas le contrat n'a pas d'impact sur la correction du système.

Mon travail de recherche (vérification d'exigence sur les délais et les occupations mémoires), présenté à la fin du paragraphe 2.2, doit développer des solutions compatibles avec le processus industriel d'intégration.

2.4. La vérification des latences dans les réseaux

Le problème de la vérification du respect des exigences temporelles d'un réseau est, en général, un problème difficile. Les pires délais sont très rares [1] et ils ne peuvent pas être évalués grâce à la simulation (technique réseau apparentée au test dans le monde du logiciel). On préfère donc utiliser des méthodes formelles.

Prenons le temps de réfléchir à ce qu'est une méthode formelle : c'est une théorie qui permet de capturer certaines caractéristiques d'un système du monde réel (un réseau dans notre cas) dans ce que l'on appelle un *modèle*, puis de déduire des propriétés dans le cadre de la théorie, valables aussi sur le système réel.

Ceci est illustré sur la figure 5 : si on cherche à vérifier que dans un réseau donné, la latence associée à un message ne dépasse pas 3ms, il faut faire un

1. Cela sera illustré plus en détail dans l'introduction du paragraphe 4.

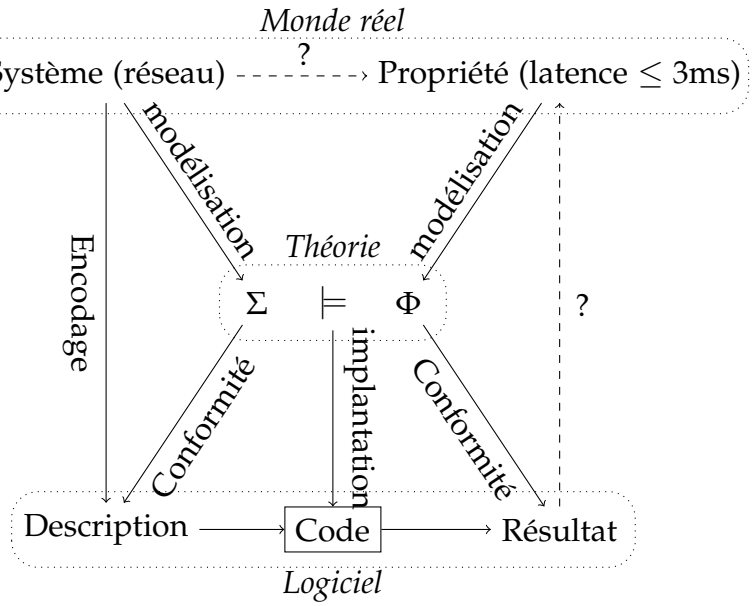


Figure 5 – *Principes de vérification à l'aide de logiciel*

modèle Σ du réseau dans une théorie, faire un encodage formel ϕ de la propriété sur la latence, puis vérifier que dans la théorie, on peut bien déduire ϕ depuis Σ.

On peut en passant remarquer que des erreurs peuvent se glisser dans la théorie mais aussi dans la modélisation (j'y reviendrai au chapitre 6).

Mais quand les systèmes sont grands, on préfère que cette déduction soit automatisable, et la faire calculer par un programme.

Dans le cas spécifique des réseaux embarqués, on associe généralement à chaque flot de donnée f une exigence sur le délais d_f^e, et on souhaite que, en notant d_m la latence du message m, on ait la propriété

$$\forall f, \forall m \in f : d_m \leq d_f^e. \tag{2.1}$$

Il faut souligner que le délai maximum, $\max_{m \in f} d_m$ est, en général, incalculable en un temps raisonnable. Plus formellement, son calcul est NP-difficile en général [24] (j'y reviendrai au chapitre 4).

Je travaille donc sur la théorie du calcul réseau [87, 59] qui, étant donné un modèle de réseau Σ calcule pour chaque flot de donnée f une borne d_f telle que tout message m de ce flot traversera le réseau en un temps $d_m \leq d_f$.

Si on a $d_f \leq d_f^e$, on a pour tout message $d_m \leq d_f^e$.

Regardons de plus près ce qui se passe si $d_f > d_f^e$, si la borne calculée est plus grande que l'exigence. Du point de vue industriel, la première chose à faire est de vérifier si l'exigence d_f^e ne peut pas être relaxée. En discutant avec les équipes utilisant le flux f, il peut arriver que l'on constate que la borne d_f est suffisante

pour les besoins réels du système. Si cela n'est pas le cas, on peut modifier à la marge le réseau pour privilégier le flux f : modifier sa priorité, son routage, et regarder si dans ce système Σ', la borne $d'_f \leq d^e_f$, et si les autres flux respectent eux aussi leurs propres contraintes dans le système modifié. Si cela échoue, il faut augmenter la puissance du réseau : au mieux ajouter un lien, au pire ajouter des éléments réseau, donc trouver une place physique dans l'avion où les placer, trouver l'alimentation électrique, etc.

L'enjeu d'un point de vue du travail de recherche est donc, tout d'abord, de savoir calculer un d_f, puis ensuite, que ce d_f soit assez proche du pire d_m, pour éviter un sur-dimensionnement du réseau, c'est à dire l'embarquement de ressources inutiles en réalité, mais dont on a besoin pour prouver le respect des contraintes.

Mais intégrer cette recherche dans un processus industriel, c'est accepter plusieurs contraintes.

Primo, comme la modélisation du système ne peut se baser que sur les contrats, la précision de l'analyse n'est proche du réel que si les contrats le sont. Notons que cela peut créer une incompréhension avec certains industriels eux mêmes : lorsqu'ils testent le système réel, les bornes d_m mesurées peuvent être très loin des bornes d_f données par la théorie. Cela peut-être dû à un pessimisme de la méthode en effet, mais cela peut aussi être lié au fait qu'il existe, théoriquement, un autre système qui respecte les contrats, et dont les bornes d_m seraient elles bien plus proches de la borne d_f (j'y reviendrai au chapitre 4).

Secundo, le développement a un coût important, surtout quand le dit logiciel doit être intégré dans un processus de certification. Dès lors, l'industriel n'est intéressé par une amélioration dans la théorie que si le gain associé compensera le coût d'un nouveau développement.

2.5. Problématique de recherche

En conclusion, mon travail de recherche consiste principalement à développer des méthodes formelles qui permettent de vérifier qu'un réseau donné délivre ses données "à temps", qu'il a toujours assez de mémoire pour stocker ces données, tout en veillant tout d'abord à ce que la théorie permette de modéliser des réseaux utilisés dans l'industrie, en respectant le processus industriel de conception du système global mais aussi à ce que l'implantation logicielle de la solution se fasse à un coût acceptable.

Chapitre 3

Calcul réseau

Cette partie fait une présentation succincte de la théorie du calcul réseau. On pourra consulter [87, 59, 20] pour plus de détails.

3.1. Modélisation d'un réseau

Le calcul réseau modélise un réseau avec deux objets : les fonctions cumulatives et les serveurs.

Definition 1 (Fonction cumulative)**.** *La* fonction cumulative *d'un flot est une fonction* $A : \mathbb{R}^+ \to \mathbb{R}^+$*, croissante, continue par morceaux, continue à gauche, et nulle en zéro. Notons* $\mathcal{C}$ *l'ensemble des fonctions cumulatives.*

La *sémantique* d'une telle fonction est que $A(t)$ représente la quantité cumulée de données envoyées par le flot jusqu'à l'instant t exclut. Si un flot envoie un unique paquet de taille 100 à l'instant 1, on aura $A(t) = 0$ pour $t \in [0, 1]$, et $A(t) = 100$ pour $t > 1$.

Definition 2 (Serveur)**.** *Un* serveur *est une relation* $\mathcal{S} \subset \mathcal{C} \times \mathcal{C}$*, totale à gauche et qui vérifie* $\forall (A, D) \in \mathcal{S} : A \geq D$*. On notera aussi* $A \xrightarrow{\mathcal{S}} D$ *pour* $(A, D) \in \mathcal{S}$*.*

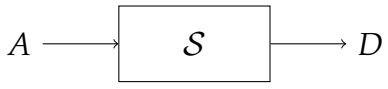


Figure 6 – *Serveur traversé par un flot*

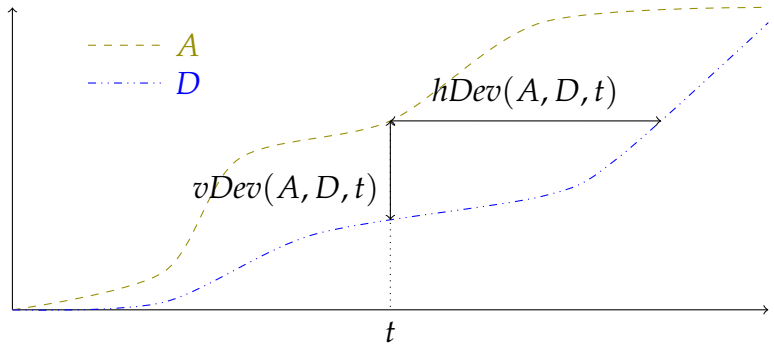


Figure 7 – *Illustration du délai et du backlog sur des courbes cumulées*

Un serveur associe à un flot en entrée A de possibles flots en sortie, et la relation $A \geq D$ traduit le fait que les données partent après avoir été reçues. Ce modèle, implicitement, ne crée ni ne perd aucune donnée : toutes les données reçues repartent.

Definition 3 (Délai et utilisation mémoire)**.** *Soit $\mathcal{S}$ un serveur. Étant donnée $A \xrightarrow{\mathcal{S}} D$, considérant un instant $t \in \mathbb{R}^+$, on définit respectivement le* délai *et le* backlog [1] *du couple (A, D) à l'instant t*

$$hDev(A, D, t) = \inf \left\{ d \in \mathbb{R}^+ \,\middle|\, A(t) \leq D(t+d) \right\}$$
$$vDev(A, D, t) = A(t) - D(t)$$

puis on généralise ces notions au couple (A, D), puis à tous les couples possibles du serveur :

$$hDev(A, D) = \sup_{t \in \mathbb{R}^+} hDev(A, D, t) \qquad vDev(A, D) = \sup_{t \in \mathbb{R}^+} vDev(A, D, t)$$
$$hDev(A, \mathcal{S}) = \sup_{(A,D) \in \mathcal{S}} hDev(A, D) \qquad vDev(A, \mathcal{S}) = \sup_{(A,D) \in \mathcal{S}} vDev(A, D)$$

Mais un élément réseau est rarement utilisé pour transporter un unique flux de données. Soit $A_1, A_2 \in \mathcal{C}$ deux fonctions cumulatives. On notera que $A_1 + A_2 \in \mathcal{C}$ et que la sémantique de $(A_1 + A_2)(t)$ consiste à compter les données du flux A_1 et du flux A_2. On dit que $A_1 + A_2$ est l'*agrégation* des deux flux.

Definition 4 (Serveur n-aire, agrégé et résiduel)**.** *Soit $n \in \mathbb{N}$. On appelle* serveur n-aire *une relation $\mathcal{S} \in \mathcal{C}^n \times \mathcal{C}^n$ totale à gauche et qui vérifie*

$$\forall (A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n), \forall i \in [1, n] : A_i \geq D_i \tag{3.1}$$

1. En français, il nous faudrait utiliser "arriéré de travail", "stock" ou "occupation mémoire"... Nous garderons dans ce document le mot anglais.

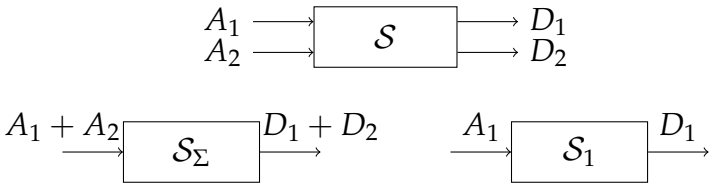


Figure 8 – *Serveur à deux flux, serveur agrégé et résiduel*

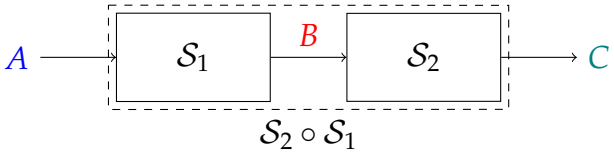


Figure 9 – *Composition de serveurs en séquence*

On définit $\mathcal{S}_\Sigma$ *le* serveur agrégé *par*

$$\forall (A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n) : \sum_{i=1}^{n} A_i \xrightarrow{\mathcal{S}_\Sigma} \sum_{i=1}^{n} D_i \tag{3.2}$$

Et pour chaque $i \in [1, n]$ *on définit le* serveur résiduel $\mathcal{S}_i$ *par*

$$\forall (A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n) : A_i \xrightarrow{\mathcal{S}_i} D_i \tag{3.3}$$

Definition 5 (Composition de serveurs)**.** *Soient* $\mathcal{S}_1, \mathcal{S}_2$ *deux serveurs. Alors* $\mathcal{S}_2 \circ \mathcal{S}_1$ *est aussi un serveur, qui a comme arrivée l'arrivée de* $\mathcal{S}_1$*, qui injecte dans* $\mathcal{S}_2$ *la sortie de* $\mathcal{S}_1$*, et dont la sortie est la sortie correspondante de* $\mathcal{S}_2$*.*

3.2. Dioid (min,plus) et (max, plus)

Le calcul réseau se base sur le dioid (min,plus). Nous ne ferons qu'une présentation rapide aussi, mais on pourra se reporter là aussi à [87, 59] pour une présentation adaptée au calcul réseau, voire [77] pour une présentation plus générale.

Commençons par introduire quelques notations :

$$a \wedge b = \min(a, b) \qquad a \vee b = \max(a, b) \qquad [a]^+ = a \vee 0 \tag{3.4}$$

Nous allons beaucoup utiliser les fonctions suivantes, illustrées sur la Figure 10.

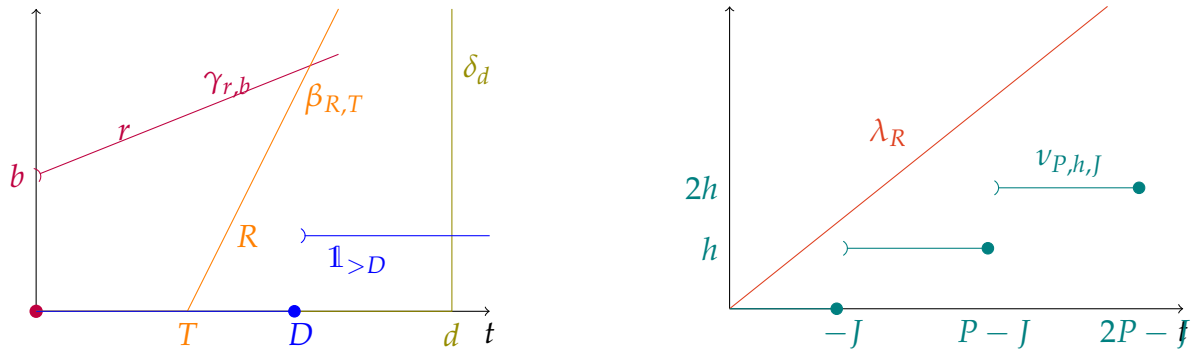


Figure 10 – *Courbes usuelles*

Definition 6 (Fonctions usuelles). *Soient $r, b, R, T, h, P \in \mathbb{R}^+$ et $d, J \in \mathbb{R}$ des paramètres, on peut définir les fonctions suivantes :*

$$\delta_d(t) = \begin{cases} 0 & si\ t \leq d \\ \infty & sinon \end{cases} \qquad \mathbb{1}_d(t) = \begin{cases} 0 & si\ t \leq d \\ 1 & sinon \end{cases}$$

$$\lambda_r(t) = rt \qquad \gamma_{r,b}(t) = (rt + b) \wedge \delta_0$$

$$\beta_{R,T}(t) = R\,[t - T]^+$$

$$\nu_{P,h,J}(t) = \left[h \left\lceil \frac{t+J}{P} \right\rceil\right]^+ \wedge \delta_0(t)$$

où $\lceil \cdot \rceil$ représente la partie entière supérieure.

Sont particulièrement utilisés les opérateurs suivants.

Definition 7 (Convolution, déconvolution, clôture sous-additive). *Soient $f, g : \mathbb{R}^+ \to \mathbb{R}$ deux fonctions. Les opérateurs de* convolution, déconvolution *et* clôture sous-additive *sont définis par*

$$(f * g)(t) = \inf_{0 \leq s \leq t} f(t-s) + g(s), \tag{3.5}$$

$$(f \oslash g)(t) = \sup_{0 \leq s} f(t+s) - g(s), \tag{3.6}$$

$$f^* = \delta_0 \wedge f \wedge (f * f) \wedge (f * f * f) \wedge \ldots \tag{3.7}$$

La convolution est associative, commutative, distributive sur le minimum, mais aussi isotone ($f \leq g \implies f * h \leq g * h$).

La déconvolution n'est ni associative ni commutative, mais elle est antitone ($f \leq g \implies h \oslash f \geq g \oslash h$).

La relation fondamentale entre convolution et déconvolution est $f * g \geq h \iff f \geq h \oslash g$.

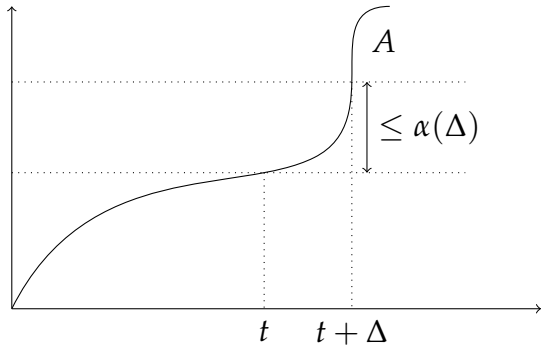


Figure 11 – *Courbe d'arrivée maximale d'une fonction cumulative*

De manière similaire, on peut travailler dans le dioïde (max, plus). Les opérateurs ont des définitions et des propriétés similaires :

Definition 8 (Max-convolution, max-déconvolution, clôture sur-additive)**.** *Soient* $f, g : \mathbb{R}^+ \to \mathbb{R}$ *deux fonctions. Les opérateurs de* max-convolution, max-déconvolution *et* clôture sur-additive *sont définis par*

$$(f \overline{*} g)(t) = \sup_{0 \leq s \leq t} f(t-s) + g(s), \tag{3.8}$$

$$(f \overline{\oslash} g)(t) = \inf_{0 \leq s} f(t+s) - g(s), \tag{3.9}$$

$$f^{\overline{*}} = 0 \vee f \vee (f \overline{*} f) \vee (f \overline{*} f \overline{*} f) \vee \dots \tag{3.10}$$

3.3. Contrats : courbes d'arrivée et de service

On ne travaille généralement pas avec les comportements réels des systèmes, trop nombreux, partiellement connus ou trop complexes, mais avec des contrats, parfois appelés enveloppes.

3.3.1. Courbes d'arrivée

Pour les flots, on utilise la notion de courbe d'arrivée[2].

Definition 9 (Courbes d'arrivée minimale et maximale)**.** *Soit A une fonction cumulative. Les fonctions* α^l, α^u *sont respectivement des fonctions d'arrivée minimale et*

2. La traduction d'"*arrival curve*" pourrait se faire par "courbe d'arrivée" (comme dans "ligne d'arrivée"), pour désigner le lieu, le passage par un point d'observation, ou "courbe des arrivées" pour signifier que ce sont les arrivées des différentes données qui sont comptées. La grammaire française oblige à trancher cette belle ambiguïté qui permet à l'anglais de porter les deux sémantiques. Nous gardons "courbe d'arrivée" dans ce document.

maximale du flux A si

$$\forall t, d \in \mathbb{R}^+ : \alpha^l(d) \leq A(t+d) - A(t) \leq \alpha^u(d). \tag{3.11}$$

Cette définition est équivalente à $A \overline{*} \alpha^l \leq A \leq A * \alpha^u$.

La courbe maximale est beaucoup plus souvent utilisée que la courbe minimale, car en réseau, on spécifie souvent un contrat maximal de trafic et beaucoup moins souvent un trafic minimal. On utilise donc souvent le terme "courbe d'arrivée" pour désigner la courbe maximale, et la notation α.

Les courbes d'arrivées ont de nombreuses propriétés, dont seules les principales sont données par la suite.

Proposition 1 (Propriétés des courbes d'arrivée)**.** *Soit A une fonction cumulative, et α^l, α^u un couple de courbes d'arrivée minimale et maximale. On a alors*

- *Si $\dot{\alpha}^l, \dot{\alpha}^u$ est un second couple de courbes minimale et maximale pour A, alors $(\dot{\alpha}^l \vee \alpha^l), (\dot{\alpha}^u \wedge \alpha^u)$ est aussi un couple de courbes minimale et maximale pour A.*
- *Le couple $A \overline{\oslash} A, A \oslash A$ est un couple de courbes d'arrivée minimale et maximale, et c'est le meilleur, c'est à dire que $\alpha^l \leq A \overline{\oslash} A$ et $A \oslash A \leq \alpha^u$.*
- *Si $\dot{A}$ est une fonction cumulative ayant $\dot{\alpha}^l, \dot{\alpha}^u$ pour couple de courbes d'arrivée minimale et maximale, alors $(\dot{\alpha}^l + \alpha^l), (\dot{\alpha}^u + \alpha^u)$ est un couple de courbes minimale et maximale pour $\dot{A} + A$.*

3.3.2. Courbes de service

Alors qu'il existe deux notions de courbes d'arrivée pour préciser le contrat sur un flot de donnée, il existe de nombreuses notions pour capturer le service offert par un serveur. Nous présenterons les quatre les plus populaires, mais on pourra se référer à [23, 25, 19] pour une liste plus complète, ainsi que leurs relations.

Definition 10 (Service minimal et maximal)**.** *Soit $\mathcal{S}$ un serveur. On dit qu'il offre un* service minimal *de courbe β et un* service maximal *de courbe β^M si pour tout couple $(A, D) \in \mathcal{S}$, on a la relation*

$$A * \beta^M \geq D \geq A * \beta. \tag{3.12}$$

Definition 11 (Période de backlog)**.** *Soient $A, D \in \mathcal{C}$ tels que $A \geq D$. Soit $I \subset \mathbb{R}^+$ un intervalle. On dit que I est* une période de backlog *si*

$$t \in I \implies A(t) > D(t). \tag{3.13}$$

Definition 12 (Service strict)**.** *Soit $\mathcal{S}$ un serveur. On dit qu'il offre un* service strict *de courbe β si*

$$\forall (s,t] \text{ période de backlog}, D(t) - D(s) \geq \beta(t-s). \tag{3.14}$$

Notons que si un serveur offre un service strict de courbe β, il offre aussi un service minimal de courbe β.

Soulignons aussi que les dénominations varient : ce qui est présenté ici comme service minimal est aussi appelé service simple, service minimal simple, service minimal min-plus, et le service strict est aussi parfois appelé service minimal strict. Ces dénominations ont une importance, mais dans le cadre de ce document qui est la présentation d'une démarche et non une contribution technique, nous avons pris les dénominations les plus simples.

Definition 13 (Shaper)**.** *Soit $\mathcal{S}$ un serveur. On dit que c'est un* shaper *de courbe σ si pour tout $A \stackrel{\mathcal{S}}{\rightarrow} D$, la sortie D admet σ pour courbe d'arrivée.*

3.4. Principaux résultats de calcul réseau

Theorem 1 (Flot traversant un serveur)**.** *Soit $\mathcal{S}$ un serveur offrant un service minimal de courbe β, un service maximal de courbe β^M et étant un shaper de courbe σ. Soient une courbe de flots d'arrivée et de départ $A \stackrel{\mathcal{S}}{\rightarrow} D$, et α^l, α^u une couple de courbes d'arrivée minimale et maximale pour A. Alors on a des bornes sur le délai et le backlog*

$$hDev(A, D) \leq hDev(\alpha, \beta), \qquad vDev(A, D) \leq vDev(\alpha, \beta), \tag{3.15}$$

et on peut aussi calculer un couple de courbes d'arrivée minimale et maximale pour le flot en sortie D

$$\dot{\alpha}^u = \left(\left(\alpha^u * \beta^M\right) \oslash \beta\right) \wedge \sigma, \tag{3.16}$$

$$\dot{\alpha}^l = \alpha^\ell * \left(\beta \overline{\oslash} \beta^M\right). \tag{3.17}$$

Ce premier résultat est fondamental puisqu'il permet de borner le délai et la quantité de mémoire associés à un flux sur un serveur, et à propager le calcul sur le serveur suivant.

Theorem 2 (Pay burst only once)**.** *Soit $\mathcal{S}_1, \mathcal{S}_2$ deux serveurs et $\beta_1, \beta_1^M, \beta_2, \beta_2^M$ tels $\mathcal{S}_1$ offre un service minimal de courbe β_1 et maximal de courbe β_1^M, et réciproquement pour β_2, β_2^M et $\mathcal{S}_2$.*

Alors, le serveur $\mathcal{S}_2 \circ \mathcal{S}_1$ offre un service minimal de courbe $\beta_1 * \beta_2$ et un service maximal de courbe $\beta_1^M * \beta_2^M$.

Ce second résultat est le plus connu du calcul réseau. Il est intéressant car il donne des bornes sur les délais plus petites que la somme des délais locaux (c-à-d que $hDev(\alpha, \beta_1 * \beta_2) \leq hDev(\alpha, \beta_1) + hDev(\alpha \oslash \beta_1, \beta_2)$). Il est appelé "pay burst only once" car dans un modèle où la courbe d'arrivée est constituée d'une

rafale et un débit (courbe $\gamma_{r,b}$), le terme de la rafale apparaît deux fois si on fait la somme des délais locaux, et une seule fois si on applique ce résultat.

Ces deux résultats permettent de calculer les performances associées à un flot traversant un ou plusieurs serveurs. Quand le serveur est partagé entre plusieurs flots, il faut calculer un service résiduel.

Theorem 3 (Multiplexage quelconque)**.** *Soit $\mathcal{S}$ un serveur n-aire, dont le serveur agrégé $\mathcal{S}_\Sigma$ offre le service strict β. Soient $(A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n)$, et soit $i \in [1, n]$. Supposons que chaque $j \neq i$, A_j admet pour courbe maximale d'arrivée α_j, alors, le serveur résiduel $\mathcal{S}_i$ offre un service minimal de courbe*

$$\left[\beta - \sum_{j \neq i} \alpha_j\right]^+_\uparrow . \tag{3.18}$$

Ce résultat s'applique quel que soit la politique de service du serveur. Il est donc appelé "ordonnanceur aveugle" (*blind multiplexing*) ou "ordonnancement arbitraire".

Si la politique de service est connue, de meilleurs résultats peuvent être calculés.

Cette rapide introduction à la théorie du calcul réseau, qui à quelques détails près présente des résultats déjà existants en 2008, lorsque j'ai commencé à travailler sur le calcul réseau.

Chapitre 4

Analyser mieux

Le calcul réseau calcule des bornes sur les délais. Ces bornes peuvent être assez éloignées des mesures réelles. Essayons de voir d'où viennent ces différences, à l'aide de la figure 12.

Le système que l'on cherche à analyser, le réseau AFDX d'un avion donné par exemple, possède un ensemble de comportements possibles. Dans chaque comportement, des trames du flux d'intérêt vont être envoyées, chacune va subir un certain délai, et on peut considérer le délai maximum associé à ce comportement. En considérant l'ensemble (potentiellement infini) de comportements possibles, on peut définir le pire cas du système réel (en bleu sur la figure).

En testant (sur un banc d'essai ou en vol) le système réel, on peut observer un sous-ensemble de ces comportements réels, et en retirer une pire mesure (en gris sur la figure).

Mais un système industriel étant un assemblage de composants répondant à des cahiers de charges, des contrats (comme évoqué au paragraphe 2.3), d'autres systèmes respectant les contrats pourraient avoir des comportements différents. Par exemple, considérons la jigue associée à un flux de données AFDX (*Virtual Link*) en sortie d'un émetteur (*End-System*). La norme AFDX [11] impose une borne supérieure à cette jigue. Considérons deux *End-Systems*, l'un ayant un délai quasiment constant et l'autre un délai très variable, mais toujours inférieur à la borne. Tous deux respectent le contrat, tout en ayant des comportements différents.

Sur la figure, cela est illustré par le fait que l'ensemble des comportements réels (en bleu) est inclus dans l'ensemble des comportements respectant les contrats (en mauve).

Puisque l'on veut pouvoir garantir le bon fonctionnement du système complet,

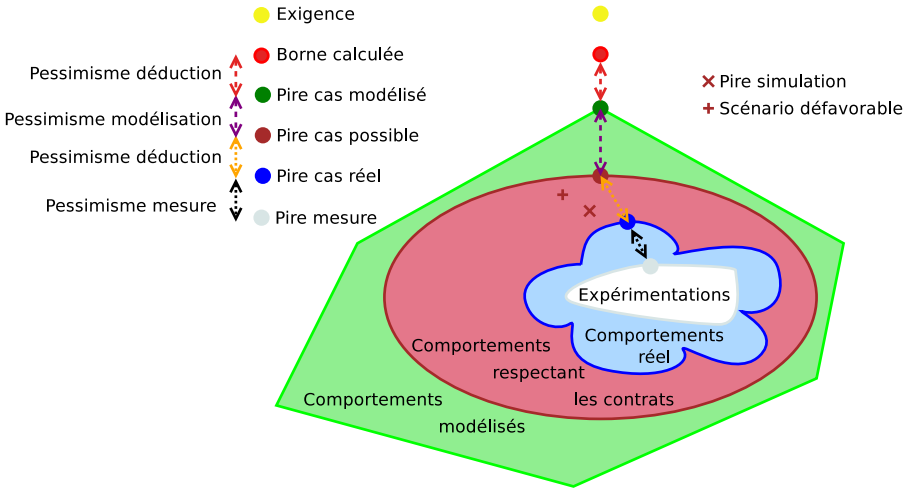


Figure 12 – *Sources de pessimisme dans une analyse formelle*

même si on remplaçait un équipement par un autre, pourvu que tous deux respectent le même contrat, il faut vérifier non pas que le pire cas *réel* soit inférieur à la borne, mais le pire cas *possible* (en mauve sur la figure).

Comme ce pire cas possible est difficile à déterminer, on a recours à des méthodes formelles (cf. paragraphe 2.4). Il va falloir faire un modèle des contrats dans la théorie : c'est l'activité de modélisation. Un tel modèle représente un ensemble de comportements (en vert sur la figure), et admet aussi une valeur maximale, le pire cas *modélisé*.

Dernier point, il reste à calculer de manière effective une borne supérieure à ce pire cas (en rouge sur la figure).

Rajoutons deux autres valeurs spécifiques.

Observer des comportements réels demande un gros effort d'expérimentation. On préfère souvent simuler le système. Un simulateur va, sauf cas particuliers, se baser sur les contrats des systèmes, et explorer ainsi un nombre fini de comportements possibles. La plus grande valeur obtenue par simulation est dite "pire simulation" (une croix mauve sur la figure).

Pour finir, on peut aussi construire (de manière automatique) un scénario défavorable, construit de manière à faire expérimenter des grands délais aux flux de données. La valeur du scénario défavorable est représentée par une croix mauve sur la figure.

Il n'existe pas de lien établi entre pire cas réel, pire simulation et scénario défavorable. Tout dépend du talent de celui qui construit le simulateur ou le scénario défavorable.

Lorsque j'ai commencé à travailler sur la théorie du calcul réseau, cette théorie avait la réputation d'être très pessimiste. Elle avait été appliquée avec succès sur le calcul de bornes du réseau AFDX embarqué dans l'A380 [79, 80, 72]. Mais Airbus

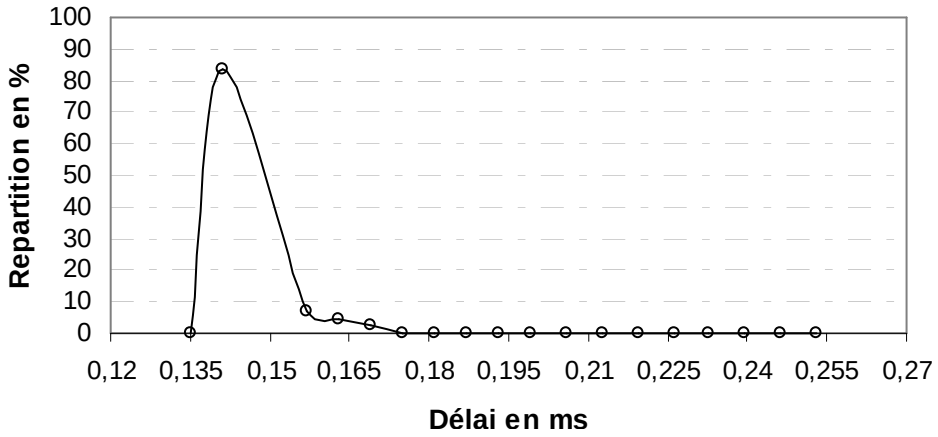


Figure 13 – *Répartition des délais pour un VL dans une configuration représentative d'une configuration avionique, [60, Figure 71]*

commençait à mesurer des délais sur le réseau AFDX de l'A380, et les bruits de couloir parlaient de mesures 10 fois inférieures aux bornes (sur la figure, cela correspond à la distance entre les points gris et rouge).

Une thèse a été menée à cette époque sur la simulation de réseau AFDX [60, 61]. Elle a permis de montrer que les délais moyens étaient 2 à 3 fois plus petits que les plus grands délais mesurés, mais elle suggérait aussi que le pessimisme du calcul réseau pouvait être d'un facteur 10, voire 100 (en mesurant cette fois la distance entre la croix mauve et le point rouge).

Comme le pire cas possible n'est pas calculable de manière générale, on ne pouvait pas savoir si ce pessimisme était réellement dû à la méthode du calcul réseau (distance entre le pire cas possible – point mauve – et la borne – point rouge) ou si elle venait du fait que le pire cas étant très rare, il était difficile de l'observer (distance entre la pire simulation – croix mauve – et le pire cas possible – point mauve – ou distance entre la pire expérimentation – point gris – et le pire cas possible).

Notons deux points avant de présenter davantage mes travaux.

Primo, il a été montré [24] que l'égalité entre le pire cas modélisé et la borne calculée en général, est un problème NP-difficile. Lorsqu'il y a égalité, on dit que la méthode est *tight*.

Le fait que le problème soit, en général, NP-difficile ne signifie pas qu'il soit insoluble en pratique : d'une part, il peut exister des types de réseaux pour lesquels l'égalité est atteignable, à condition d'utiliser des algorithmes spécifiques, et d'autre part, il peut aussi exister des algorithmes qui, sans donner le véritable pire cas, donnent des valeurs "assez proches".

Une partie de mes travaux s'est donc portée sur la réduction de ce pessimisme : dans le cas de la politique FIFO (paragraphe 4.1), dans le cas de la politique NP-SP (paragraphe 4.2), dans la modélisation des flux sporadiques (paragraphe 4.3), puis

sur l'évaluation de ce pessimisme (paragraphe 4.4).

4.1. Politique FIFO

Mes premiers travaux en calcul réseau cherchaient donc à calculer des bornes plus petites dans les systèmes utilisant une politique FIFO.

4.1.1. Etat de l'art (de 2008)

Introduisons plus précisément le problème, tel qu'il existait vers 2008.

Pour calculer le délai d'un flux traversant un serveur appliquant une politique FIFO, il existait deux résultats.

Proposition 2 (Service résiduel FIFO)**.** *Soit $\mathcal{S}$ un serveur offrant un service minimal de courbe β, traversé par un ensemble de flux $A_1, \ldots, A_n$ de courbes d'arrivée respectives $\alpha_1, \ldots, \alpha_n$. Soit $i \in [1, n]$, alors le serveur résiduel $\mathcal{S}_i$ offre les services minimaux suivant*

$$\delta_{d_{\mathcal{S}}} \text{ avec } d_{\mathcal{S}} = hDev\left(\sum_{i=1}^{n} \alpha_i, \beta\right), \tag{4.1}$$

$$\forall \theta \in \mathbb{R}^+ : \beta_i^\theta = \left[\beta - \sum_{j \neq i} \alpha_j\right]^+ \wedge \delta_\theta. \tag{4.2}$$

L'équation (4.1) traduit simplement que, avec une politique FIFO, le délai du flux agrégé se reporte sur chaque flux individuel. Avec ce résultat, le calcul du délai de bout en bout est la simple somme des délais locaux, et l'usage du "Pay Burst Only Once" (Thm. 2) ne peut pas améliorer ce résultat. C'est cette équation qui était utilisée dans [79].

L'équation (4.2) calcule elle, un service résiduel par flux, et permet d'utiliser le Thm. 2 pour calculer un résultat de bout en bout plus précis. Mais cela suppose de savoir choisir, pour chaque serveur $\mathcal{S}_i$ traversé, un bon paramètre θ_i. Considérons la configuration avec 3 flux et 3 serveurs de la figure 14, et intéressons nous au flux F_2. On peut calculer des courbes de service résiduel par serveur $\beta_{1|F_2}^{\theta_1} = \left[\beta_1 - \alpha_1 * \delta_{\theta_1}\right]^+ \wedge \delta_{\theta_1}$, $\beta_{2|F_2}^{\theta_2} = \left[\beta_2 - \alpha_1' * \delta_{\theta_2}\right]^+ \wedge \delta_{\theta_2}$, $\beta_{3|F_2}^{\theta_3} = \left[\beta_3 - \alpha_3 * \delta_{\theta_3}\right]^+ \wedge \delta_{\theta_3}$, ce qui suppose de savoir calculer la courbe d'arrivée α_1' du flux F_1 en sortie du serveur $\mathcal{S}_1$, ce qui peut se faire en utilisant aussi l'équation (4.2), et en introduisant un autre θ.

On peut aussi utiliser le théorème 2 et considérer la séquence de serveurs $\mathcal{S}_1, \mathcal{S}_2$ comme un unique serveur de politique FIFO et de service $\beta_1 * \beta_2$, et calculer la courbe $\beta_{1,2|F_2}^{\theta_1} = \left[(\beta_1 * \beta_2) - \alpha_1 * \delta_{\theta_1}\right]^+ \wedge \delta_{\theta_1}$.

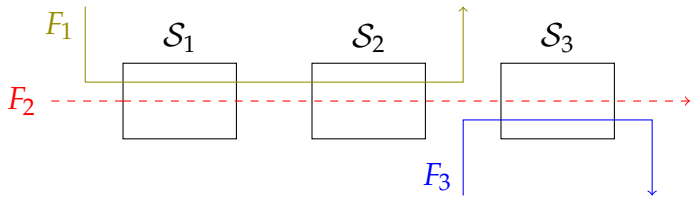


Figure 14 – *Séquence de serveurs partagés*

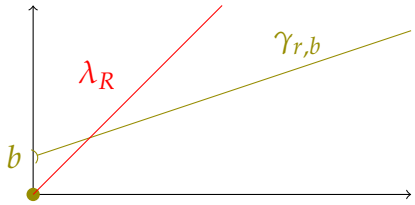


Figure 15 – *Groupage et courbe d'arrivée de type seau percé*

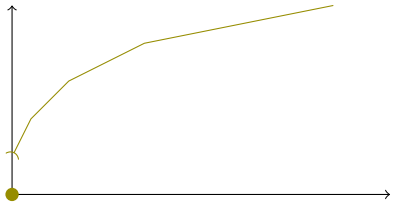


Figure 16 – *Fonction linéaire par morceau concave*

Suivant les cas, le délai de bout en bout sera soit $hDev\left(\alpha_2, \beta^{\theta_1}_{1|F_2} * \beta^{\theta_2}_{2|F_2} * \beta^{\theta_3}_{3|F_2}\right)$ ou $hDev\left(\alpha_2, \beta^{\theta_1}_{1,2|F_2} * \beta^{\theta_3}_{3|F_2}\right)$, c'est à dire une expression $d(\theta_1, \ldots, \theta_n)$ dont la valeur dépendra du choix des paramètres θ_i.

Au début des années 2000, des travaux sur la recherche de la valeur optimale du délai $d(\theta_1, \ldots, \theta_n)$ ont été menés à l'université de Pise, [90, 89, 88], et implantés dans l'outil DEBORAH [18].

Ces travaux faisaient l'hypothèse que les courbes de service étaient de la forme débit-latence $\beta_{R,T}$, et les courbes d'arrivée des seaux percés $\gamma_{r,b}$.

4.1.2. Groupage

Autant l'hypothèse de service était réaliste dans un réseau AFDX (où, après un délai de commutation T, le serveur offre un débit continu R), autant la courbe d'arrivée pouvait être améliorée, par la prise en compte du *groupage*. Cette notion a été nommée et mise en lumière dans [80, 79].

Le principe est le suivant : quand un flux traverse un élément réseau, par exemple un port à 100Mb/s, quelle que soit la courbe cumulée en entrée, la sortie ne pourra jamais être supérieure au débit maximal de l'élément réseau. En terme de courbe d'arrivée, c'est le terme σ de l'équation (3.16) qui représente cette contrainte.

En anglais, cette même notion est appelée *grouping* (traduction directe), *shaping* (pour faire le lien avec la notion de *shaper* définie en calcul réseau) ou *serialization* (pour souligner que plusieurs trames venant d'une même source ne peuvent pas être reçues toutes en même temps, mais sont reçues les unes après les autres, en série). Ainsi, un flux de type seaux percés (de courbe d'arrivée $\gamma_{r,b}$) passant par

un lien de débit R aura pour courbe d'arrivée $\lambda_R \wedge \gamma_{r,b}$ (cf. figure 15). Cette courbe plus petite donne des bornes plus petites. Cette notion de groupage a été depuis présentée de manière très détaillée et pédagogique dans [118].

4.1.3. Contributions

Mes premiers travaux en calcul réseau ont donc été de contribuer à l'étude de la politique FIFO en calcul réseau en prenant en compte le groupage.

Dans [40], j'ai montré qu'il ne suffit pas de considérer des courbes à deux pentes, comme sur la figure 15, car lorsque l'on considère un élément réseau avec plusieurs entrées, il faut considérer la somme de fonctions à deux pentes, ce qui amène dans la classe plus générale des fonctions concaves linéaires par morceaux (cf. figure 16). Après avoir donné quelques propriétés algorithmiques de ces courbes, j'y ai donné une borne analytique d'une expression

$$\inf_{\theta \geq 0} \alpha \oslash \beta^{\theta} \tag{4.3}$$

avec $\beta^{\theta} = [\beta_{R,T} - \gamma_{r,b}]^{+} \wedge \delta_{\theta}$, et α une courbe linéaire par morceaux concave. C'est à dire

- le calcul, local, d'une courbe d'arrivée en sortie d'un serveur FIFO en considérant toutes les valeurs θ possibles dans l'équation (4.2),
- en modélisant le groupage dans le flux considéré,
- mais avec juste des seaux percés pour le flux incident.

Ensuite, dans [32], je me suis intéressé cette fois

- au délai de bout en bout d'une topologie en séquence, comme celle de la figure 14,
- en modélisant le groupage dans le flux considéré,
- mais avec juste des seaux percés pour le flux incident.

Il s'agit d'une généralisation (partielle) des travaux de l'université de Pise [90, 89, 88] qui ne modélisait pas le groupage (mais considérait des topologies plus complexes).

4.1.4. Un outil : NC-maude

Pour réaliser les expérimentations j'ai développé un outil, NC-maude [33, 34]. Il existait à l'époque quelques outils, dont les principaux sont :

- DISCO, un outil d'analyse de calcul réseau en Java, libre [111, 76],
- un outil développé par Fabrice Frances et Jérome Grieu, en Java, non public mais auquel je pouvais avoir accès,

— la RTC Toolbox, avec une partie en Matlab, publique, et un cœur de calcul en Java, dont seul le binaire était diffusé [117].

Plutôt que de me lancer dans la reprise de milliers de lignes de code Java, et en considérant que mon problème était simple, je décidais de commencer le développement d'un nouvel outil, en utilisant un langage particulier, maude [100, 63]. Ce langage est un langage de ré-écriture, qui possède des qualités importantes pour faire du calcul réseau. Entre autres, il a une syntaxe très proche des mathématiques, et il permet des calculs symboliques. On pourra trouver dans [33] une présentation plus détaillée des différents outils et mon analyse des qualités de maude pour modéliser le calcul réseau.

Néanmoins, j'ai abandonné l'implantation de NC-maude en 2014. Plusieurs raisons à cela. Pour commencer, maude est un langage de recherche, et il manque des commodités des langages modernes : test unitaire, debug. Ensuite, sa syntaxe très riche, très permissive, rend difficile la recherche d'erreur de syntaxe. Mais surtout, j'ai commencé à collaborer en 2009 avec la société RTaW, qui commence alors à développer son propre outil de calcul réseau.

Une possibilité était de continuer à avoir un prototype académique, qui servirait de bac à sable pour la recherche, et un outil industriel, qui intégrerait les avancées. RTaW a proposé un montage différent : l'ouverture du cœur de calcul min-plus par l'interface d'un interpréteur en ligne de commande, et un développement de leur outil en deux branches, une branche académique, dans laquelle pouvait être testée des idées nouvelles, et une branche industrielle, stable, dans laquelle ne sont maintenues que les mécanismes ayant reçus un intérêt industriel. C'est cette deuxième solution qui a été retenue : dans le cadre d'un accord cadre, j'ai un accès privilégié a des parties privées de leur outil, et nous testons régulièrement de nouvelles idées, dont les preuves et algorithmes sont publiés, mais dont leur code reste leur propriété.

Ce travail initial sur NC-maude a été extrêmement formateur et m'a permis d'appréhender les enjeux liés à l'analyse automatique d'un réseau temps réel à l'aide de la théorie du calcul réseau.

4.1.5. Conclusion

Mes travaux sur la politique FIFO, qui généralisaient partiellement les résultats de l'université de Pise [18] n'ont jamais été appliqués sur un exemple industriel,

— pour une question théorique, parce qu'ils ne modélisaient le groupage du flux d'intérêt et pas sur les flux concurrents, et que la généralisation me semblait trop complexe,

— pour une question d'implantation, car il restait des bugs dans l'outil NC-maude, et qu'il mettait quasiment une semaine pour évaluer une configuration industrielle.

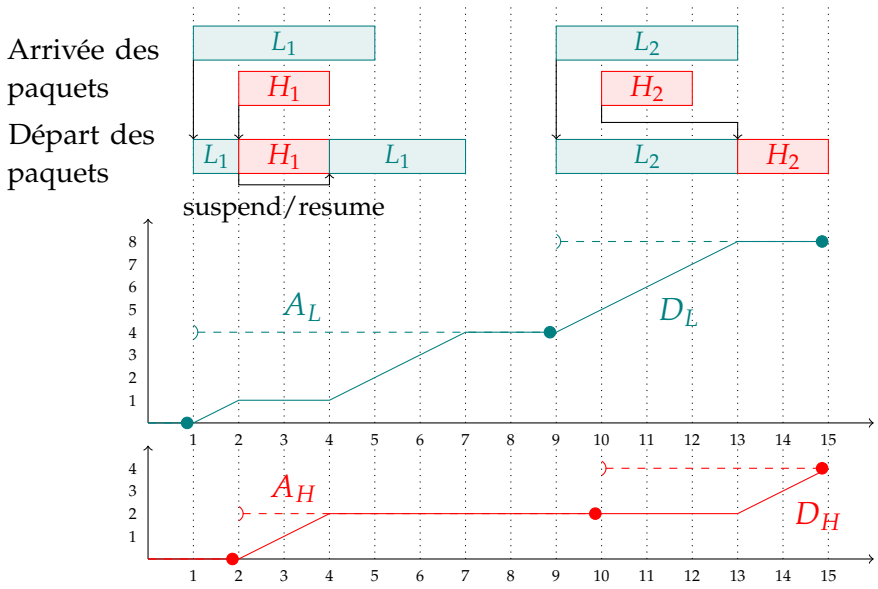


Figure 17 – *Ordonnancement à priorité statique : version préemptive et non-préemptive*

Une solution a été apportée par Anne Bouillard et Giovanni *Stea* en 2012, en changeant complètement d'approche [26, 27]. Au lieu d'utiliser les résultats de la proposition 2, elle encode la sémantique du calcul réseau comme un problème linéaire mixte à variables entières. Cela généralise directement les travaux de l'université de Pise comme les miens, en prenant en compte le groupage sur tous les flux.

4.2. Politique "priorité statique non préemptive" et bus CAN

Une autre politique importante en ordonnancement est la politique à priorité statique (SP, *static priority*), et plus particulièrement la politique à priorité statique non préemptive (NP-SP, *non-preemptive static priority*).

Considérons deux flux de données, l'un de priorité haute, H et l'autre de priorité faible L (*low*). Considérons un serveur à débit constant. Si le serveur est libre et qu'une requête de priorité faible arrive à un instant ($t = 1$ ou $t = 9$ sur la figure 17), le serveur commence à servir le flux de priorité basse. Si une requête de priorité haute arrive une unité de temps plus tard, en cas d'ordonnancement préemptif (cas $t = 2$), le serveur cesse de servir le flux de priorité faible et sert celui de priorité haute, alors qu'en cas d'ordonnancement non-préemptif, la requête de priorité faible est servie complètement avant de servir le flux de haute priorité.

Considérons le cas du flux de priorité faible : en calcul réseau, il reçoit un

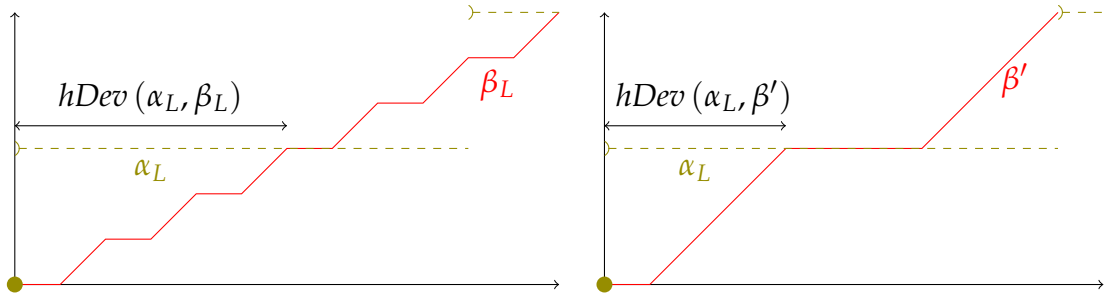


Figure 18 – *Service résiduel de priorité faible :* $\beta = \lambda_1$, $\alpha_H = \nu_{2,1}$, $\alpha_L = \nu_{10,3}$, $\beta_L = [\beta - \alpha_H]^+_\uparrow$,

service résiduel de courbe $[\beta - \alpha_H]^+_\uparrow$ si β représente la courbe du service strict du serveur, et α_H la courbe d'arrivée du flux de haute priorité. Ceci que la politique soit préemptive ou non.

Mais ce résultat ne prend pas en compte le fait que lorsqu'une trame de priorité faible est servie, elle accapare toute la capacité du serveur. Illustrons cela sur un exemple : supposons que le flux de haute priorité envoie au plus toutes les 2 unités de temps une trame de taille 1, alors que le flux de plus faible priorité envoie une trame de taille 3 toutes les 10 unités de temps, et que le serveur offre un débit λ_1.

Du point du vue du flux de plus faible priorité, le pire cas d'attente consiste à arriver en même temps qu'une trame prioritaire, à attendre une unité de temps qu'elle soit servie, puis il occupe le serveur pendant 3 unités de temps, soit un délai de 4.

Si on utilise la fonction de service résiduel, $[\beta - \alpha_H]^+_\uparrow$ on obtient un délai de 5 unités de temps (cf. Figure 18).

Se posait la question de savoir si on pouvait construire (c'est à dire trouver une expression et une preuve) une fonction β' dont la déviation horizontale donnerait le délai recherché de 4 unités de temps.

Ce fut l'amorce du sujet de thèse de William Mangoua Sofack, et il réussit en effet à construire une telle fonction : dans [97, 98], une telle expression est construite, avec l'hypothèse que tous les paquets d'un flux de données ont la même taille. Nous n'avons pas de preuve qu'elle calcule le pire cas, mais William Mangoua Sofack l'a comparé avec [68] qui calcule les pires cas dans le cas d'un bus CAN, sur 10.000 cas d'études. Dans tous les cas, la nouvelle méthode calcule exactement le pire cas.

L'objectif a été partiellement atteint : le service résiduel a été amélioré, il semble aussi bon que les méthodes existantes dans le cas du bus CAN, mais son expression est vraiment très complexe, il ne traite que des paquets de taille fixe, et il n'y a pas de résultat d'exactitude.

Je ne sais pas dire, à ce jour, si ces contraintes pourraient être levées.

4.3. Flux sporadiques et fonctions en escalier

Les fonctions linéaires par morceaux concaves (cf. figure 16) permettent de modéliser le groupage associé aux flux contraints par un seau percé. Mais d'où viennent ces seaux percés ? Ils peuvent être utilisés pour modéliser un flux sporadique, mais un modèle plus précis peut être utilisé.

Un flux *sporadique* de pseudo-période P envoie des requêtes en séquence, avec une distance minimale P entre deux requêtes. Si une requête a pour taille maximale L, la courbe d'arrivée maximale d'un tel flux est $\nu_{P,L,0}$ (cf. definition 6 et figure 10). À ce contrat peut s'ajouter une jigue J, une variation autour de la date théorique d'arrivée, ce qui donne une courbe d'arrivée $\nu_{P,L,J}$.

Un flux de données AFDX est un flux sporadique, dont la pseudo-période est nommée *Bandwidth Allocation Gap* (BAG) et la taille maximale $L^{\max}$. Sa jigue doit être statiquement bornée à l'émission, et évolue au fur et à mesure des traversées de commutateurs.

Une telle courbe d'arrivée admet une sur-approximation linéaire $\gamma_{r,b}$, avec $r = \frac{L}{P}$ et $b = rJ + L$. Si on s'intéresse à la rafale maximale associée à ce flux, qui dans ce cas correspond à la limite à droite de la valeur à l'origine, on voit que le modèle linéaire est "un peu trop grand", d'une valeur $\frac{LJ}{P}$ (cf. figure 19.a).

En pratique, l'usage des fonctions en escalier donne de meilleurs résultats que les approximations linéaires (une comparaison sera donnée au paragraphe 4.4). Mais les fonctions en escalier sont moins pratiques à manipuler, en terme d'implantation. En particulier, la somme de fonctions en escaliers est bien moins régulière qu'une fonction linéaire par morceaux concave (cf. figure 16).

L'idée que j'ai développée dans [44] consiste en fait à construire une courbe à deux pentes, comme sur la figure 19.b, qui soit une sur-approximation de la courbe en escalier. L'idée de l'algorithme est alors, sur chaque port, lorsqu'il s'agit d'agréger les flux de données, et donc de faire les sommes de courbes d'arrivée, de calculer les sur-approximations, et de faire les sommes de ces sur-approximations pour calculer le délai, mais ensuite, de propager ce délai sur les courbes en escaliers elles mêmes.

Sur un cas d'étude fourni pas Thales Avionics, avec une centaine de calculateurs, 8 commutateurs et 6500 chemins de données, nous avons utilisé les courbes linéaires par morceaux concaves (LMC), notre proposition, et une classe générale (UPP, [29]) qui permettait de capturer les courbes en escalier.

Les temps de calcul respectifs étaient de 0,9s, 1,1s et 7,5s. La nouvelle méthode était plus précise que les LMC, et bien plus rapide que les UPP.

Ici aussi, la vérité pleine est que l'implantation des courbes en escalier avait un bug algorithmique, et que lorsque cette étude a démarré, le travail avec les courbes en escalier était environ 100 fois plus lent que celui avec les courbes linéaires par morceaux concaves, LMC. L'implantation des UPP a été améliorée, pour n'être plus que 8 fois plus lente que les LMC dans le cas de flux harmoniques (où il

Figure 19 – *Modèles en escalier et linéaire d'un flux sporadique avec jigue*

existe une période de base, et où toutes les périodes P sont de la forme $P = 2^k P_0$) et c'est désormais elle qui est la plus utilisée.

4.4. Évaluation du pessimisme

Comme présenté en introduction du paragraphe 4, le calcul du pire cas est hors de portée en général, et beaucoup se posaient la question du pessimisme induit par le calcul réseau dans l'évaluation de bornes.

C'est alors qu'Henri Bauer a proposé, dans [16], une méthode pour construire des scénarios défavorables dans le cas de l'AFDX (en fait, dans le cas des réseaux à politique FIFO et SP). Si on note $A_1, \ldots, A_n$ les trafics générés par n VL en entrée du réseau AFDX pour un comportement donné, et $d_i(A_1, \ldots, A_n)$ le délai associé au VL d'indice i, sa méthode permet de calculer un ensemble de scénario $A_1^i, \ldots, A_n^i$ construits de façon à ce que $d_i(A_1^i, \ldots, A_n^i)$ soit très grand. Si on note $b_i = \max_{A_1,\ldots,A_n} d(A_1, \ldots, A_n)$ le véritable pire cas, et $m_i(\alpha_1, \ldots, \alpha_n)$ la borne calculée en calcul réseau, on a $d_i(A_1^i, \ldots, A_n^i) \leq b_i \leq m_i(\alpha_1, \ldots, \alpha_n)$, et le pessimisme $m_i(\alpha_1, \ldots, \alpha_n) - b_i$ peut être borné pas $m_i(\alpha_1, \ldots, \alpha_n) - d_i(A_1^i, \ldots, A_n^i)$.

En utilisant cette borne inférieure sur un réseau AFDX représentatif de 6500 flux de données, il a été montré dans [45] que le pessimisme légendaire du calcul réseau était, en réalité, autour de 20%...

Le résultat de l'expérience est reporté à la figure 20. Pour chaque VL sont tracés les bornes calculées avec des fonctions linéaires par morceaux concaves (ICC), avec la stratégie présentée au paragraphe 4.3 (ShSt), avec les fonctions qui capturent le mieux les comportements (UPP), et la borne inférieure (*Lower Bound*). Les VL sont triés par ordre croissant suivant la borne UPP, ce qui explique que cette fonction soit la plus lisse. Le véritable pire cas se situe entre la courbe UPP et la courbe Lower Bound.

À titre personnel, la faible valeur du pessimisme fut une bonne et une mauvaise nouvelle. Bonne car elle montrait que cet outil théorique dans lequel j'avais investi

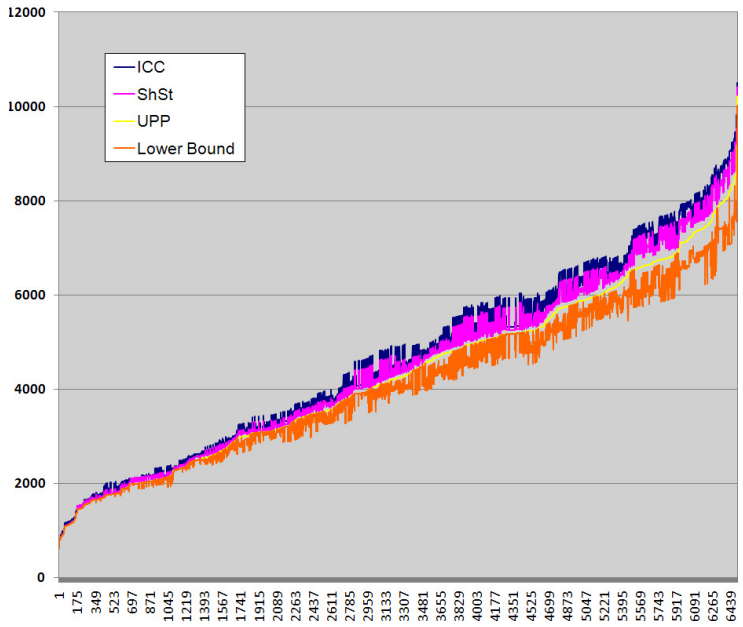


Figure 20 – *Comparaison de délais calculés par différentes méthodes sur une configuration réaliste de 6500 flux de données.*

restait tout à fait pertinent pour analyser les réseaux embarqués. Mauvaise parce qu'elle fermait une piste de recherche, ou plutôt la rendait moins prioritaire : en effet, du point de vue industriel, il était inutile d'investir beaucoup d'énergie pour ne gagner que 20%. Si d'un point de vue académique, la recherche du pire cas exact est en défi ambitieux, du point de vue d'une recherche à visée plus industrielle, utiliser une méthode plus précise, et certainement plus complexe, implique de développer un nouvel outil, plus complexe que le précédent.

À ce jour, mes contacts industriels considèrent qu'un gain limité à 20% ne justifie pas de développer un nouvel outil suivant les exigences d'un processus de certification (cf. paragraphe 2.3).

Notons que des travaux plus récents suggèrent que le pessimisme pourrait être encore plus réduit, de l'ordre de 10% [112].

4.5. Un modèle intégrant bits et paquets

Le calcul réseau modélise des quantités de données, envoyées par les flots (courbes d'arrivées), ou servies par les éléments réseau (courbes de service). Il ne modélise pas, dans ses outils, la décomposition en paquets des données, alors que cette décomposition peut avoir un impact sur les performances, par exemple dans la politique *Weighted Round Robin* mais aussi sur des passerelles, qui font passer les données d'un type de réseau à un autre, en adaptant donc le format

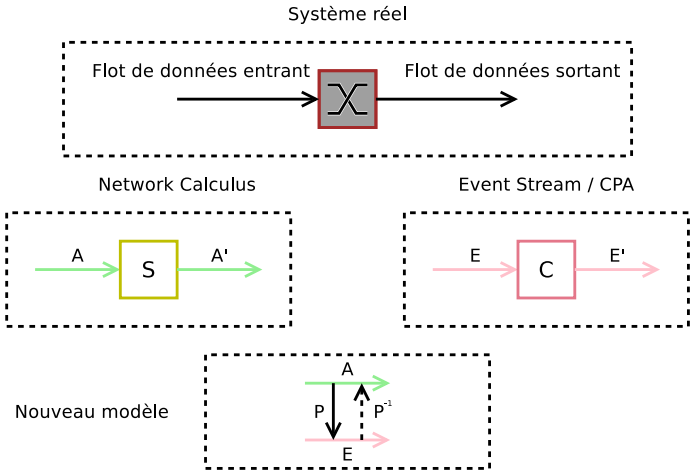


Figure 21 – *Modèle reliant quantité de donnée et nombre d'événements*

des paquets (en fragmentant un paquet en plusieurs autres, ou en assemblant plusieurs paquets en un seul).

Inversement, la théorie dite *Event Stream/Compositionnal Performance Analysis* (CPA) [105, 81, 70, 106] modélise elle des nombres cumulés d'événements, un événement étant soit la réception d'un paquet ou le réveil d'une tâche, à l'aide d'*event stream* qui sont en fait des courbes d'arrivée qui comptent les événements, et non les quantités de données comme en calcul réseau.

Avec Pierre Roux, nous avons développé un modèle théorique qui relie le décompte de paquets (fonction E issue des *event stream*) et le décompte des quantités de données (fonction A, issue du calcul réseau), par le biais de la fonction qui capture les tailles de paquets (fonction P telle que $E(t) = P(A(t))$), introduite dans [21, 22]. Ce modèle a de bonnes propriétés théoriques, comme nous l'avons montré dans [50, 51].

Nous avons ensuite travaillé avec des experts des *event stream*. Nous avons alors pu reformuler certaines expressions de leur domaine, liées à la charge de travail associée à un ensemble d'événement (*workload*) en un langage formel commun, et définir un modèle "CPA+" qui prend en entrée non pas seulement un *event stream* mais aussi une courbe d'arrivée, afin de calculer de meilleures bornes sur le temps de réponses [86].

L'évaluation de ces travaux se heurte à un fort effort d'implémentation. Les outils de calcul réseau ou CPA ne manipulent qu'un type de contrat (courbe d'arrivée ou *event stream*) et aller modifier les outils existants pour manipuler trois courbes (courbe d'arrivée, *event stream* et courbe de paquets) demanderait une modification profonde des codes.

4.6. Conclusion

J'ai présenté dans cette section mes travaux, principalement de 2007 à 2012, consacrés à l'amélioration des valeurs des bornes calculées à l'aide du calcul réseau. Une première partie a portée sur la politique FIFO (paragraphe 4.1), une seconde sur la politique NP-SP (paragraphe 4.2), une troisième sur la modélisation efficace des flux sporadiques (paragraphe 4.3). Tout cela pour conclure, au paragraphe 4.4, que le calcul réseau n'était pas si pessimiste que ça (sur une configuration réaliste, ce pessimisme est inférieur à 20%).

La question du plus petit temps de réponse n'étant plus si critique, je me suis intéressé à la possibilité de modéliser d'autres choses que les politiques FIFO/NP-SP utilisées dans les réseaux AFDX.

Je suis revenu à la question des performances, par le biais d'une réflexion sur le modèle lui même et de ses liens avec le modèle "*event stream*" (paragraphe 4.5).

Chapitre 5

Analyser plus

Une fois passée la révolution de l'AFDX, une fois démontré qu'il était possible de faire un réseau embarqué temps réel critique connectant des centaines d'abonnés, les ingénieurs se sont bien sûr posé la question de savoir si on pouvait penser à améliorer cette technologie AFDX, ou si d'autres secteurs industriels pouvaient eux aussi mettre en place ce type de technologie.

Une amélioration naturelle du réseau AFDX était une augmentation du débit : après avoir fait voler un réseau temps réel à 100Mb/s, pouvait on passer à 1Gb/s ? Cette question ouvre de nombreux enjeux dans la couche physique, dans la conception des circuits, mais du point du vue du calcul réseau, ce n'est qu'un paramètre.

Je vais présenter des travaux menés sur les politiques d'équilibrage de charge (paragraphe 5.1), sur la politique EDF (paragraphe 5.3), sur la technologie AVB (paragraphe 5.2) et sur la prise en compte directe de l'ordonnancement des tâches (paragraphe 5.4).

5.1. Les politiques d'équilibrage de charge

Une des difficultés industrielles de conception liée au réseau, c'est que c'est une ressource partagée par de nombreux équipements, par de nombreuses équipes, et chaque modification du contrat d'un système a un impact sur les garanties de performances des autres systèmes.

Sur les calculateurs, cette contrainte a été levée grâce à l'IMA (*Integrated Modular Avionic*) qui permet, avec des mécanismes de partitionnement spatial et temporel, d'héberger plusieurs applications sur un même calculateur, en plaçant

une application dans une *partition* à laquelle est alloué un budget, de telle manière qu'on puisse garantir des performances à l'application uniquement en fonction du budget de sa partition, indépendemment des budgets des autres partitions.

La question qui se posait était donc : peut-on partitionner virtuellement le réseau ? Peut-on découper, virtuellement, ce réseau à 100Mb/s en 10 réseaux à 10Mb/s, en assurant aux flux utilisant une partition une performance ne dépendant que du budget de la partition ?

Une réponse à cette question, ce sont les politiques à équilibrages de charge (*fair queuing*).

5.1.1. GPS et P-GPS

La politique idéale de partage de charge, GPS (*Generalized Processor Sharing*, [103]), suppose que l'on divise la capacité d'un serveur entre n classes, en associant à chaque classe d'index i un paramètre ϕ_i, dont la sémantique est, globalement, que chaque classe reçoit une fraction $\frac{\phi_i}{\sum_j \phi_j}$ de la capacité du serveur (sa définition plus formelle prend en compte le fait que si une classe n'utilise pas sa quote-part, elle est répartie entre les autres classes, suivant leurs poids respectifs).

Dans [99], avec William Mangoua Sofack, nous avons traduit la politique GPS dans le cadre du calcul réseau, et donné le service résiduel associé à une classe, qui, comme espéré, permet de calculer les délais des flots utilisant cette classe uniquement en fonction des valeurs des paramètres de partage $\phi_1, \ldots, \phi_n$.

Mais la politique GPS est une politique idéale, qui suppose que l'on peut toujours interrompre le service d'un flot pour en servir un autre, alors que dans les réseaux, on sert en général les flots trame par trame.

De nombreuses approximations de cette politique ont été définies. La plus célèbre est la politique P-GPS [103], aussi nommée WFQ (*Weighted Fair Queuing*). Il a été montré que si chaque paquet p avait été émis à une date d_p^{GPS} par l'algorithme idéal GPS, la date $d_p^{\text{P-GPS}}$ a laquelle P-GPS émet cette trame est en retard d'au plus le temps d'émettre une trame de taille maximale $L^{\max}$, c-à-d $d_p^{\text{P-GPS}} - d_p^{\text{GPS}} \leq \frac{L^{\max}}{C}$, avec C le débit du lien.

Mais cette politique n'est pas implantée en pratique, car sa complexité est trop importante : $O(n)$, si n est le nombre de classes.

De nombreuses approximations de GPS ont été définies par la suite : chacune devant avoir une complexité inférieure à $O(n)$ tout en introduisant un retard par rapport à GPS le plus faible possible.

5.1.2. Deficit Round Robin

La politique *Deficit Round Robin* (DRR) est une approximation populaire de GPS car sa complexité est $O(1)$. En quelques mots, elle associe elle aussi un

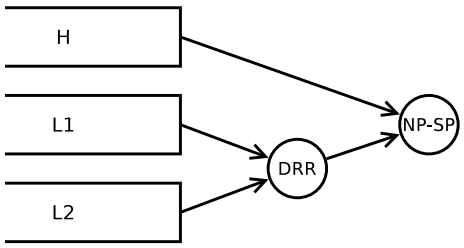


Figure 22 – *Architecture d'ordonnancement hiérarchique DRR/NP-SP*

paramètre ϕ_i à la file d'indice i, ainsi qu'un compteur D_i et fonctionne ensuite comme un tourniquet. À chaque sélection de la file par le tourniquet, on augmente le crédit D_i de la valeur du paramètre ϕ_i, puis on émet les trames en tête de file, en retirant de la valeur du crédit la taille de la trame, tant que cela ne mène pas à une valeur négative.

Toujours avec William Mangoua Sofack, nous avons montré comment la politique DRR pouvait se modéliser en calcul réseau [54].

Soulignons là une particularité du calcul réseau : sa capacité à traiter un ordonnancement hiérarchique. Elle est rarement utilisée, car les ordonnancements hiérarchiques ne sont pas communs dans un contexte temps-réel. Mais présentons d'abord ce qu'est un ordonnancement hiérarchique, et illustrons le sur un exemple.

Un ordonnancement hiérarchique utilise en fait plusieurs politiques d'ordonnancement en cascade : les n flux d'entrée sont partitionnés en m classes ($m < n$), et à l'intérieur de chaque classe, une politique est utilisée pour sélectionner le prochain flux à servir. Cela réduit donc le nombre de flux éligibles à m. On peut ensuite recommencer à partitionner, sélectionner et réduire, en autant d'étapes que souhaité, pour finir par sélectionner un seul flux.

Dans notre cas, on peut considérer qu'il existe une classe de très haute priorité, H, qui doit passer avant toutes les autres, et que le reste de la capacité doit être partagé suivant une politique DRR, comme illustré sur la figure 22. Ce type d'ordonnancement a été normalisé sous l'expression "Enhanced Traffic Scheduling for Bandwidth Sharing Between Traffic Classes" dans l'extension Qaz au standard IEEE 802.1Q [6], dédiée aux réseaux pour les fermes de calcul (*data center*).

En calcul réseau, sous réserve que les types de service (simple ou strict) soient compatibles, il suffit de composer les expressions de service résiduels de chaque politique pour analyser un ordonnancement hiérarchique.

Une fois ce résultat théorique établi, nous avons étudié son usage dans le domaine avionique.

Dans une étude faite avec Thales Avionics et RTaW [46], nous avons regardé en combien de classes on pouvait couper un réseau tout en gardant les mêmes performances qu'un réseau sans indépendance.

Sachant que GPS n'offre à un flux qu'une fraction de la bande passante, et que DRR introduit lui même un retard par rapport à GPS, pour que chaque classe ait

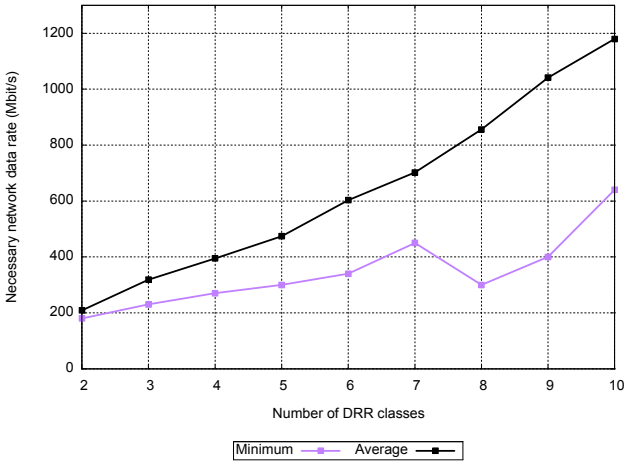


Figure 23 – *Débit minimal que doit avoir un réseau AFDX pour pouvoir être découpé en n classes DRR de manière à ce que chaque flux respecte ces contraintes de latence.*

les mêmes performances dans une classe DRR que sur un réseau sans partition à 100Mb/s, il faut augmenter la capacité du réseau que l'on cherche à partager.

Les résultats montraient que, pour pouvoir découper, en appliquant une politique DRR, un réseau en n classes de manière à ce que chaque classe subisse les même délais que si toutes les classes se mélangeaient sur un réseau AFDX à 100Mb/s, il faut multiplier la capacité du réseau par environ n.

L'utilisation d'un ordonnancement DRR/SP, dans lequel les flux avec des contraintes de latence très basses sont traités en priorité haute a aussi été étudié, avec là aussi un comportement globalement linéaire, mais avec une pente plus favorable.

Ce résultat, mi figue mi raisin, signifie qu'avec DRR, le coût de l'indépendance est globalement linéaire. Mais que l'indépendance des performances dans un système partagé soit un problème plus compliqué que le partitionnement temporel sur un calculateur unique n'est pas spécifiquement surprenant.

On pourra objecter que les politiques dirigées par le temps permettent une telle indépendance. En effet, l'existence d'une horloge commune transforme quasiment un système distribué en système centralisé.

5.1.3. Weighted Round Robin

La politique *Weighted Round Robin* (WRR) est une politique simple de partage d'un serveur. Étant donné n flux, $f_1, \ldots, f_n$, et des poids $w_1, \ldots, w_n$ associés à chaque flux, le serveur va travailler par cycles successifs, et pendant chaque cycle, offrir à chaque flux f_i un nombre w_i d'opportunités d'émission d'un paquet (quelle que soit sa taille). Si les paquets ont tous la même taille, c'est une approximation de GPS.

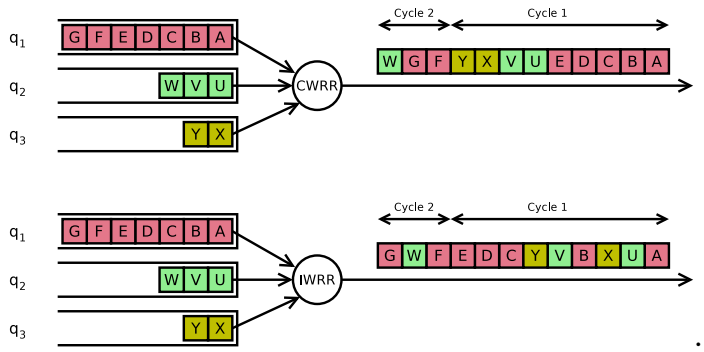


Figure 24 – *Illustration de deux variantes de WRR.*

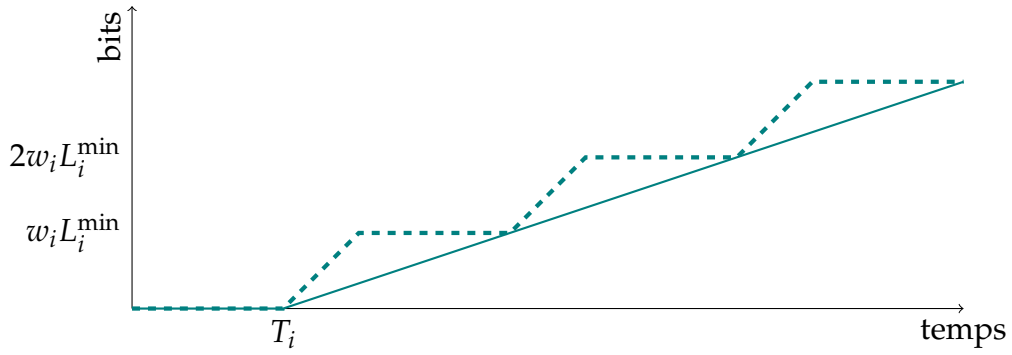


Figure 25 – *Courbes de service garanties par le partage d'un débit constant par une politique CWRR*

Il existe plusieurs variantes de WRR. La plus connue, Classical-WRR (CWRR) offrent les w_i opportunités "d'un bloc" [62], alors que la description initiale Interleaved-WRR (IWRR) entrelace les opportunités [84], comme illustré sur la Figure 24.

Pour CWRR, il apparaît assez clairement que le pire qui puisse arriver pour les paquets d'un flux f_i est d'arriver dans le serveur juste après que leur opportunité d'émission est passée, puis que toutes les autres files soient pleines de paquets les plus grands possibles. Si on note $L_j^{\max}$ la taille maximale d'un paquet du flux f_j, il est possible qu'il faille attendre que $\sum_{j\neq i} w_j$ paquets, soit $\sum_{j\neq i} w_j L_j^{\max}$ bits, soient servis avant de commencer à être servis. Ensuite, si le flux f_i envoie les paquets les plus petits possibles, de taille $L_i^{\min}$, alors le flux reçoit, à long terme, une fraction $\phi_i = \frac{L_i^{\min}}{L_i^{\min}+\sum_{j\neq i} w_j L_j^{\max}}$ de la capacité du serveur. Si le serveur offre un débit de capacité constante R, le flux f_i reçoit au moins le service latence-débit β_{R_i,T_i} avec $R_i = R\phi_i$ et $T_i = \frac{\sum_{j\neq i} w_j L_j^{\max}}{R}$ comme illustré par la courbe en trait continu de la Figure 24. Ce résultat existait dans la littérature de manière plus ou moins formelle [114, 75, 113].

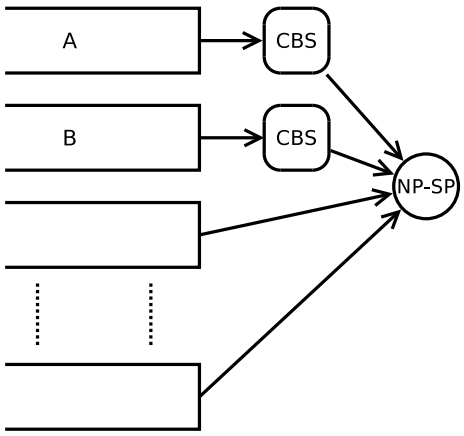


Figure 26 – *Architecture d'un port de sortie AVB*

Mais ce modèle ne représente pas le comportement réel, qui est une alternance de phases de service à la pleine capacité du serveur et de phase d'attente, comme illustré par la courbe en pointillés de la Figure 24.

J'ai donc proposé dans [20, § 8.2.4] une expression du service résiduel qui modélise ce comportement. J'ai ensuite collaboré, avec Seyed Mohammadhossein Tabatabaee and Jean-Yves Le Boudec à une modélisation équivalente de la politique IWRR [115].

5.2. Le réseau AVB

La technologie AVB (*Audio Video Bridging*) est une extension temps-réel d'Ethernet, initialement destinée au multimédia. Formellement, c'est une série d'ajouts au standard (IEEE 802.1 BA [4], IEEE 802.1 AS [5] , IEEE 802.1 Qat [3], IEEE 802.1 Qav [2]).

La technologie AVB a intéressé l'industrie de l'automobile en recherche d'une solution Ethernet temps-réel moins onéreuse que l'AFDX.

En terme de mécanisme pour le temps-réel, AVB se distingue des solutions présentées jusqu'à maintenant, car son mécanisme de partage de capacité dans les ports de sortie, appelé *Credit Based Shaper* utilise un crédit qui évolue avec le temps.

L'architecture générale est représentée sur la figure 26 : un port possède 8 files en entrée, une par classe de traffic, classées par priorité, dont les deux plus prioritaires s'appellent respectivement classes A et B, la classe A étant la plus prioritaire.

Les files des classes A et B sont contraintes en sortie par un *crédit shaper*. Pour chaque classe $X \in \{A, B\}$ est maintenu une variable c_W, dite de crédit. A chaque classe est aussi associé un paramètre, *idle slope*, noté $idSl_X$. Si on note C le débit en

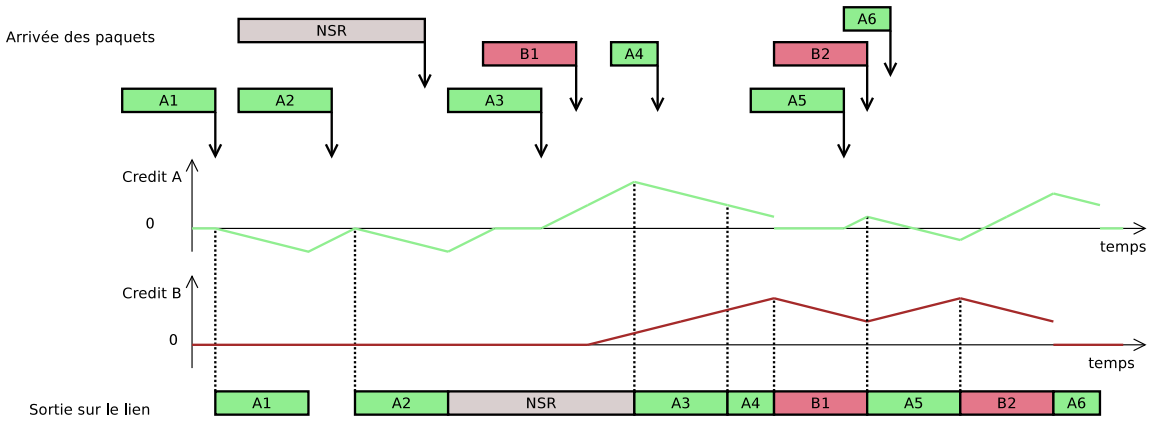


Figure 27 – *Exemple d'ordonnancement AVB avec évolution du crédit.*

sortie du port, on a la contrainte $idSl_A + idSl_B \leq C$.

Les règles de sélection d'une file X et d'évolution de son crédit c_X sont les suivantes :

— les crédits sont initialisés à 0 ;

— une trame ne peut être sélectionnée pour émission que si $c_X \geq 0$, *et* qu'il n'y a pas de trame plus prioritaire qui puisse être émise ;

— quand une trame est émise, le crédit diminue avec le temps avec une pente $idSl_X - C$ (appelée dans la norme *Send Slope*) ;

— une trame qui est sélectionnée est émise jusqu'à sa fin sans interruption (politique non préemptive) ;

— quand la file est non vide mais n'est pas sélectionnée (il y a donc au moins une trame qui y attend), son crédit augmente en fonction du temps avec la pente $idSl_X$;

— si une file est vide et que son crédit est positif, il est remis à 0, s'il est négatif, il augmente avec le temps suivant la pente $idSl_X$.

Cette politique à base de crédit contraint les flux les plus prioritaires à rester dans une certaine enveloppe.

Illustrons le comportement d'AVB sur un exemple. La figure 27 représente des arrivées des paquets, l'évolution des valeurs des crédits des classes A et B, et la sortie du lien.

À l'instant initial, tous les crédits sont nuls.

Lorsque la trame A1 du flux A est reçue, le crédit de la classe A n'est pas négatif, la trame A1 commence dont à être émise, et le crédit diminue à la vitesse $idSl_A - C$. Une fois cette trame émise, le crédit recommence à augmenter à la vitesse $idSl_A$.

Lorsque la trame A2 arrive, le crédit est encore négatif, et la trame doit attendre qu'il devienne nul pour pouvoir être sélectionnée pour émission.

On remarquera que le lien de sortie est inactif, alors qu'il existe une file en attente. L'ordonnancement AVB n'est donc pas *work conserving*.

À la fin de l'émission de A2, il n'existe qu'une trame de faible priorité (NSR : Non Stream-Reservation), qui est sélectionnée pour émission. Dans un ordonnancement non préemptif, lorsque la trame A3 est reçue, elle ne peut pas accéder au lien, et son crédit augmente donc.

À la fin de l'émission de la trame NSR, il y a deux trames en attente, A3 et B1. Les crédits de chaque classe sont positifs, l'ordonnanceur sélectionne donc la trame A3, plus prioritaire, puis la trame A4 à la suite.

À la fin de l'émission de A4, le crédit de la classe A est positif, alors qu'il n'y a plus de trame en attente dans cette classe. Il est donc ramené à 0. Et la trame B1 peut donc être envoyée.

À la fin de l'émission de B1, la trame A5 est envoyée : son crédit est positif et elle est plus prioritaire que B2.

À la fin de l'émission de A5, il existe une trame A6 en attente, mais le crédit de la classe A est négatif. C'est donc la trame B2 qui est sélectionnée pour émission.

De ce petit exemple, on peut remarquer que le crédit de A ne peut devenir positif que suite à l'effet de non-préemption, alors que le crédit de B peut avoir des valeurs plus grandes, puisqu'il peut être bloqué par l'émission des trames de A.

Nous avons, avec Joan Adrià Ruiz de Azua, modélisé en calcul réseau un port de sortie AVB, en donnant pour chaque classe, la courbe de son service minimal, celle de son service strict, celle de son service maximum et celle de *shaping* [107].

5.3. Ordonnancement EDF

La politique *Earliest Deadline First* (EDF) est une politique d'ordonnancement de tâches.

A chaque tâche est associée une date d'échéance au plus tard, et EDF sélectionne toujours celle avec l'échéance la plus proche. EDF est célèbre car, en considérant un processeur à vitesse constante et certains types de contrat sur les dates d'arrivée des tâches [15, 14] c'est une politique optimale, c-à-d que si un ordonnancement permet de respecter les échéances de chaque tâche, alors EDF le réalise aussi.

C'est néanmoins une politique qui n'est pas vraiment appliquée au niveau réseau, car elle suppose d'être capable de sélectionner le message avec l'échéance la plus courte, ce qui est trop coûteux en terme de calcul pour être faisable à la vitesse d'émission du réseau (un port Ethernet à 100Mb/s ne traitant que des trames de taille maximale envoie une trame toutes les 120μs.).

Ceci explique peut-être pourquoi elle a reçu peu d'attention dans la communauté du calcul réseau. Mais j'ai été intrigué par les travaux de Jorg Liebeherr,

présentés dans [92] : il y définit la notion de Δ-scheduler, qui généralise les politiques FIFO, EDF, et SP. L'étude est faite dans le cadre du calcul réseau, elle généralise les résultats connus sur FIFO, SP et EDF, en faisant l'hypothèse que le service offert soit strict et constant. Or, en calcul réseau, on sait que l'hypothèse "service strict" est nécessaire pour la politique SP, alors que l'hypothèse "service minimal" est suffisante pour FIFO.

Obtenir le même résultat dans le cadre du calcul réseau, en le généralisant à n'importe quel type d'arrivée de tâche et n'importe quel type de service m'a occupé de 2014 à 2018,

J'ai cherché à généraliser ce résultat à n'importe quel type de service en explicitant les équations fondamentales d'EDF en calcul réseau, et il a fallu les conseils de Jorg Liebeherr (qui trouva une erreur dans une première version) et la collaboration avec Anne Bouillard pour arriver à faire cette généralisation [20, § 7.3.4]. C'est à ce jour la preuve la plus difficile que j'ai eu à faire, et elle me laisse un goût d'inachevé. En effet, la preuve fait l'hypothèse d'un service strict, et je n'arrive toujours pas à savoir si cette condition est réellement nécessaire (existerait-il une preuve pour le service résiduel EDF à partir d'un service simple ?) et sinon, d'où vient le besoin de cette hypothèse supplémentaire ?

5.4. La prise en compte de l'ordonnancement des tâches productrices

Comme présenté au paragraphe 2.3, le calcul de performances d'un réseau se fait en général en considérant un contrat entre l'application émettrice et le réseau, basé simplement sur un nombre maximal de messages envoyés dans une fenêtre.

Avec David Doose, nous nous sommes demandés ce qu'il adviendrait si l'on avait plus d'informations.

En pratique, les données sont produites par des tâches, qui sont soumises à des contraintes d'ordonnancement sur le processeur local. De plus, dans le cas des systèmes embarqués critiques, le plus courant est d'avoir un ensemble connu de tâches, chaque tâche étant réveillée périodiquement. À chaque réveil, elle commence par lire des données, puis exécute des calculs, avant d'émettre des résultats. Ce modèle est nommé *Acquisition - Exécution - Restitution* (AER) [96].

En faisant donc l'hypothèse que les données étaient émises sur le réseau en fin d'exécution d'une tâche, nous avons proposé une nouvelle manière de calculer la courbe d'arrivée des flux issus d'un ensemble de tâches partageant un processeur suivant une politique de priorité fixe, et avons montré sur un cas d'étude théorique comment cela permettait de lisser la charge sur le réseau [38].

Nous avons repris la même idée dans [52], en considérant cette fois l'ordonnancement entre VL mis en place dans les *End-System* AFDX de la société Thales. Sur un cas d'étude réaliste, cela permettait un gain moyen de 38%, comme illustré

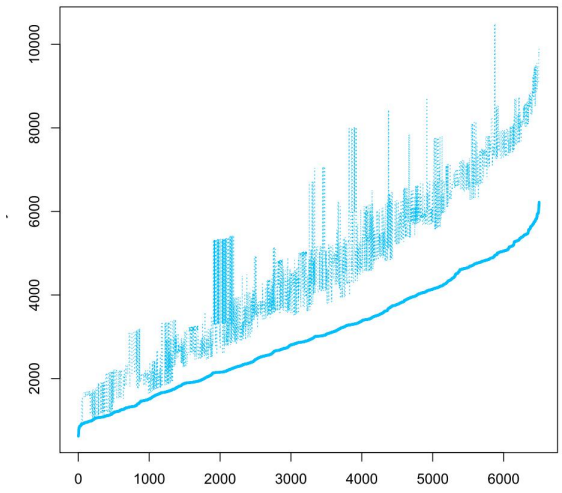


Figure 28 – *Bornes sur les latences de VL d'un réseau AFDX : avec prise en compte du contrat "BAG,L^{max}" (courbe haute) et avec prise en compte de l'ordonnancement dans le* End System *(courbe basse)*

sur la Figure 28.

Ces deux résultats ont reçu un intérêt industriel, mais ils créent une dépendance forte entre un ordonnancement local, sur un abonné du réseau, et les temps de réponse du réseau. Une telle dépendance impliquerait que toute modification de l'ordonnancement local nécessiterait une re-validation du comportement de tout le réseau. Entre un gain de performances et un alourdissement du processus industriel, l'arbitrage a été jusqu'à ce jour de privilégier le processus.

Chapitre 6

Augmenter la confiance

Comme présenté au paragraphe 2.3, avant de pouvoir voler dans l'espace aérien, un avion doit passer un processus de certification, qui consiste à convaincre une autorité que l'avion répond bien à certaines exigences.

En ce qui concerne ma partie, le calcul de latence, il s'agit de montrer qu'un message traversant le réseau subira un délai plus petit qu'une exigence. Si on revient au diagramme de la figure 5, cette propriété repose sur de nombreuses sous - propriétés.

1. Il faut tout d'abord que la théorie soit correcte. Il peut subsister des erreurs dans des articles publiés, comme ce fut le cas dans le domaine des latences de réseau avec une analyse de bus CAN [68] ou la théorie des trajectoires [91].
2. Il faut aussi que la façon dont le monde réel est représenté dans la théorie soit correcte. Par exemple, les matériels réels sont cadencés par des horloges qui peuvent dériver.
3. L'application de la théorie se fait à travers un logiciel, qui en implante souvent une sous-partie. Cette implantation peut être défectueuse (présence de bug).
4. Pour finir, le logiciel utilise des fichiers en entrée, et produit des fichiers en sortie. Si les fichiers ne représentent pas exactement le système auquel on s'intéresse (on peut par exemple ajouter dans le fichier un lien qui n'existe pas dans le système réel), les résultats ne seront pas applicables.

Si la recherche se focalise surtout sur le développement correct de théories, j'ai été amené, suite à des interpellations d'industriels, à travailler un peu sur deux aspects : la modélisation et l'implantation.

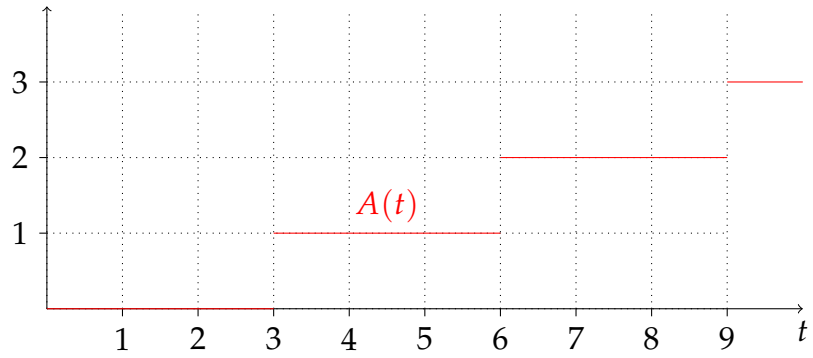


Figure 29 – *Courbe cumulée d'un flux périodique*

6.1. Calcul réseau et continuité

Pour introduire les questions de continuités, commençons par un exemple. Soit un flux qui, toutes les 3 unités de temps, envoie un paquet de taille 1, en commençant à l'instant $t = 3$.

Soit A la fonction cumulative qui représente ce flux. Jusqu'à l'instant $t = 3$, aucun paquet n'a été émis, la fonction est donc nulle. Après l'instant $t = 3$ et avant $t = 6$, un seul paquet a été reçu, donc la fonction vaut 1, etc. La question est : que vaut la fonction à l'instant 3 ? Que vaut $A(3)$?

Si $A(3) = 0$, la fonction est *continue à gauche*. Si $A(3) = 1$, la fonction est *continue à droite*. Seules ces deux valeurs méritent qu'on s'y intéressent.

En terme de *sémantique*, dans le cas d'une fonction continue à gauche, même si un paquet est émis dans le monde réel à $t = 3$, il n'est pas représenté dans sa courbe cumulative à $t = 3$, mais il le sera pour tout instant $t + \epsilon$, avec $\epsilon > 0$ aussi petit que l'on souhaite (plus petit qu'un battement de l'horloge du circuit électronique par exemple).

Ainsi, avec une fonction continue à gauche, un paquet reçu à un instant donné n'est visible dans sa courbe cumulée que "juste après". Cette notion de "juste après", un peu contre intuitive, peut trouver une justification au niveau le plus bas de l'informatique, au niveau du circuit, du processeur. Si une instruction dépose une donnée, elle ne sera manipulable qu'à l'instruction suivante, après un temps faible mais non nul.

Inversement, avec une fonction continue à droite, un paquet reçu à un instant t est représenté par la valeur à l'instant t de sa fonction cumulée.

D'un point de vue mathématique, les preuves de calcul réseau font souvent l'hypothèse que les fonctions sont toujours continues à gauche, pour une question un peu technique (présentée dans [39]).

Un partenaire industriel, Cédric Mauclair, nous a interpellés sur l'impact de cette hypothèse de continuité à gauche. Avec quelques collègues de l'ONERA,

Guillaume Dufour et Luca Santinelli, nous avons revisité les principaux résultats de calcul réseau en faisant l'hypothèse que les fonctions cumulées étaient continues à droite [39]. Nous avons montré que cela changeait les preuves, que cela modifiait légèrement les formules, mais que cela n'avait pas d'impact sur les valeurs numériques des résultats.

Cela confortait l'intuition que le choix de continuité était plus un choix de modèle mathématique qu'un aspect du système réel, et confortait les résultats obtenus jusqu'à présent.

6.2. Assistant de preuve

Les assistants de preuves sont des logiciels qui manipulent des preuves. Cette manipulation est au mieux complètement automatique, le plus souvent semi-automatique, en interaction avec un humain. Mais surtout, un assistant de preuve va vérifier chaque étape élémentaire d'une preuve. Ces assistants sont aussi souvent utilisés pour prouver qu'un *logiciel* est correct.

Dès lors, une preuve ou un logiciel vérifié par un tel assistant acquiert un très haut niveau de confiance.

Or, comme exposé au début du paragraphe 6, le réseau étant un élément critique de l'architecture informatique d'un avion, le résultat du logiciel qui calcule les bornes sur les latences doit bénéficier d'une très haute confiance.

Les assistants de preuve présentent donc un intérêt pour fournir des résultats de haut niveau de confiance, surtout dans le domaine du calcul réseau qui a une expression très mathématique, proche de ce que les assistants de preuve manipulent. Les deux principaux assistants de preuve sont à ce jour Isabelle et Coq.

Deux applications apparaissent à première vue : vérifier la théorie et vérifier le logiciel.

6.2.1. Vérifier la bonne application des résultats théoriques par le logiciel

Nous nous sommes tout d'abord intéressé à la vérification du logiciel, ou, plus précisément, à la vérification de la bonne application des résultats théoriques par le logiciel.

Commençons par une analogie : il est relativement compliqué de prouver qu'un algorithme de tri, surtout un algorithme efficace, est correct. Il est beaucoup plus simple de vérifier que la liste $1,3,7,9$ est le tri de la liste $7,3,1,9$. De même, prouver qu'un algorithme calcule la racine carré d'un nombre est bien plus compliqué que de vérifier que le carré du résultat est assez proche du nombre initial.

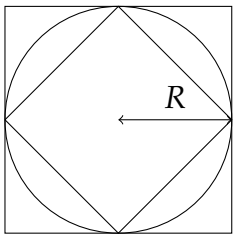


Figure 30 – *Approximations de l'aire d'un cercle par des carrés*

Continuons avec un second exemple, plus proche de ce qui a été réalisé. Imaginons un logiciel qui cherche à borner l'aire de figures. Soit un cercle de rayons R : on peut sur-approximer son aire par l'aide d'un carré englobant, $4R^2$, et la sous-approximer par celle d'un carré inscrit, $2R^2$.

Lorsque le logiciel approxime l'aire d'une figure contenant un cercle, il peut dire à un assistant de preuve quel théorème il a utilisé. On demande ensuite à l'assistant non pas de prouver les théorèmes géométriques, mais de vérifier que les bons théorèmes ont été appliqués.

L'application au calcul réseau pour la vérification de la correction de latence suivait cette idée : non pas de vérifier la théorie elle-même, mais d'imposer au logiciel de fournir à un assistant de preuve la trace de son raisonnement, quel théorème a été appliqué sur tel flux et tel serveur.

Cette approche s'est avérée très prometteuse : avec Stephan Merz, expert Isabelle, nous avons encadré un stagiaire, Étienne Mabille pendant 6 mois, puis il a continué ses travaux pendant 6 mois de plus en CDD chez RealTime-at-Work [94, 95, 30]. Au bilan, a été développé en un an, un logiciel d'analyse de réseau AFDX et une bibliothèque Isabelle. Le logiciel n'implantait pas la version la plus à jour de la théorie, mais globalement les algorithmes présentés dans [79]. Quant à la bibliothèque Isabelle, de la même manière, elle n'encode que la partie nécessaire à une telle analyse, mais ce qui inclus tout de même les notions de courbes cumulées, de serveur, de courbes d'arrivée, de service, et les principaux théorèmes de calcul réseau.

Au final, l'outil est capable d'analyser en quelques minutes une configuration AFDX représentative, et de générer la preuve. Et Isabelle valide la preuve en quelques heures. Cette réalisation en 12 mois de travail d'un ingénieur débutant est une grande réussite si on la compare à l'effort nécessaire pour développer un logiciel suivant les critères de la DO 178 [1].

6.2.2. Vérification de la théorie du calcul réseau

Nous nous sommes ensuite intéressés à la question de la vérification de la théorie du calcul réseau elle-même.

C'est le sujet de la thèse de Lucien Rakotomalala. L'objectif cette fois était

bien de vérifier la théorie elle même, et non son application. En un an, Lucien Rakotomalala a été capable de prouver en Coq les bases du calcul réseau : principaux résultats sur les dioïdes, propriétés des courbes d'arrivée et de service, service résiduel SP et FIFO, borne sur les délais, etc [104]. Il ne manque à ce jour qu'un résultat lié à la paquetisation pour permettre de valider complètement l'analyse d'un réseau de type AFDX.

6.2.3. Vérification de la correction des opérations sur les fonctions

Le calcul réseau manipule des fonctions, courbes d'arrivée et de service principalement. Comme évoqué dans la présentation du calcul réseau, on peut prouver que si un serveur offre un service strict β à deux flux A_1, A_2 de courbes d'arrivée respectives α_1, α_2, alors il offre au flux A_1 au moins le service $[\beta - \alpha_2]^+_\uparrow$, d'où l'on déduit que ce flux subira dans ce serveur un délai plus petit que $hDev(\alpha_1, [\beta - \alpha_2]^+_\uparrow)$. Ce type de résultat a été prouvé en Coq. Mais pour obtenir une valeur numérique, il faut, soit une expression analytique simple de cette expression, soit un algorithme général. Or, si l'on cherche des résultats précis, il n'existe pas d'expression simple, et même si les algorithmes existent [28, 29] ils sont trop complexes pour être réalisés à la main, et on se repose sur une implémentation logicielle. Dès lors, quelle confiance avoir en les résultats calculés ? Il peut y avoir une erreur dans l'implémentation, voire dans les algorithmes eux-mêmes.

Notre proposition est de fournir des algorithmes de vérification de calculs. Par exemple, si on se restreint à la classe UPP, la classe de fonctions affines par morceaux avec un comportement pseudo-périodique, on peut restreindre le critère d'égalité entre deux fonctions f et g

$$\forall t \in \mathbb{R}^+ : f(t) = g(t) \tag{6.1}$$

par le critère

$$\forall t_I \in I_{f,g} : f(t_i) = g(t_i) \tag{6.2}$$

avec $I_{f,g}$ un ensemble fini de points "bien choisis". Dès lors, si un logiciel calcule h comme étant la somme de deux fonctions f et g, il suffit de vérifier que pour tout t_i dans l'ensemble $I_{f+g,h}$, $f(t_i) + g(t_i) = h(t_i)$.

En appliquant ce principe, non pas de calcul mais de condition suffisante de validité à l'ensemble des opérateurs utilisés en calcul réseau, nous pouvons prouver la validité de résultats sans avoir à prouver la correction des implémentations.

Chapitre 7

Évaluer des mécanismes réseau

Une forte activité de la communauté réseau, au sens large, consiste à proposer de nouveaux mécanismes, et à les évaluer, par simulation, méthode analytique ou expérimentation.

En ce qui concerne les réseaux temps réels, la communauté est plus réduite et, à ma connaissance, les propositions d'évolution sont plus portées par des industriels, dans des produits ou des standards. Les évaluations des performances temps-réels sont effectuées a posteriori.

Je vais présenter dans ce chapitre quelques travaux d'évaluation de mécanismes de réseaux temps-réels.

7.1. Préemption et fragmentation

Le réseau AFDX offre deux niveaux de priorité. Les flux ayant des exigences de latence faible peuvent être mis dans la classe de haute priorité. Et ces flux sont souvent des VL avec des trames de petites tailles. Lorsque l'on regarde ce qui constitue le délai d'attente d'une trame de haute priorité, on trouve deux termes : le temps d'attente dans la file, qui est partagée avec les autres flux de haute priorité, et un temps de "non-préemption" (cf. figure 17) lié au fait que lorsqu'une trame de haute priorité cherche à accéder au lien en sortie, ce même lien peut être utilisé par une trame de plus faible priorité, et qu'elle doit attendre la fin de l'émission de la dite trame.

Pour réduire ce temps d'attente, nous avons envisagé avec Luca Santinelli et Rubén Trillo Flores deux mécanismes : la fragmentation à l'émission et la préemption.

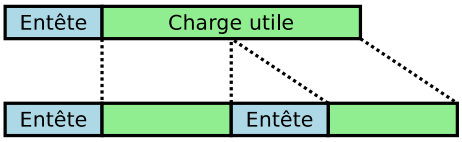


Figure 31 – *Fragmentation d'une trame en deux fragments : la charge utile est divisée en deux, mais chaque fragment a son propre entête*

Dans la fragmentation, une trame de grande taille est découpée en n trames de plus petite tailles, mais la somme des n plus petites est supérieure à la taille initiale. Il y a donc une utilisation moins efficace du réseau, mais moins de gêne pour les trames de plus haute priorité.

Il faut donc évaluer le gain associé à la baisse de la gêne, et le comparer avec la perte d'efficacité.

La préemption, elle, suppose que l'on peut interrompre une trame en cours d'émission, pour émettre une trame de plus haute priorité. Une fois la trame de haute priorité émise, deux solutions distinctes existent : soit ré-émettre la trame interrompue depuis le début (*preemption resume*), soit la ré-émettre depuis l'instant d'interruption (*preemption restart*). La première solution est plus efficace en terme d'utilisation du réseau, mais elle demande un matériel plus complexe. La seconde gâche un peu de la capacité du réseau, mais est plus simple.

Lors de son stage de fin d'étude, Rubén Trillo Flores a étudié la fragmentation et la préemption, mais n'a publié que les travaux sur la préemption [116].

Maintenant que l'IEEE a standardisé un mécanisme de préemption pour Ethernet [9, 82], ces travaux mériteraient ré-évaluation.

7.2. Synchroniser, oui, mais pas trop

Deux approches co-existent depuis les origines de l'informatique pour spécifier un partage de ressources, de calcul ou réseau : le partage temporel ou le partage dynamique.

7.2.1. Comparaison des partages temporels et dynamiques

Dans un partage temporel, ou déclenché par le temps (*Time-Triggered*, TT ; *Time Division Multiple Access*, TDMA), des périodes de temps d'usage d'une ressource sont, statiquement, à l'avance, allouées à des utilisateurs.

Les avantages et inconvénients de cette approche sont bien connus depuis des années [13].

Le principal inconvénient est le besoin d'une horloge commune entre l'ensemble des participants, qui implique un coût supplémentaire (car une horloge commune nécessite souvent un matériel (*hardware*) dédié, pour avoir une horloge

assez précise), mais aussi des questions de sécurité (l'horloge commune est un point unique de panne, et se pose la question du temps nécessaire pour retrouver une horloge commune au démarrage ou redémarrage). Un second inconvénient, moindre, est le besoin de calculer l'ordonnancement global, de savoir allouer à chaque flux les créneaux temporels dont il a besoin. Mais des solutions efficaces existent à ce jour. Un troisième inconvénient est la sous-utilisation de la capacité du réseau : si un flux n'utilise pas le créneau temporel qui lui est réservé, il n'existe pas en général de moyen de l'allouer à un autre utilisateur.

Un dernier inconvénient, souvent oublié, est le besoin de synchroniser les applications productrices de données et le réseau : si une application a un créneau réseau toutes les T unités de temps, et qu'elle calcule une donnée juste après un créneau, elle doit attendre T unités de temps avant de pouvoir émettre cette donnée sur le réseau.

Les avantages sont assez évidents. Le premier avantage est un très faible temps de traversée du réseau : si tous les créneaux temporels ont été bien alloués, dès qu'une trame est reçue sur un lien d'entrée dans un commutateur, il existe un créneau temporel "juste après" pour quitter le commutateur sur un lien de sortie. Le second avantage est l'absence de jigue : la variabilité temporelle a un impact négatif sur les systèmes cyber-physiques, et il y a généralement des contre-mesures mises en place pour l'absorber. Quand il n'y a plus de jigue, le logiciel peut être plus simple. Soulignons que ce gain est très difficile à quantifier.

Les avantages et inconvénients des partages dynamiques sont, globalement, les complémentaires des partages temporels : un système plus simple, au niveau logiciel et matériel, et donc moins coûteux, plus facile à certifier ; une meilleure utilisation du réseau ; des délais plus réduits pour faire passer des communications asynchrones (comme les alarmes).

Étrangement, les partisans de chaque solution assurent que leur solution est plus composable que l'autre. Comme présenté au paragraphe 2.3, l'intégrateur réseau doit considérer les contrats de chaque utilisateur du réseau et vérifier que les contraintes sont respectées.

Dans le cas des systèmes déclenchés par le temps, la facilité de composition vient du fait qu'une fois que chaque application connaît ses fenêtres temporelles d'émission, son comportement est indépendant des autres applications. Mais cela passe sous silence comment est négocié l'allocation des fenêtres, qui impose l'ordonnancement des tâches productrices des données. De plus, si une application demande une modification de son contrat, ou si une nouvelle application est ajoutée au système, il se peut que l'on puisse lui allouer des nouvelles fenêtres sans modifier les autres, mais il se peut aussi qu'il faille modifier des fenêtres déjà allouées.

Dans le cas des systèmes dynamiques, la modification d'un contrat, ou l'ajout d'un nouveau, demande à l'intégrateur réseau de vérifier que dans la nouvelle configuration, toutes les contraintes sont respectées. Mais si cette vérification

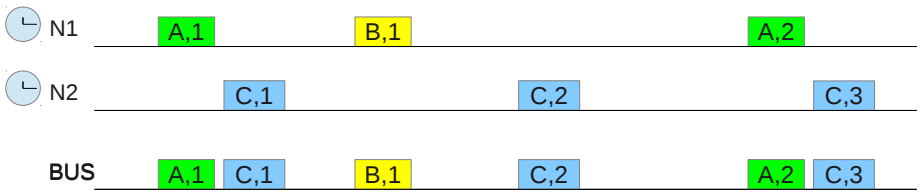


Figure 32 – *Partitionnement temporel de l'accès à un bus (TDMA)*

réussit, aucun contrat n'a à être modifié. Si cette vérification échoue, il faut générer une nouvelle configuration, en modifiant un routage, une priorité, etc.

On a donc, dans les deux cas, besoin de ré-évaluer globalement l'usage du réseau (avec dans le cas du partage par le temps la recherche de fenêtres temporelles pour le nouveau venu) et si l'évaluation échoue, modifier des contrats existants.

Il n'existe pas à ma connaissance d'étude comparant de manière quantitative la capacité de composition des systèmes à partage temporel à celle des systèmes à partage dynamique.

7.2.2. Usage des décalages locaux (*offsets*)

Dans le cadre de l'ordonnancement de tâches [74], comme dans l'étude de temps de réponses d'un bus CAN [101], il est courant de considérer (ou d'imposer) des décalages (*release dates*, *offsets*) aux tâches ou aux trames. L'idée n'est pas, comme dans les systèmes à partage de temps, d'imposer une partitionnement strict, pour éviter tout conflit d'accès (au processeur ou au médium), mais de considérer que leur usage lisse la charge, car les utilisateurs de la ressource (tâches ou flux) ne peuvent pas chercher à y accéder tous en même temps.

Dans le cas d'un réseau, on parle d'*offset* pour signifier le fait que chaque émetteur possède localement un ordonnancement statique, un partage dans le temps des demandes d'accès au médium, mais que les émetteurs ne sont pas synchronisés entre eux. L'usage d'un ordonnancement local permet d'éviter des conflits entre flux issus du même émetteur, et même s'il n'évite pas les conflits entre flux issus d'émetteurs différents, il permet de les réduire.

Illustrons cela sur un exemple. Considérons la Figure 32 : l'ordonnancement global garantit que les flux A, B, C ne sont jamais en conflit pour accéder au médium. Maintenant, si les horloges des noeuds N1 et N2 ne sont plus synchronisées, le système devient un système à ordonnancement local : le flux C peut être en conflit soit avec le flux A, soit avec le flux B, mais jamais avec les deux.

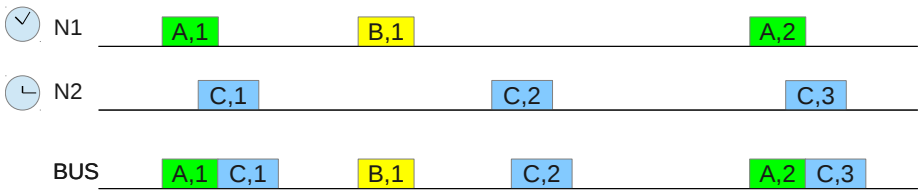


Figure 33 – *Ordonnancement local pour l'accès à un bus*

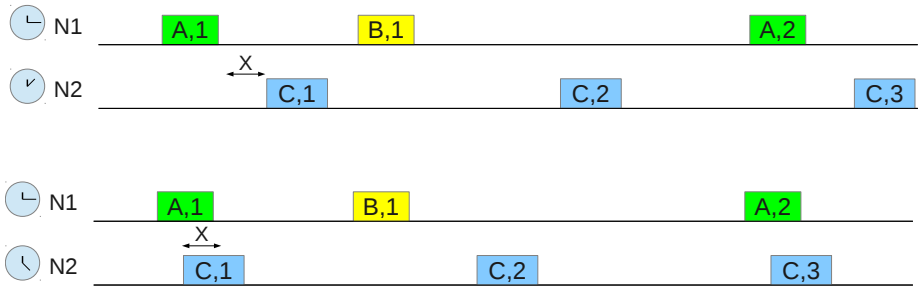


Figure 34 – *Ordonnancement à déphasage borné pour l'accès à un bus*

7.2.3. Systèmes à déphasage borné

Reprenons l'ordonnancement global de la Figure 32 et supposons cette fois que la synchronisation entre les horloges de N1 et N2 n'est pas parfaite, mais néanmoins que le déphasage est borné par une valeur X. Dès lors, le flux C peut être en conflit d'accès au bus avec le flux A, mais pas avec le flux B.

Nous avons voulu, avec Hugo Daigmorte, étudier de tels systèmes, où la synchronisation entre noeuds est "faible" mais bornée. Nous entendons par "faible" le fait qu'un système à partitionnement temporel nécessite une synchronisation entre horloge de l'ordre de grandeur de l'émission d'un bit (soit $10\mu s$ à 100Kb/s), alors que nous étudions des systèmes où la synchronisation serait 100 fois moins précise (de l'ordre de la milliseconde).

La contribution a dans un premier temps consisté à développer des méthodes pour pouvoir borner les délais dans cette classe de système [64, 66]. Dans un second temps, nous avons évalué quel pouvait être le gain associé à cette méthode, nous avons montré qu'alors qu'un bus CAN sans maîtrise des identifiants pouvait être chargé à environ 50% sans synchronisation, une synchronisation de simplement 1ms permettait de monter la charge à plus de 80% [66].

Chapitre 8

Synthèse des contributions

Alors que j'avais promis dans l'introduction une synthèse, une mise en cohérence et une cartographie, les chapitres 4, 5, 6 et 7 ont pu donner une impression de catalogue [1].

Je vous propose ici donc une synthèse suivant 4 axes, un peu différente de l'organisation du mémoire.

J'ai donc présenté dans ces parties mes travaux sur la vérification de bornes de mémoire et de latence dans les réseaux temps réel, à l'aide du calcul réseau : la modélisation d'ordonnancements réseau en calcul réseau, la confiance dans le calcul réseau, une extension du modèle du calcul réseau et l'évaluation des mécanismes des réseaux temps réels.

Lorsque j'ai commencé à travailler dans cette thématique, vers 2007, la théorie du calcul réseau avait la réputation d'être pessimiste (cf. chapitre 4). J'ai donc travaillé quelques années à l'amélioration de la méthode et à l'évaluation de ce pessimisme. Lorsqu'en 2012, il est apparu que ce pessimisme était, pour des configurations AFDX réalistes, de l'ordre de 20%, que chercher à le réduire demanderait des travaux conséquents, et que du point de vue industriel, l'intérêt était faible, j'ai considéré que chercher à améliorer la précision des résultats de calcul réseau sur les réseaux AFDX n'était plus une de mes priorités.

Une deuxième piste consistait à savoir analyser d'autres types de réseaux. Si l'AFDX fut une rupture pour le monde avionique, il n'est pas l'unique solution

1. Je passe sous silence ici la partie sur les réseaux de Petri : je ne considère pas qu'ils s'inscrivent dans un projet de recherche qui me fût propre. Sous la direction, ou pour mieux dire l'impulsion de mon directeur de thèse, Michel Diaz, j'ai appris à faire de la recherche, appris à lire des articles de recherche, à développer de nouveaux résultats, à écrire des articles pour communiquer ces résultats. Mais ce ne fut que l'ébauche d'un projet de recherche.

pour les systèmes aérospatiaux, et d'autres technologies pourraient aussi être embarquées. D'un point de vue théorique, le réseau AFDX est un réseau à priorité statique non préemptive entre files (ordonnancement FIFO/NP-SP). J'ai donc travaillé d'autres types d'ordonnancement :

— les politiques d'équilibrage de charge (PGS, P-GPS, DRR, WRR, IWRR),

— une politique à base de crédit (AVB),

— une politique à base d'échéances (EDF),

— les interactions entre systèmes synchrones et asynchrones (TTEthernet, ainsi qu'un usage commun de TSN [119], mélange d'AVB et de flux dirigés par le temps, que je n'ai pas présenté dans ce document car il ne s'agit que d'une combinaison de mécanismes déjà présentés),

— les interractions entre systèmes faiblement synchronisés (déphasages bornés),

— ainsi que les corrélations entre flux induit par l'ordonnanement des tâches émettrices de données.

Dans la quasi majorité des cas, je me suis assuré que l'approche pouvait être appliquée aux systèmes embarqués du monde aérospatial en les évaluant sur des cas d'étude représentatifs.

Je suis désormais convaincu que le calcul réseau peut modéliser quasiment toutes les politiques d'ordonnancement "raisonnables" pour les réseaux temps réels.

J'ai aussi voulu "consolider" le modèle : en discutant la représentation des systèmes réels par le modèle, en regardant les problèmes de continuité, en rajoutant quelques résultats qui viennent compléter les résultats de base (non présentés dans ce document [2]), ou en utilisant un assistant automatique de preuve (Coq ou Isabelle), pour prouver la correction des résultats et de leur implémentation.

J'ai aussi cherché à étendre le modèle du calcul réseau, en contribuant à rapprocher les modèles à base de nombre d'événements (ou de paquets) et à base de quantité de donnée, en passant par une relation de charge de travail associés à un événement (ou des tailles de paquets). Je suis en effet persuadé que les systèmes embarqués combinent des éléments (surtout le réseau) où les quantités de données sont l'élément clé des performances, alors que dans d'autres éléments (surtout les calculateurs et les passerelles), ce sont les nombres d'événements qui sont l'élément clé.

À ce travail sur les méthodes d'analyse de réseaux temps-réel, qui se concentrait sur les méthodes elles-mêmes, s'est aussi ponctuellement ajouté un travail

2. Comme le service résiduel d'ordonnancement arbitraire d'un service simple [20, § 7.3.2], l'interprétation que l'on pouvait donner à une courbe de service négative, le fait que la borne sur la politique FIFO présentée dans l'équation 4.1 était atteignable (*tight*, [20, Thm. 7.4].

d'analyse des performances des réseaux, qui se concentrait sur les réseaux cette fois-ci. J'ai ainsi travaillé sur les qualités de DRR comme mécanisme de partionnement de réseau, sur les gains et coûts associés à la préemption et à la fragmentation, sur les gains que l'on peut attendre d'un mécanisme de synchronisation faible, ou sur l'impact de détails de gestion de crédit en TSN (non présenté dans ce document, car il ne s'agit que de simulations d'un détail technique mis en lumière lors de sa modélisation en calcul réseau, [65]).

Chapitre 9

Activités de transfert hors de la communauté de recherche

Je voudrais aussi évoquer mon activité de transfert de connaissances hors du monde académique : vers le monde industriel et le grand public. Car si la partie la plus quantifiable et visible de mon travail consiste en des articles de recherche, une part importante de mon travail consiste à transférer les connaissances vers le monde industriel, dans le cadre de projets collaboratifs.

Comme cela ne donne pas lieu à publication, d'autant que les industriels aiment rester discrets sur certaines activités de recherche et développement, il n'y a pas d'indicateur public lié à cette activité. Je considère néanmoins que cela fait partie des missions d'un chercheur, et encore plus à l'ONERA.

Je vais donc présenter dans cette partie cette part de mon activité.

J'évoquerai aussi, au paragraphe 9.6, mes contributions à l'encyclopédie collaborative Wikipedia.

9.1. ADCN+

Le projet ADCN+ regroupait Airbus, l'IRIT, l'ONERA et le LAAS/CNRS pour une réflexion autour du réseau de données (Avionic Data Communication Network) de 2009 à 2012.

Ce projet fut bien sûr le support de travaux sur la théorie du calcul réseau pour les performances pire cas dans les réseaux AFDX, déjà présentés dans la première partie de ce manuscrit [42, 37, 38, 33, 32].

Mais ce fut aussi le lieu d'une réflexion plus globale sur l'architecture réseau de

l'avionique Airbus, avec de nombreux collègues de l'IRIT, principalement Jérome Ermont, Chritian Fraboul et Jean-Luc Scharbarg. En partant des passerelles entre le nouveau réseau AFDX et l'ensemble des bus CAN, ARINC 429, comme liens analogiques, ce fut le lieu de réflexions sur la pile protocolaire, implicite, dans l'avionique modulaire intégrée (*Integrated Modular Avionic*, IMA). Les questions d'interconnexion, de passerelles, donnèrent même lieu à la proposition d'une couche de niveau 3 dédiée à l'avionique, pensée pour l'interconnexion des bus avioniques et l'AFDX, avec prise en compte des spécificités des besoins des applications avioniques.

Ce travail de réflexion n'a pas eu, à ma connaissance, de suite. La définition d'une pile protocolaire explicite, spécifique au monde avionique, serait un choix de l'ensemble de la communauté industrielle avionique. Mais ce fut, il me semble, l'occasion pour un groupe d'ingénieurs chez Airbus de porter un nouveau regard sur leur architecture de communication.

9.2. PEGASE

Le projet PEGASE, financé par l'ANR de 2009 à 2013, dont j'ai eu le plaisir d'être le coordinateur, avait pour but de faire des avancées théoriques dans la théorie du calcul réseau dans l'optique d'une application aux réseaux embarqués avioniques et spatiaux.

Ce projet regroupait plusieurs composantes du groupe Thales (Thales Research and Technologies, Thales Alenia Space, Thales Avionics), 4 laboratoires (ONERA, ENS Cachan – Bretagne, LIP – Laboratoire de l'Informatique du Parallélisme, INRIA Rhône-Alpes) et une PME, RealTime-at-Work (RTaW).

Ce fut bien sûr le lieu de nombreux travaux théoriques, qui donnèrent lieu à publications de la part des différents partenaires. Puisqu'il s'agit ici de mon habilitation à diriger les recherches et non d'un bilan du projet, je ne mets ici que la liste des travaux dont je suis coauteur [47, 97, 43, 43, 45, 98, 99, 54, 46].

En terme de transfert, ce projet a initié une longue collaboration entre la société RealTime-at-Work et moi même. Je n'étais dans le projet pas le plus expert en calcul réseau, ou en réseau AFDX, mais je connaissais les deux et j'avais l'expérience de l'implantation d'un outil d'analyse de réseau AFDX basé sur la théorie du calcul réseau, NC-maude [33].

J'ai donc activement participé, principalement avec Anne Bouillard et Éric Thierry, à l'aide au développement d'un prototype de logiciel d'analyse de réseau AFDX à l'aide de la théorie du calcul réseau. Le travail consistait d'une part à homogénéiser les résultats épars de calcul réseau, à sélectionner les plus utiles, ainsi qu'à donner des conseils d'architecture du logiciel.

Cette base logicielle est le cœur de l'outil d'analyse RTaW-PEGASE actuellement commercialisé par la société.

9.3. Airbus Hélicoptères

En 2012, le pôle de compétitivité PEGASE a lancé un appel d'offre pour préparer la prochaine génération d'hélicoptère. Avec RTaW, nous devions aider Airbus hélicoptères (à l'époque Eurocopter) à préparer l'intégration d'un réseau ARINC 664 (AFDX) dans les hélicoptères.

Le travail réalisé a consisté en plusieurs points :

1. tout d'abord, une formation à la technologie AFDX, ainsi qu'aux technologies ARINC 429, CAN, ARINC 825,
2. mais surtout une formation au calcul réseau, et son application au calcul de bornes dans l'AFDX,
3. le tout, accompagné de la société RealTime-at-Work qui avait développé un prototype d'outil d'analyse de réseau AFDX en calcul réseau dans le cadre du projet ANR PEGASE (2009-2012).

Le travail autour du calcul réseau a été extrêmement intéressant : il ne s'agissait pas là de développer de nouveaux résultats, mais de faire une synthèse cohérente. Cela commença par extraire une sous-partie cohérente de la théorie du calcul réseau, en unifiant les notations, et en donnant les références exactes aux preuves (pas juste la référence des articles ou livres, mais dans chaque document, la référence exacte du théorème utilisé). Ensuite, il s'est agit de modéliser en calcul réseau un réseau conforme à la norme AFDX [11], en justifiant à partir des paragraphes de la norme chaque élément du modèle. C'est à cette occasion que j'ai découvert une lacune dans la norme : je n'y ai trouvé nulle part la spécification que la politique à l'intérieur d'une file d'attente soit de type FIFO... Pour terminer, il s'agissait d'expliciter un algorithme d'analyse de réseau correspondant au modèle réalisé, et de faire la preuve de correction de cet algorithme (preuve faite en langage naturel).

C'est d'ailleurs en présentant le calcul réseau chez Eurocopter que j'ai été interpellé par Cédric Mauclair sur les questions de continuité présentées au paragraphe 6.1.

L'ensemble des documents produits est propriété d'Eucopter, et je ne peux donc pas les présenter, sauf autorisation de leur part.

Le travail de formation aux technologies réseau fut plus surprenant : pour chaque technologie présentée, j'avais dans l'assistance 1 ou 2 ingénieur bien plus qualifié que moi sur la dite technologie, mais j'étais quasiment le seul à avoir la vision complète.

Par contre, la véritable plus-value apportée consista à faire la liste des enjeux liés à l'interconnexion de réseau AFDX et des bus CAN ou ARINC 429. L'équipe ressortit enthousiaste des discussions menées sur ce sujet, alors que je n'y avais apporté que des questions, mis le doigt sur des points durs, et donné très peu de pistes de réponses.

9.4. Expertise TTEthernet pour le CNES

La technologie TTEthernet [110] est une extension d'Ethernet dirigée par le temps (cf. paragraphe 7.2). Cette technologie devrait constituer le cœur du réseau de communication du lanceur Arianne 6. Le CNES nous a donc contactés en 2015 pour l'aider à monter en compétence sur cette technologie, non pas dans le sens d'une mise en place opérationnelle (les vendeurs de solution TTEthernet ont des services d'assistance très bien faits) mais plus dans le sens d'un travail critique.

Nous avons répondu à l'appel avec la société RealTime-at-Work. L'ONERA, avait en charge une réflexion sur les pannes (sûreté de fonctionnement) associées à cette technologie, menée par Pierre Bieber, ainsi que sur les preuves déjà faites dans la littérature scientifique, menée par Pierre Roux. La PME RealTime-at-Work avait en charge le développement d'un simulateur. Je faisais pour ma part le lien entre les différents intervenants, ainsi qu'une mise à niveau globale sur la technologie.

9.5. Réseau sur puce MPPA

Depuis 2006, l'ONERA a investi la problématique de l'utilisation des processeurs à plusieurs cœurs (multi ou pluri-cœurs) dans les systèmes embarqués temps-réel critiques.

On fait généralement la distinction entre processeurs multi-cœurs et pluri-cœurs en considérant leur modèle mémoire et leur système d'interconnexion. Un système multi-cœurs adopte une vision de mémoire partagée, avec cohérence de caches, entre les différents cœurs, des communications entre cœurs implicites, passant par la mémoire globale, et une interconnexion entre cœurs et avec la mémoire globale de type bus partagé. Un système pluri-cœurs considère au contraire que chaque processeur possède sa propre mémoire, privilégie les communications explicites entre cœurs, et s'il existe une mémoire globale partagée, le matériel n'offre pas de mécanisme de cohérence. L'interconnexion entre cœurs se fait à travers un réseau sur puce (*Network on Chip* – NoC).

J'ai commencé à m'intéresser à ces architectures dans le cadre d'un projet de recherche interne ONERA, avec comme support matériel la plate-forme expérimentale d'Intel SCC [67].

Mais c'est dans le cadre du projet CAPACITÉS, démarré fin 2014, que mes travaux ont réellement commencé. La plate-forme au cœur du projet CAPACITÉS était le MPPA de la société Kalray. Et si son NoC a été pensé pour offrir des garanties temps-réel [71], une partie de mon travail a consisté à étudier le fonctionnement de ce NoC, à vérifier que le modèle en calcul réseau qui en avait été fait était correct, à le comparer à d'autres approches [41] et à formuler des remarques, dont une part devait servir à concevoir la prochaine génération du

processeur.

9.6. Contributions Wikipedia

L'encyclopédie collaborative Wikipedia a, au moins dans sa version anglaise, atteint une maturité suffisante pour me servir de point d'entrée sur certains sujets.

J'ai aussi contribué à l'amélioration de certains articles, soit parce qu'ils sont le cœur de ma recherche, comme les articles Network Calculus, Wormhole switching, FLIT, soit parce qu'en faisant un état de l'art sur un domaine, j'en profite pour améliorer les articles associés, comme pour Fair queuing, Fairness measure, Weighted fair queueing, Weighted round robin, Deficit round robin, et Generalized processor sharing.

Je n'ose pas ajouter ces contributions à ma bibliographie, mais ils font partie de mes activités de transfert de connaissance.

Chapitre 10

Perspectives

Après ces années de recherche dans le domaine, je vais présenter quatre grandes perspectives qui me tiennent à cœur, deux partiellement présentées dans ce manuscrit, et deux autres sur lesquelles je n'ai pas encore eu le temps de travailler.

Il y aura bien sûr aussi dans les années futures la continuation de travaux déjà commencés :

— la modélisation de nouvelles politiques de service, de leurs interactions : le nouveau réseau TSN a défini de nombreux mécanismes, et si des bornes sur les délais de chaque mécanisme pris isolément sont connues, ce n'est souvent pas le cas dès lors que plusieurs mécanismes sont utilisés en même temps sur un port de sortie. Il reste de nombreux résultats à découvrir dans ce domaine.

— l'analyse de mécanismes temps réels : s'il arrive qu'un industriel ait choisi une technologie temps réel et son mécanisme associé, son besoin n'étant alors que de pouvoir vérifier des garanties de performances, il arrive aussi qu'il ait à choisir entre plusieurs technologies, ou entre plusieurs mécanismes offerts par une même technologie. Il y a alors un travail d'évaluation du mécanisme lui même à réaliser. Par exemple, je travaille en ce moment, avec Damien Guidolin en Jean-Yves Le Boudec à comparer le mécanisme CQF (*Cyclic Queuing and Forwarding*, [10]) et la politique FIFO.

Je ne présente pas ce type de travaux comme des perspectives, dans le sens où ils ne présentent ni défi, ni rupture, même s'ils ne manquent pas d'intérêt.

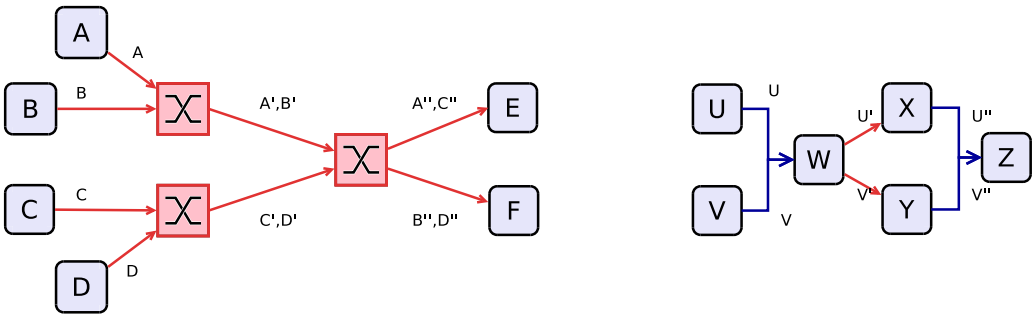


Figure 35 – *Topologie avec corrélation de flux.*

10.1. Propagation de corrélations entre flux

Les travaux menés par Hugo Daigmorte dans la première partie de sa thèse (la définition d'un modèle en calcul réseau permettant d'analyse des systèmes faiblement synchronisés entre eux) ouvrent la voie à des travaux plus généraux. En prenant un peu de recul, ces travaux généralisent les notions de groupage présentés au paragraphe 4.1.2. Étant donné un ensemble de flux $A_1, \ldots, A_n$, on peut avoir une courbe d'arrivée α_i pour chaque flux A_i mais aussi une courbe d'arrivée $\alpha_{1,\ldots,n}$ pour la somme des flux $A_1 + \cdots + A_n$ telle que $\alpha_{1,\ldots,n}$ soit "plus petit" que $\alpha_1 + \cdots + \alpha_n$, ç-à-d formellement

$$\exists t : \alpha_{1,\ldots,n}(t) < \alpha_1(t) + \cdots + \alpha_n(t) \tag{10.1}$$

Avec le groupage, on prenait simplement $\alpha_{1,\ldots,n} = \lambda_R \wedge (\alpha_1 + \cdots + \alpha_n)$, avec λ_R le débit du lien traversé (section 4.1.2). Avec la synchronisation faible (section 7.2, le calcul de $\alpha_{1,\ldots,n}$ se faisait à partir des fenêtres d'émission des flux et le déphasage entre les horloges locales. Dans la prise en compte de l'ordonnancement des tâches (section 5.4), ce calcul se faisait en fonction des dates de réveil et de terminaison des tâches.

Prenons un exemple : considérons la topologie de la Figure 35, et supposons que les flux A, B, C, D soient des flux dirigés par le temps, et que l'on connaisse un déphasage entre les horloges locales. On peut calculer une courbe d'arrivée $\alpha_{A,B}$ ou $\alpha_{A,C}$ pour capturer la corrélation entre ces flux. Mais une fois traversé un premier commutateur, la corrélation est modifiée, et on ne sait plus modéliser l'information de corrélation initiale dans le port menant au calculateur E.

Mais même le simple groupage peut être perdu : considérons trois calculateurs U, V, W reliés par un bus, et deux flux de données U, V convergeant vers W, on peut simplement dire que $\alpha_{U,V}$ qui entre dans le calculateur W est $\alpha_{U,V} = \lambda_R \wedge (\alpha_U + \alpha_V)$. Si ensuite chaque flux se propage, U vers un calculateur X, devenant U', et parallèlement, V part vers Y devenant V', puis tous deux convergent vers un calculateur Z. Dans ce cas aussi, lorsque l'ensemble converge vers Z, la corrélation initiale est perdue.

Nous avons, avec Geoffrey Nelissen, de l'université d'Eindhoven (*Eindhoven University of Technology*) quelques résultats préliminaires en ce sens, qui me semblent ouvrir la voie.

J'aimerais pour ma part utiliser ces résultats pour étudier la question des applications synchrones sur un réseau asynchrone. En effet, un réseau dirigé par le temps dédie des fenêtres temporelles à chaque message. Pour obtenir une faible latence, il faut donc que les données soient produites "juste avant" la fenêtre de sortie du calculateur. Donc que les applications soient ordonnancées par rapport à une horloge globale. Mais si les données sont émises de façon à respecter un ordonnancement global, qui est construit pour éviter les contentions, même une politique asynchrone de la part du réseau va bénéficier de ces flux d'entrée sans contention. C'est ce qui a été montré dans les travaux sur la synchronisation faible pour un bus, et que j'aimerais pouvoir étudier de manière quantitative sur un réseau.

10.2. Un modèle intégrant données et tâches

J'ai présenté au paragraphe 4.5 un modèle liant quantité de données et nombre de paquets. Je pense que ce modèle mérite d'être développé et éprouvé, car il a des applications dans plusieurs domaines.

10.2.1. Domaines d'application du modèle

Données générées par les tâches En effet, un réseau transfère des données, produites par des applications (des tâches), regroupées sous forme de paquets. L'ordonnancement de tâches est généralement analysé avec des méthodes dites de temps de réponse (*response time analysis*), et nous avons montré dans [38], [52] (présenté au paragraphe 5.4) que des informations sur l'ordonnancement des taches permettent d'avoir une meilleure information sur le trafic généré, donc en calcul réseau, de meilleures courbes d'arrivées, et donc de meilleures bornes.

Comportement réseau dépendant des paquets En ce qui concerne le réseau, il existe de nombreux cas où les performances ne dépendent pas seulement de la quantité de données mais aussi du découpage en paquets.

D'une part, il existe des politiques d'ordonnancement réseau qui comptent des paquets, indépendemment de leurs tailles (WRR principalement, présenté au paragraphe 5.1.3), et les analyses actuelles ne prennent en compte qu'une taille minimale ou maximale pour les paquets d'un flux. Mais quand un flux est l'agrégation de plusieurs flux élémentaires, chaque flux élémentaire porte sa propre contrainte, et le modèle actuel doit faire comme si tous les flux avaient les mêmes tailles minimales et maximales.

Les réseaux TSN manipulent explicitement des paquets dans le cadre de mécanismes de dédoublement/consolidation définis pour augmenter la fiabilité du réseau (mécanisme "Frame Replication and Elimination for Reliability", FRER, [8]). Une analyse de l'impact temporel de ces mécanismes nécessitera une modélisation de la notion de paquet.

D'autre part, il existe un "angle mort" de l'analyse réseau : les passerelles entre réseaux de technologies différentes. Lorsque deux calculateurs reliés chacun à des réseaux de technologies différentes (CAN et Ethernet par exemple) doivent communiquer entre eux, il faut qu'un élément réseau, la passerelle, fasse la conversion. Il s'agit alors de décomposer le paquet en deux parties : la "charge utile" (*payload*), qui contient les informations à transmettre, et l'"enrobage" (*overhead*) spécifique à chaque technologie réseau. L'enrobage étant en général de taille fixe pour une technologie donnée, mais spécifique à chaque technologie. En recevant un paquet de taille d, une passerelle simple va émettre pour chaque paquet reçu un nouveau paquet dont la taille est $d + (e_s - e_e)$, avec où e_e est la taille de l'enrobage dans la technologie en entrée et e_s celle correspondante à la sortie. De manière plus générale, pour évaluer correctement la quantité de données émises par une passerelle, il faut évaluer finement à la fois le nombre de paquets et la quantité de données reçues. C'est encore plus vrai avec des stratégies de passerelle plus avancées, lorsqu'un message est trop grand pour être retransmis en un seul paquet et doit être découpé (ce que l'on appelle "fragmentation"), ou lorsque l'on décide de regrouper plusieurs (petits) paquets en un seul (plus gros), ce que l'on appelle "multiplexage". Mon expérience personnelle est que c'est un réel besoin industriel (auquel j'ai été sensibilisé lors du projet ADCN+, présenté au paragraphe 9.1), assez peu traité dans la littérature scientifique, ce qui amène les industriels à mettre en place des solutions sur mesure, pas forcément efficaces, à "réinventer la roue" à chaque fois, à se reporter sur une expertise industrielle et non sur un corpus scientifique documenté et outillé.

Tâches activées par le réseau Dernier point, symétrique de la question des "données générées par les tâches", celle des tâches activées par le réseau.

Les modèles d'ordonnancement des tâches supposent que les tâches sont activées (on dit souvent "réveillées") suivant un certain modèle (périodique, périodique avec décalage [74], avec des dépendances [57]) mais l'analyse est faite en considérant que ces activations sont indépendantes des dates d'arrivée des messages, ou en prenant des délais réseaux simplifiés [85]. Or il existe des contextes où les tâches sont activées par la réception d'un message. Pour faire des analyses d'ordonnancement plus précises, il faudrait que l'on puisse, à partir des caractéristiques des flux entrant dans un calculateur, évaluer finement le nombre de messages reçus dans un intervalle de temps.

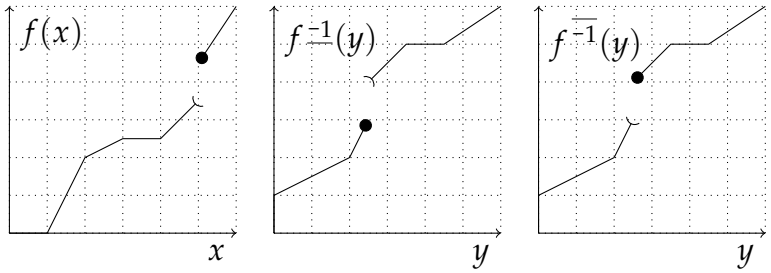


Figure 36 – *Pseudo-inverses : illustration sur un exemple*

10.2.2. Perspectives de travaux

A la suite des premiers travaux sur les tailles de paquets présentés dans [22], j'ai travaillé sur un support théorique systématique d'un tel modèle, et sur un premier lien avec un modèle orienté tâche, travaux présentés dans le paragraphe 4.5.

Le travail sur ce modèle pourrait s'organiser suivant deux pistes, une piste purement théorique et une liée aux domaines d'application.

Travaux purement théorique D'un point de vue théorique, il me semble qu'il faudrait faire un lien avec les travaux autour du calcul réseau et de l'algèbre max-plus [93]. Alors que le modèle que nous avons proposé dans [50, 51] s'intéresse aux liens entre une fonction $A(t)$ qui associe la quantité de données reçues jusqu'à un instant t (venant du calcul réseau) et une fonction $E(t)$ qui associe le nombre d'événements vus jusqu'à un instant t (venant du modèle CPA), une vue duale dans max-plus s'intéresse à une fonction $T_A(d)$ qui donne la date associée à la réception d'une quantité de données d, et il existait dans les articles CPA une fonction implicite, T_E qui donnait la date associée à la réception d'une quantité de données d. Ces fonctions peuvent se déduire les unes des autres à l'aide des opérateurs de pseudo-inverse, $f^{\underline{-1}}$ et $f^{\overline{-1}}$, donnés dans les équations qui suivent et illustrés à la Figure 36 :

$$T_A = A^{\overline{-1}} \qquad A = T_A{}^{\underline{-1}}$$
$$T_E = E^{\underline{-1}} \qquad E = T_E{}^{\overline{-1}}$$
$$P(A) = E$$

On considère généralement en calcul réseau que certaines opérations se font plus facilement en travaillant avec A et d'autres avec T_A. Lorsque l'on dessine l'ensemble des relations sur un diagramme (cf. figure 37) on peut d'une part être surpris par la différence dans l'usage des pseudo-inverses entre A et T_A d'une part, et E et T_E d'autre part. Il se pourrait que les deux ne représentent pas exactement la même chose, ou que ce soit une erreur de frappe dans un article.

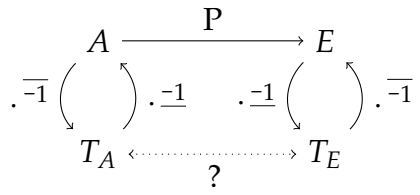


Figure 37 – *Relations entre décompte de données (A), de paquets (E), et dates d'arrivées (T_A, T_E)*

Mais il faudrait surtout regarder s'il existe une relation en T_A et T_E qui fasse apparaître les tailles de paquets.

Évaluation sur cas d'études Les résultats existant sur ce modèle n'ont pas encore été évalués sur des cas d'étude. Un petit cas d'étude à 2 flux a été présenté dans [51], qui montrait un gain possible de la méthode. Mais cela reste à faire sur des exemples plus conséquents. En particulier, des résultats préliminaires sur l'agrégation de flux ont montré des résultats assez décevants.

Mon expérience de recherche est que les résultats purement théoriques permettent de forger un outil théorique plus robuste, mais que pour obtenir un outil plus efficace, c'est à dire dans notre cas donnant des meilleures bornes, il faut confronter sur des cas d'étude les résultats obtenus et les résultats attendus, soutenus par l'expertise du domaine.

Il faudrait donc reprendre les domaines d'application présentés au paragraphe 10.2.1, prendre des systèmes où l'on espère un gain du nouveau modèle, mais assez petits pour être traités et interprétés avec des outils simples, et confronter les résultats obtenus avec une intuition du domaine.

10.3. Calcul de configuration respectant des exigences

Le calcul réseau, à ce jour, permet, étant donné un ensemble de flux (avec leurs caractéristiques), un réseau (avec ses algorithmes d'ordonnancement et leurs paramètres) et un routage, de calculer des bornes sur l'occupation mémoire des serveurs et les délais des flux.

Mais d'où viennent les paramètres, le routage ? Le besoin initial de l'ingénieur ressemble plutôt à celui ci : étant donné un ensemble d'applications, étant donné les besoins de communications entre applications, étant donné une architecture matérielle (un ensemble de calculateurs, un réseau d'interconnexion), comment placer les applications sur les calculateurs et comment configurer le réseau (paramètres et routes) pour que toutes les contraintes soient respectées ? Ces contraintes sont diverses : certaines applications doivent être placées sur tel ou tel calculateur, car seul ce calculateur a certaines fonctionnalités matérielles (connexion à un senseur ou un actuateur), certaines applications ne doivent pas être placées sur

le même calculateur (pour que la panne du calculateur ne fasse pas perdre les deux fonctions), les performances d'un calculateur (en place mémoire comme en puissance de calcul) doivent être suffisantes pour satisfaire toutes les applications qui y sont placées, et les latences de communications doivent être satisfaites.

Actuellement, le problème est plutôt traité en deux phases : une première qui ignore le réseau, puis une seconde qui paramètre le réseau et vérifie qu'il satisfait les contraintes de latence et d'occupation mémoire. Et si la configuration ne satisfait pas les exigences, il faut modifier la configuration du réseau, voire le placement des applications, en espérant que la nouvelle configuration respecte les contraintes.

Ceci est dû au fait que les algorithmes qui permettent de calculer les bornes liées au réseau s'intègrent mal dans les solveurs de contraintes qui servent à faire le placement des applications.

Un enjeu consiste donc à résoudre simultanément les deux problèmes. Cela passera sûrement par le calcul de bornes réseau un peu moins précises : fournir des approximations des expressions des bornes compatibles avec les types d'expression admissible par les solveurs. Une telle approche a été initiée dans [73, 71].

10.4. Intégration fine des délais et des lois de commande

Les systèmes temps-réel critiques pour lesquels les concepteurs ont besoin de garanties fermes sur les délais sont généralement des systèmes "cyber-physiques" : des systèmes physiques qui sont partiellement (ou totalement) commandés par un système logiciel. Le principe de la commande, en peu de mots, consiste si un système est dans un état x à un instant t (par exemple, une voiture roule sur la voie de droite), et qu'on désire l'amener à un état x' dans un futur "proche" (par exemple, rouler sur la voie de gauche), on va lui appliquer une commande $u(t) = f(x(t), t, x')$ (tourner le volant à gauche, mais pas trop fort) jusqu'à ce que l'objectif soit atteint. Dans un système cyber-physique, l'état à un instant t_i est mesuré par un capteur, reçu par le calculateur à un instant $t_i + d_i^e$. A l'instant $t_i + d_i^e$, le calculateur connaît donc la valeur $x(t_i)$, mais peut ignorer t_i lui même. Si on considère le temps de calcul de la commande comme négligeable, une commande $u(t_i + d_u^e) = f(x(t_i), t_i + d^e, x')$ sera envoyée et reçue (donc appliquée) à un instant $t_i + d_i^e + d_i^c$.

Comme d_i^e et d_i^c sont inconnus lors de la conception du système, et variables pour chaque message, ils sont, suivant les cas, considérés comme négligeables (nuls), constants, ou uniformément répartis, respectivement, dans des intervalles $[\underline{d}_i^e, \overline{d}_i^e]$ et $[\underline{d}_i^c, \overline{d}_i^c]$. Mais dans un réseau réel, les délais successifs liés à un message ne sont ni constant, ni uniformément répartis entre deux valeurs extrêmes.

Il a été suggéré dans [55, 7] qu'en considérant une borne $\Delta(n)$ telle que

$$\forall i \in \mathbb{N} : \sum_{k=i}^{k=i+n} d_k^e + d_k^c \leq \Delta(n) \tag{10.2}$$

on peut déduire plus d'informations sur les propriétés de la loi de commande et du système commandé.

Or, les recherches dans le domaine des temps de réponse se sont focalisées jusqu'à ce jour sur les meilleurs et pires temps de réponse. Mais les notions de courbes d'arrivée et de services sont plus riches que cela, et permettent de capturer des bornes sur des fenêtres de temps.

Il m'apparaît donc prometteur de travailler conjointement avec des automaticiens pour définir avec eux le type de contrat sur les délais qui permettraient d'améliorer la qualité de l'analyse de la commande, et de voir ce qui dans le calcul réseau permettrait de valider ce type de contrat.

Bibliographie

[1] Software considerations in airborne systems and equipment certification. Technical Report DO-178B, RTCA, December 1 1992.

[2] Virtual Bridged Local Area Networks Amendment 12 : Forwarding and Queuing Enhancements for Time-Sensitive Streams. Technical Report IEEE 802.1Qav, IEEE, 2010.

[3] Virtual Bridged Local Area Networks Amendment 14 : Stream Reservation Protocol (SRP). Technical Report IEEE 802.1Qat, IEEE, 2010.

[4] Local and metropolitan area networks - Audio Video Bridging (AVB) Systems. Technical Report IEEE 802.1BA, IEEE, 2011.

[5] Local and metropolitan area networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Network. Technical Report IEEE 802.1AS, IEEE, 2011.

[6] Media access control (MAC) bridges and virtual bridged local area networks – amendment 18 : Enhanced transmission selection for bandwidth sharing between traffic classes. Technical Report IEEE 802.1Qaz, IEEE, September, 30 2011.

[7] *Event density analysis for event triggered control systems*, Grenoble, France, May 2013.

[8] Ieee draft standard for local and metropolitan area networks – frame replication and elimination for reliability. Technical Report 802.1CB, IEEE, September 2017.

[9] Ieee standard for local and metropolitan area networks – bridges and bridged networks – amendment 26 : Frame preemption. IEEE Standard 802.1Qbu, 2016.

[10] Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 29 : Cyclic queuing and forwarding. IEEE Standard 802.1Qch, IEEE, 2017.

[11] AEEC. Arinc 664 p7-1 : Aircraft data network, part 7, avionics full-duplex switched ethernet network. Technical report, Airlines Electronic Engineering Committee, september 2009.

[12] AEEC. Arinc 825-2 : General standardization of can (controller area network) bus protocol for airborne use. Technical report, Airlines Electronic Engineering Committee, July 2011.

[13] Amos Albert. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded world*, 2004 :235–252, 2004.

[14] Sanjoy Baruah, Deji Chen, Sergey Gorinsky, and Aloysius Mok. Generalized multiframe tasks. *Real-Time Systems*, 17 :5–22, 1999.

[15] Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *In Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190. IEEE Computer Society Press, 1990.

[16] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Trans. Industrial Informatics*, 6(4) :521–533, 2010.

[17] B. Berthomieu, M. Boyer, and M. Diaz. Les réseaux de Petri temporels. In Diaz [69], pages 161–198.

[18] Luca Bisti, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. DEBORAH : a tool for worst-case analysis of FIFO tandems. In T. Margaria and B. Steffen, editors, *Proc. of the 4th Int. Symp. On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010)*, LNCS. Springer, 2010.

[19] Anne Bouillard. Composition of service curves in network calculus. In *Proceedings of the 1st International Workshop on Worst-Case Traversal Time (WCTT'2011)*, WCTT '11, pages 35–42, 2011.

[20] Anne Bouillard, Marc Boyer, and Euriell Le Corronc. *Deterministic Network Calculus – From theory to practical implementation*. Number ISBN : 978-1-119-56341-9. Wiley, 2018.

[21] Anne Bouillard, Nadir Farhi, and Bruno Gaujal. Packetization and aggregate scheduling. Technical Report 7685, INRIA, 2011.

[22] Anne Bouillard, Nadir Farhi, and Bruno Gaujal. Packetization and packet curves in network calculus. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012. Invited Paper.

[23] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Service curves in Network Calculus : dos and don'ts. Research Report RR-7094, INRIA, 2009.

[24] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. Technical Report 7012, INRIA, 2009.

[25] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Comparison of different classes of service curves in network calculus. In *Proc. of the 10th International Workshop on Discrete Event Systems (WODES 2010)*, Technische Universität Berlin, August 30 - September 1 2010.

[26] Anne Bouillard and Giovanni Stea. Exact worst-case delay for FIFO-multiplexing tandems. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012.

[27] Anne Bouillard and Giovanni Stea. Exact worst-case delay for FIFO-multiplexing feed-forward networks. *IEEE/ACM Transactions on Networking*, 2014.

[28] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. Rapport de recherche 6094, INRIA, January 2007.

[29] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1) :3–49, october 2008. http ://www.springerlink.com/content/ 876x51r6647r8g68/.

[30] Loïc Boyer, Marc ad Fejoz and Stephan Merz. Proof-by-instance for embedded network design - from prototype to tool roadmap. In *Proc. of the 6th Embedded Real Time Software and System Congress (ERTSS 2014)*, 2014.

[31] Marc Boyer. Translation from timed Petri nets with interval on transitions to interval on places (with urgency) – extended abstract. In *Proceedings of the first Workshop on Theory and Practice of Timed Systems (A satelite event of ETAPS 2002)*, volume 65 of *Electronic Notes in Theoretical Computer Science*, Grenoble, France, April 2002. Elsevier Science.

[32] Marc Boyer. Half-modeling of shaping in FIFO net with network calculus. In *Proc. of the 18th International Conference on Real-Time and Network Systems (RTNS 2010)*, Toulouse, France, November 4-5 2010.

[33] Marc Boyer. NC-maude : a rewriting tool to play with network calculus. In T. Margaria and B. Steffen, editors, *Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010)*, LNCS. Springer, 2010.

[34] Marc Boyer. NC-maude : maude for computation of worst bounds on real-time (embedded) networks. Technical Report 1/16417, ONERA, 2010.

[35] Marc Boyer and Michel Diaz. Non equivalence between time Petri nets and time stream Petri nets. In Peter Bucholz and Manuel Silva, editors, *Proceedings of 9th IEEE International Workshop on Petri Nets and Performance*

Modeling (PNPM'99), pages 198–207, Zaragoza, Spain, September 1999. IEEE Computer Society.

[36] Marc Boyer and Michel Diaz. Multiple enabledness of transitions in time Petri nets. In *Proceeding of the 9th IEEE International Workshop on Petri Nets and Performance Models*, Aachen, Germany, September 11–14 2001. IEEE Computer Society.

[37] Marc Boyer and David Doose. Collaboration entre méthode d'ordonnancement et calcul réseau. In *Actes des 2èmes Journées du GdR Génie de la programmation et du logiciel (GdR GPL 2010)*, Pau, France, 10-12 mars 2010.

[38] Marc Boyer and David Doose. Combining network calculus and scheduling theory to improve delay bound. In *Proc of the 20th International Conference on Real-Time and Network Systems (RTNS 2012)*, Pont á Mousson, France, November 8-9 2012.

[39] Marc Boyer, Guillaume Dufour, and Luca Santinelli. Continuity for network calculus. In *Proc of the 21th International Conference on Real-Time and Network Systems (RTNS 2013)*, pages 235–244, Sophia Antipolis, France, October 16-18 2013. ACM.

[40] Marc Boyer and Christian Fraboul. Tightening end to end delay upper bound for AFDX network with rate latency FCFS servers using network calculus. In *Proc. of the 7th IEEE Int. Workshop on Factory Communication Systems Communication in Automation (WFCS 2008)*, pages 11–20. IEEE, May 21-23 2008.

[41] Marc Boyer, Amaury Graillat, Benoît Dupont De Dinechin, and Jörn Migge. Bounding the delays of the MPPA network-on-chip with network calculus : models and benchmarks. *Performance Evaluation*, July 2020.

[42] Marc Boyer, Laurent Jouhet, and Anne Bouillard. Notations pour le calcul réseau. *Journal Européen des Systèmes Automatisés*, 43(7–9) :921–935, 2009. Actes du 7ème colloque francophone sur la Modelisation des Systemes Reactifs (MSR'09).

[43] Marc Boyer, Jörn Migge, and Marc Fumey. PEGASE, a robust and efficient tool for worst case network traversal time. In *Proc. of the SAE 2011 AeroTech Congress & Exhibition*, Toulouse, France, 2011. SAE International.

[44] Marc Boyer, Jörn Migge, and Nicolas Navet. An efficient and simple class of functions to model arrival curve of packetised flows. In *Proc. of the 1st Int. Workshop on Worst-Case Traversal Time (WCTT'2011)*, pages 43–50, New York, NY, USA, novembre 2011. ACM.

[45] Marc Boyer, Nicolas Navet, and Marc Fumey. Experimental assessment of timing verification techniques for AFDX. In *Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems*, Toulouse, France, February 2012.

[46] Marc Boyer, Nicolas Navet, Marc Fumey, Jörn Migge, and Lionel Havet. Combining static priority and weighted round-robin like packet scheduling in AFDX for incremental certification and mixed-criticality support. In *Proc. of the 5th European Conference for Aeronautics and Space Sciences – Real Time Avionics and Networks Session (EUCASS 2013)*, Munich, Germany, July 1st–5th 2013.

[47] Marc Boyer, Nicolas Navet, Xavier Olive, and Eric Thierry. The PEGASE project : precise and scalable temporal analysis for aerospace communication systems with network calculus. In T. Margaria and B. Steffen, editors, *Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010)*, LNCS. Springer, 2010.

[48] Marc Boyer and Olivier Henri Roux. Comparison of the expressiveness of arc, place and transition time Petri nets. In Jetty Kleijn and Alex Yakovlev, editors, *Proceedings of the 28th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (ATPN)*, volume 4546 of *LNCS*, Siedlce, Poland, June 2007. Springer-Verlag.

[49] Marc Boyer and Olivier Henri Roux. On the compared expressiveness of arc, place and transition time petri nets. *Fundamenta Informaticae*, 88(3) :225–249, January 2008.

[50] Marc Boyer and Pierre Roux. A common framework embedding network calculus and event stream theory. working paper / hal-01311502, May 2016.

[51] Marc Boyer and Pierre Roux. Embedding network calculus and event stream theory in a common model. In *Proc. of the 21st IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2016)*, Berlin, Germany, September 2016.

[52] Marc Boyer, Luca Santinelli, Nicolas Navet, and Marc Migge, Jörn annd Fumey. Integrating end-system frame scheduling for more accurate afdx timing analysis. In *Proc. of the 7th Int. Congress on Embedded Real Time Software and Systems (ERTSS 2014)*, Toulouse, France, February 2014.

[53] Marc Boyer and Mihaela Sighireanu. Synthesis and verification of constraints in the pgm protocol. In *FME 2003 : Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 264–281. Springer Berlin / Heidelberg, 2003. 10.1007/978-3-540-45236-2_16.

[54] Marc Boyer, Giovanni Stea, and William Mangoua Sofack. Deficit round robin with network calculus. In *Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012)*, Cargese, France, October, 9–12 2012.

[55] Tobias Bund and Frank Slomka. A delay density model for networked control systems. In *Proc. of the 21st Int. Conf. on Real-Time Networks and Systems (RTNS '13)*,, RTNS '13, pages 205–212, New York, NY, USA, 2013. ACM.

[56] Donald T. Campbell. Assessing the impact of planned social change. *Evaluation and Program Planning*, 2(1) :67 – 90, 1979.

[57] Louis-Claude Canon, Emmanuel Jeannot, Rizos Sakellariou, and Wei Zheng. Comparative evaluation of the robustness of dag scheduling heuristics. In *Grid Computing*, pages 73–84. Springer, 2008.

[58] Antonio Cerone and Andrea Maggiolo-Schettini. Time-bases expressivity of time Petri nets for system specification. *Theoretical Computer Science*, 216(1–2) :1–53, March 1999.

[59] Cheng-Shang Chang. *Performance Guarantees in communication networks*. Telecommunication Networks and Computer Systems. Springer, 2000.

[60] Hussein Charara. *Évaluation des performances temps réel de réseaux embarqués avioniques*. Thèse de doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, novembre 2007.

[61] Hussein Charara, Jean-Luc Scharbarg, and Christian Fraboul. Analysing end-to-end delays on an AFDX network by simulation. In *Communication Systems and Networks (IASTED-CSN), Palma de Mallorca, Spain, 28/08/2006-30/08/2006*, ISBN 0-88986-606-6, pages 171–176, http ://www.ieee.org/, 2006. IEEE.

[62] Hemannt. M. Chaskar and Upamanyu Madhow. Fair scheduling with tunable latency : a round-robin approach. *IEEE/ACM Transactions on Networking*, 11(4) :592–601, Aug 2003.

[63] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude – A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer, 2007.

[64] Hugo Daigmorte and Marc Boyer. Traversal time for weakly synchronized can bus. In *Proc. of the 24th Int. Conf. on Real-Time Networks and Systems (RTNS 2016)*, Brest, France, 19-21th October 2016.

[65] Hugo Daigmorte and Marc Boyer. Impact on credit freeze before gate closing in cbs and gcl integration into tsn. In *Proc. of the 27th Int. Conf. on Real-Time Networks and Systems (RTNS 2019)*, Toulouse, France, November 2019.

[66] Hugo Daigmorte, Marc Boyer, and Jörn Migge. Reducing can latencies by use of weak synchronization between stations. In *Proc. of the 16th international CAN Conference (iCC 2017)*, Nuremberg, Germany, March 2017. CIA.

[67] Bruno d'Ausbourg, Marc Boyer, Eric Noulard, and Claire Pagetti. Deterministic execution on many-core platforms application to the scc. In *Proc. of the 4th symposium of the Many-core Applications Research Community (MARC'11b)*, 2011.

[68] Robert Davis, Alan Burns, Reinder Bril, and Johan Lukkien. Controller area network (CAN) schedulability analysis : Refuted, revisited and revised. *Real-Time Systems*, 35 :239–272, 2007. 10.1007/s11241-007-9012-7.

[69] Michel Diaz, editor. *Les réseaux de Petri – Modèles fondamentaux*. Number ISBN 2-7462-0250-6 in IC2 – Informatique et systèmes d'information. Hermes, 2001.

[70] Jonas Diemer, Philip Axer, and Rolf Ernst. Compositional performance analysis in python with pyCPA. *Proc. of 3rd Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2012)*, July 2012.

[71] Benoît Dupont de Dinechin and Amaury Graillat. Network-on-chip service guarantees on the kalray MPPA-256 bostan processor. In *Proc. of the 2nd Int. Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems (AISTECS 2017)*, Stockholm, Sweden, January 2017.

[72] Fabrice Frances, Christian Fraboul, and Jérôme Grieu. Using network calculus to optimize AFDX network. In *Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06)*, Toulouse, January 2006.

[73] Antonio Frangioni, Laura Galli, and Giovanni Stea. Optimal joint path computation and rate allocation for real-time traffic. *The Computer Journal*, 58(6) :1416–1430, 2014.

[74] MR Garey, David S. Johnson, Barbara B. Simons, and Robert Endre Tarjan. Scheduling unit–time tasks with arbitrary release times and deadlines. *SIAM Journal on Computing*, 10(2) :256–269, 1981.

[75] Jean-Philippe Georges, Thierry Divoux, and Éric Rondeau. Network calculus : application to switched real-time networking. In *Proc. of the 5th Int. ICST Conf. on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11, pages 399–407, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[76] Nicos Gollan, Frank A. Zdarsky, Ivan Martinovic, and Jens B. Schmitt. The DISCO Network Calculator. In *14th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2008)*, Dortmund, Germany, March 2008. GI/ITG.

[77] Michel Gondran and Michel Minoux. *Graphes, dioïdes et semi-anneaux : Nouveaux modèles et algorithmes*. Editions Tec & Doc, 2001.

[78] Charles Goodhart. Problems of monetary management : The U.K. experience. Technical report, Reserve Bank of Australia, Sydney, 1975.

[79] Jérôme Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. PhD thesis, Institut National Polytechnique de Toulouse (INPT), Toulouse, Juin 2004.

[80] Jérôme Grieu, Fabrice Frances, and Christian Fraboul. Preuve de déterminisme d'un réseau embarqué avionique. In *Actes du Colloque Francophone sur l'Ingenierie des Protocoles (CFIP 2003)*, Paris, Octobre 2003.

[81] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System level performance analysis - the symta/s approach. *IEEE Proceedings on Computers and Digital Techniques*, 152(2) :148 – 166, march 2005.

[82] IEEE. Ieee standard for ethernet amendment 5 : Specification and management parameters for interspersing express traffic. IEEE Standard 802.3br, 2016.

[83] ISO. Road vehicles – controller area network (CAN) – part 1 : Data link layer and physical signalling. Technical Report ISO 11898-1, International Organization for Standardization, 2003.

[84] Manolis Katevenis, Stefanos Sidiropoulos, and Costas Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communications*, 9(8) :1265–1279, 1991.

[85] Dzmitry Kliazovich, Johnatan E Pecero, Andrei Tchernykh, Pascal Bouvry, Samee U Khan, and Albert Y Zomaya. CA-DAG : Communication-aware directed acyclic graphs for modeling cloud computing applications. In *2013 IEEE sixth international conference on cloud computing*, pages 277–284. IEEE, 2013.

[86] Leonie Köhle, Borislav Nikolić, and Marc Boyer. Increasing accuracy of timing models : From cpa to cpa+. In *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Florence, Italy, March 2019.

[87] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus*, volume 2050 of *LNCS*. Springer Verlag, 2001. http ://lrcwww.epfl.ch/PS_files/NetCal.htm.

[88] L. Lenzini, E. Mingozzi, and G. Stea. End-to-end delay bounds in FIFO-multiplexing tandems. In Peter Glynn, editor, *Proc. of the 2nd International Conference on Performance Evaluation Methodologies and Tools (ValueTool07*, Nantes, France, October 23-25 2007. ICST.

[89] Luciano Lenzini, Linda Martorini, Enzo Mingozzi, and Giovanni Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree network. *Performance Evaluations*, 63 :956–987, 2005.

[90] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Delay bounds for FIFO aggegates : a case study. *Computer Communications*, 28 :287–299, 2004.

[91] Xiaoting Li, Olivier Cros, and Laurent George. The trajectory approach for AFDX FIFO networks revisited and corrected. In *Proc. of the 20th Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA'2014)*, Chongqing, China, 2014.

[92] Jorg Liebeherr, Yashar Ghiassi-Farrokhfal, and Almut Burchard. On the impact of link scheduling on end-to-end delays in large networks. *IEEE Journal on Selected Areas in Communications*, 29(5) :1009–1020, May 2011.

[93] Jörg Liebeherr. Duality of the max-plus and min-plus network calculus. *Foundations and Trends in Networking*, 11(3-4) :139–282, 2017.

[94] Etienne Mabille, Loïc Boyer, Marc ad Fejoz, and Stephan Merz. Certifying network calculus in a proof assistant etienne. In *Proc. of the 5th European Conference for Aeronautics and SpaceSciences (EUCASS)*, 2013. Real Time Avionics and Networks Session.

[95] Etienne Mabille, Loïc Boyer, Marc ad Fejoz, and Stephan Merz. Towards certifying network calculus. In *Proc. of the 4th Conference on Interactive Theorem Proving (ITP 2013)*, Rennes, France, July 2013.

[96] Cláudio Maia, Luis Nogueira, Luis Miguel Pinho, and Daniel Gracia Pérez. A closer look into the aer model. In *Proc. of the IEEE 21st Int. Conf. on Emerging Technologies and Factory Automation (ETFA 2016)*, pages 1–8, 2016.

[97] William . Mangoua Sofack and Marc Boyer. Non preemptive static priority with network calculus. In *Proc. of the 16th IEEE Conference on Emerging Technologies Factory Automation (ETFA'2011)*, pages 1–8. IEEE, sept. 2011.

[98] William . Mangoua Sofack and Marc Boyer. Non preemptive static priority with network calculus : Enhancement. In *Proc. of the Workshop on Network Calculus (WoNeCa 2012)*, Kaiserslautern, Germany, March 2012.

[99] William Mangoua Sofack and Marc Boyer. Generalisation of gps and p-gps in network calculus. In *Proc. of the Work in Progress session of 9th IEEE Workshop on Factory Communication Systems (WFCS'2012)*, Lemgo/Detmold, Germany, May 21–24 2012.

[100] The Maude system home page. http ://maude.cs.uiuc.edu/.

[101] P. Meumeu Yomsi, D. Bertrand, N. Navet, and R.I. Davis. Controller area network (can) : Response time analysis with offsets. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 43–52, 2012.

[102] Aircraft internal time division command/response muliplex data bus. Technical Report MIL-STD-1553, United States Department of Defense, September 1978.

[103] A. Parekh and R. Gallager. A generalised processor sharing approach to flow control in integrated services networks : the single-node case. *IEEE transactions on networking*, June 1993.

[104] Lucien Rakotomalala, Marc Boyer, and Pierre Roux. Formal verification of real-time networks. In *Proc. of the 13th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2019)*, Toulouse, France, November 2019.

[105] K. Richter and R. Ernst. Event model interfaces for heterogeneous system analysis. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE'2002)*, pages 506 – 513, 2002.

[106] Jonas Rox and Rolf Ernst. Compositional performance analysis with improved analysis techniques for obtaining viable end-to-end latencies in distributed embedded systems. *International Journal on Software Tools for Technology Transfer*, 15(3) :171–187, 2013.

[107] Joan Adrià Ruiz de Azua and Marc Boyer. Complete modelling of AVB in network calculus framework. In *Proc. of the 22nd Int. Conf. on Real-Time Networks and Systems (RTNS 2014)*, Versailles, France, October 8-10 2014.

[108] Tarek Sadani, Marc Boyer, Pierre de Saqui-Sannes, and Jean-Pierre Courtiat. Effective representation of rt-lotos terms by finite time petri nets. In Elie Najm, Jean Pradat-Peyre, and Véronique Donzeau-Gouge, editors, *Proc. of the 26th IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE 2006)*, volume 4229 of *Lecture Notes in Computer Science*, pages 404–419. Springer Berlin / Heidelberg, 2006. 10.1007/11888116_29.

[109] Tarek Sadani, Marc Boyer, Pierre de Saqui-Sannes, and Jean-Pierre Courtiat. Mapping rt-lotos specifications into time petri nets. In *Proc. of 8th Int. Conf. on Formal Engineering Methods (ICFEM 2006)*, pages 360–379, Macao, China 2006.

[110] SAE. Time-triggered ethernet. Technical Report AS6802, SAE, 2011.

[111] Jens B. Schmitt and Frank A. Zdarsky. The DISCO network calculator - a toolbox for worst case analysis. In *Proc. of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), Pisa, Italy*, Pisa, Italy, November 2006. ACM.

[112] A. Soni, X. Li, J. L. Scharbarg, and C. Fraboul. Work in progress paper : pessimism analysis of network calculus approach on afdx networks. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–4, June 2017.

[113] Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Wctt analysis of avionics switched ethernet network with wrr scheduling. In *Proc. of the 26th International Conference on Real-Time Networks and Systems (RTNS)*, pages 213–222. ACM, 2018.

[114] Dimitrios Stiliadis and Anujan Varma. Latency-rate servers : A general model for analysis of traffic scheduling algorithms. *IEEE/ACM Trans. Netw.*, 6(5) :611–624, October 1998.

[115] Seyed Mohammadhossein Tabatabaee, Jean-Yves Le Boudec, and Marc Boyer. Interleaved weighted round-robin : A network calculus analysis. In *Proc. of the 32nd Int. Teletraffic Congress (ITC 32)*, September 22-24 2020.

[116] Rubén Trillo Flores and Marc Boyer. Performance analysis of the disrupted static priority scheduling for afdx. In *Proc. of the 12th ACM-IEEE Int. Conf. on Formal Methods and Models for System Design (MEMOCODE 14)*, 2014.

[117] Ernesto Wandeler and Lothar Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[118] Luxi Zhao, Qiao Li, Ying Xiong, Zhong Zheng, and Huagang Xiong. Using multi-link grouping technique to achieve tight latency in network calculus. In *Proc. of the 32nd IEEE/AIAA Digital Avionics Systems Conference (DASC 2013)*, pages 2E3–1–2E3–10, East Syracuse, NY, USA, Oct 2013.

[119] Luxi Zhao, Paul Pop, Zhong Zheng, Hugo Daigmorte, and Marc Boyer. Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus. *IEEE Transactions on Industrial Electronics*, 2020.